Report

v. 2.0

Customer

zkLink

Smart contract Audit

Mergetoken

1st July 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
1.1	01.07.24	A. Zveryanskaya	Title page fix
2.0	01.07.24	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 were asked to review commit e97f756 along with the related files:

governance/	
Governance.sol	IGovernance.sol
interfaces/	
IERC20MergeToken.sol	IMergeTokenPortal.sol
l1-contracts/bridge/interfaces/	
IL1Bridge.sol	IL2Bridge.sol

We have also checked the differences compared to the zklink-testnet:

l1-contracts/bridge/	
L1ERC20Bridge.sol	L1WethBridge.sol
l2-contracts/bridge/interfaces/	
IL2Bridge.sol	IMergeTokenPortal.sol
l2-contracts/bridge/	
L2ERC20Bridge.sol	L2WethBridge.sol
merge/	
ERC20MergeToken.sol	MergeTokenPortal.sol

The fixes were provided in 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 found 2 major, and a few less important issues.
All identified Major issues have been fixed.

Fixed 2 out of 2 issues

6 Major Issues

CVF-1. FIXED

- **Category** Unclear behavior
- **Source** MergeTokenPortal.sol

Description This function allows removing non-existing tokens.

Recommendation Consider explicitly forbidding such usage.

```
115 115 function removeSourceToken(address _sourceToken) external
        ↪ onlyOwnerOrSecurityCouncil {
```

CVF-2. FIXED

- **Category** Unclear behavior
- **Source** IERC20MergeToken.sol

Description This function looks like it allows burning tokens from other's accounts. This requires some kind of authorization that should be clearly described.

```
8 8 /// @notice Burn merge token
9 9 function burn(address _from, uint256 _amount) external;
```

7 Moderate Issues

CVF-3. INFO

- **Category** Suboptimal
- **Source** Governance.sol

Description Using the timestamp field to mark executed operations makes code more complicated and more error prone. Also, it hides original timestamp values for executed operations, making investigations more complicated.

Recommendation Consider using a separate flag for marking executed operations. It is possible to fit both, a timestamp and a flag into a single storage slot.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit*

```
30 30  /// @dev - 1 (EXECUTED_PROPOSAL_TIMESTAMP) means the operation is
          ↪ already executed.
```

CVF-4. INFO

- **Category** Unclear behavior
- **Source** Governance.sol

Recommendation A more conventional way to prevent reentrancy would be to mark the operation as executed before actually executing it.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit*

```
175 175  // Reconfirming that the operation is still ready after execution.
176 176  // This is needed to avoid unexpected reentrancy attacks of re-
            ↪ executing the same operation.
177 177  require(isOperationReady(id), "Operation␣must␣be␣ready␣after␣
            ↪ execution");
178 178  // Set operation to be done
179 179  timestamps[id] = EXECUTED_PROPOSAL_TIMESTAMP;
```

CVF-5. INFO

- **Category** Unclear behavior
- **Source** Governance.sol

Description In case of a successful execution, the returned data is dropped, which could make investigations harder.

Recommendation Consider emitting an even with the returned data.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit*

```
226 226 (bool success, bytes memory returnData) = _calls[i].target.call{
        ↪ value: _calls[i].value}(_calls[i].data);
```

CVF-6. INFO

- **Category** Suboptimal
- **Source** Governance.sol

Recommendation The error, returned in case of a failed execution, should include the index of the failed call to simplify investigations.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit*

```
230 230 revert(add(returnData, 0x20), mload(returnData))
```

8 Recommendations

CVF-7. INFO

- **Category** Bad datatype
- **Source** IL2Bridge.sol

Recommendation The type for these parameters should be more specific.

Client Comment *We do not intend to modify.*

```
19 +address indexed l2Token,
20 +address mergeToken,
```

CVF-8. INFO

- **Category** Bad datatype
- **Source** IL2Bridge.sol

Recommendation The type for this argument should be more specific.

Client Comment *We do not intend to modify.*

```
42 +address _l1Token,
```

CVF-9. INFO

- **Category** Procedural
- **Source** IMergeTokenPortal.sol

Recommendation Specifying a particular compiler version makes it harder migrating to newer versions. Also this version requirement is inconsistent with other files in the same code base. Consider specifying as "^0.8.0".

Client Comment *All solidity files in l2-contracts use 0.8.20.*

```
2 +pragma solidity 0.8.20;
```

CVF-10. INFO

- **Category** Bad datatype
- **Source** IMergeTokenPortal.sol

Recommendation The type for this field should be more specific.

Client Comment *We do not intend to modify.*

```
13 +address mergeToken;
```

CVF-11. FIXED

- **Category** Documentation
- **Source** IMergeTokenPortal.sol

Description This field is not documented.

Recommendation Consider documenting.

```
13 +address mergeToken;
```

CVF-12. INFO

- **Category** Bad datatype
- **Source** IMergeTokenPortal.sol

Recommendation The type for the "sourceToken" arguments should be more specific.

Client Comment *We do not intend to modify.*

```
23 +function getSourceTokenInfos(address _sourceToken) external view
   ↪ returns (SourceTokenInfo memory);
```

```
31 +function deposit(address _sourceToken, uint256 _amount, address
   ↪ _receiver) external;
```

```
39 +function withdraw(address _sourceToken, uint256 _amount, address
   ↪ _receiver) external;
```

CVF-13. INFO

- **Category** Bad datatype
- **Source** L2ERC20Bridge.sol

Recommendation The type for the token arguments should be more specific.

Client Comment *We do not intend to modify.*

```
98  +    address _l1Token,
```

```
117 +function depositToMerge(address _l2Token, uint256 _amount, address
    ↪ _l2Receiver) external {
```

```
124 +function _getExpectedL2Token(address _l1Token, bytes calldata _data
    ↪ ) internal returns (address) {
```

```
169 +function withdrawFromMerge(address _l1Receiver, address _l2Token,
    ↪ uint256 _amount) external {
```

```
180 +function _sendWithdrawalMessage(address _l1Receiver, address
    ↪ _l2Token, uint256 _amount) internal {
```

CVF-14. INFO

- **Category** Bad datatype
- **Source** L2ERC20Bridge.sol

Recommendation The return type should be "IL2StandardToken".

Client Comment *We do not intend to modify.*

```
124 +function _getExpectedL2Token(address _l1Token, bytes calldata _data
    ↪ ) internal returns (address) {
```

CVF-15. INFO

- **Category** Bad datatype
- **Source** L2WethBridge.sol

Recommendation The type for this argument should be more specific.

Client Comment *We do not intend to modify.*

```
104 +address, // _l1Token,
```

CVF-16. INFO

- **Category** Bad datatype
- **Source** IL1Bridge.sol

Recommendation The type for this field should be more specific.

Client Comment *We do not intend to modify.*

```
23 +address l1Token,
```

```
45 +address _l1Token,
```

CVF-17. INFO

- **Category** Bad datatype
- **Source** IL2Bridge.sol

Recommendation The type for this argument should be more specific.

Client Comment *We do not intend to modify.*

```
18 +address _l1Token,
```

CVF-18. INFO

- **Category** Bad datatype
- **Source** L1ERC20Bridge.sol

Recommendation The type for these arguments should be more specific.

Client Comment *We do not intend to modify.*

```
223 +address _l1Token,
```

```
242 +address _l1Token,
```

CVF-19. INFO

- **Category** Bad datatype
- **Source** L1WethBridge.sol

Recommendation The type for this argument should be more specific.

Client Comment *We do not intend to modify.*

```
199 +address, // _l1Token,
```

CVF-20. INFO

- **Category** Bad datatype
- **Source** ERC20MergeToken.sol

Recommendation The type for this variable should be more specific.

Client Comment *We do not intend to modify.*

```
8  8   address public immutable MERGE_TOKEN_PORTAL;
```

CVF-21. INFO

- **Category** Bad datatype
- **Source** ERC20MergeToken.sol

Recommendation The type for this argument should be more specific.

Client Comment *We do not intend to modify.*

```
17 17  address _mergeTokenPortal,
```

CVF-22. INFO

- **Category** Bad datatype
- **Source** MergeTokenPortal.sol

Recommendation The key type for this mapping should be more specific.

Client Comment *We do not intend to modify.*

```
16 16  mapping(address sourceToken => SourceTokenInfo) public
       ↪ sourceTokenInfoMap;
```

CVF-23. INFO

- **Category** Bad naming
- **Source** MergeTokenPortal.sol

Description Despite the name, this function returns just one token info.

Recommendation Consider renaming.

Client Comment *We do not intend to modify.*

```
56 56  function getSourceTokenInfos(address _sourceToken) public view
       ↪ override returns (SourceTokenInfo memory) {
```

CVF-24. INFO

- **Category** Bad datatype
- **Source** MergeTokenPortal.sol

Recommendation The type for the "_sourceToken" arguments should be more specific.

Client Comment *We do not intend to modify.*

```
56 56  function getSourceTokenInfos(address _sourceToken) public view
          ↪ override returns (SourceTokenInfo memory) {

61 61  function deposit(address _sourceToken, uint256 _amount, address
          ↪ _receiver) external override nonReentrant {

80 80  function withdraw(address _sourceToken, uint256 _amount, address
          ↪ _receiver) external override nonReentrant {

99 99  function addSourceToken(address _sourceToken, address _mergeToken,
          ↪ uint256 _depositLimit) external onlyOwner {

115 115  function removeSourceToken(address _sourceToken) external
          ↪ onlyOwnerOrSecurityCouncil {

124 124  function updateDepositStatus(address _sourceToken, bool _isLocked)
          ↪ external onlyOwnerOrSecurityCouncil {

134 134  function setDepositLimit(address _sourceToken, uint256 _limit)
          ↪ external onlyOwner {
```

CVF-25. INFO

- **Category** Bad datatype
- **Source** MergeTokenPortal.sol

Recommendation The type for the "_mergeToken" argument should be more specific.

Client Comment *We do not intend to modify.*

```
99 99  function addSourceToken(address _sourceToken, address _mergeToken,
          ↪ uint256 _depositLimit) external onlyOwner {
```

CVF-26. FIXED

- **Category** Procedural
- **Source** MergeTokenPortal.sol

Recommendation This check should be performed earlier.

```
102 102 require(_sourceToken != address(0) && _mergeToken != address(0), "
        ↪ Invalid␣token␣address");
```

CVF-27. FIXED

- **Category** Unclear behavior
- **Source** MergeTokenPortal.sol

Description This event is emitted even if nothing actually changed.

```
130 130 emit SourceTokenLocked(_sourceToken, _isLocked);
```

```
140 140 emit DepositLimitUpdated(_sourceToken, _limit);
```

CVF-28. FIXED

- **Category** Procedural
- **Source** IERC20MergeToken.sol

Description Specifying a particular compiler version makes it harder migrating to newer versions.

Recommendation Consider specifying as ”^0.8.0”. Also relevant for: IMergeTokenPortal.sol.

```
2 2 pragma solidity 0.8.23;
```

CVF-30. INFO

- **Category** Bad datatype
- **Source** IMergeTokenPortal.sol

Recommendation The type for the "sourceToken" and "mergeToken" parameters should be more specific.

Client Comment *We do not intend to modify.*

```
6  6      address indexed sourceToken,
7  7      address mergeToken,
```

```
14 14     address indexed sourceToken,
15 15     address mergeToken,
```

```
21 21 event SourceTokenAdded(address indexed sourceToken, address
      ↪ mergeToken, uint256 depositLimit);
```

```
23 23 event SourceTokenRemoved(address indexed sourceToken);
```

```
25 25 event SourceTokenLocked(address indexed sourceToken, bool isLocked);
```

```
27 27 event DepositLimitUpdated(address indexed sourceToken, uint256
      ↪ depositLimit);
```

CVF-31. INFO

- **Category** Suboptimal
- **Source** IMergeTokenPortal.sol

Recommendation Some of the address parameters should probably be indexed.

Client Comment *We do not intend to modify.*

```
7  7      address mergeToken,
8  8      address sender,
```

```
10 10     address receiver
```

```
15 15     address mergeToken,
16 16     address sender,
```

```
18 18     address receiver
```

```
21 21 event SourceTokenAdded(address indexed sourceToken, address
      ↪ mergeToken, uint256 depositLimit);
```

CVF-32. INFO

- **Category** Bad naming
- **Source** IMergeTokenPortal.sol

Recommendation Events are usually named via nouns, such as "SourceToken" or "SourceTokenRemoval"..

Client Comment *We do not intend to modify.*

```
21 21 event SourceTokenAdded(address indexed sourceToken, address
      ↪ mergeToken, uint256 depositLimit);

23 23 event SourceTokenRemoved(address indexed sourceToken);

25 25 event SourceTokenLocked(address indexed sourceToken, bool isLocked);

27 27 event DepositLimitUpdated(address indexed sourceToken, uint256
      ↪ depositLimit);

29 29 event SecurityCouncilUpdated(address indexed oldCommitee, address
      ↪ indexed newCommitee);
```

CVF-33. INFO

- **Category** Suboptimal
- **Source** IMergeTokenPortal.sol

Recommendation The "oldCommitee" paramter is redundant, as its value could be derived from previous events.

Client Comment *We do not intend to modify.*

```
29 29 event SecurityCouncilUpdated(address indexed oldCommitee, address
      ↪ indexed newCommitee);
```

CVF-34. INFO

- **Category** Bad datatype
- **Source** IMergeTokenPortal.sol

Recommendation The type for this field should be more specific.

Client Comment *We do not intend to modify.*

```
39 39 address mergeToken;
```

CVF-35. INFO

- **Category** Bad datatype
- **Source** IMergeTokenPortal.sol

Recommendation The type for the "sourceToken" arguments should be more specific.

Client Comment *We do not intend to modify.*

```
49 49 function getSourceTokenInfos(address _sourceToken) external view
      ↪ returns (SourceTokenInfo memory);
```

```
57 57 function deposit(address _sourceToken, uint256 _amount, address
      ↪ _receiver) external;
```

```
65 65 function withdraw(address _sourceToken, uint256 _amount, address
      ↪ _receiver) external;
```

CVF-36. FIXED

- **Category** Procedural
- **Source** IGovernance.sol

Recommendation These commented out functions should be removed.

```
50 50 // function scheduleShadow(bytes32 _id, uint256 _delay) external;
```

```
56 56 // function executeInstant(Operation calldata _operation) external
      ↪ payable;
```

CVF-37. INFO

- **Category** Bad naming
- **Source** IGovernance.sol

Recommendation Events are usually named via nouns, such as "TransparentOperation" or "ShadowOperation".

Client Comment *We do not intend to modify.*

```
65 65  event TransparentOperationScheduled(bytes32 indexed _id, uint256
          ↪ delay, Operation _operation);

68 68  event ShadowOperationScheduled(bytes32 indexed _id, uint256 delay);

71 71  event OperationExecuted(bytes32 indexed _id);

74 74  event ChangeSecurityCouncil(address _securityCouncilBefore, address
          ↪ _securityCouncilAfter);

77 77  event ChangeMinDelay(uint256 _delayBefore, uint256 _delayAfter);

80 80  event OperationCancelled(bytes32 indexed _id);
```

CVF-38. FIXED

- **Category** Suboptimal
- **Source** IGovernance.sol

Recommendation The parameters should be indexed.

```
74 74  event ChangeSecurityCouncil(address _securityCouncilBefore, address
          ↪ _securityCouncilAfter);
```

CVF-39. INFO

- **Category** Procedural
- **Source** IGovernance.sol

Recommendation The old value parameters are redundant, as they could be derived from previous events.

Client Comment *We do not intend to modify.*

```
74 74  event ChangeSecurityCouncil(address _securityCouncilBefore, address
          ↪ _securityCouncilAfter);

77 77  event ChangeMinDelay(uint256 _delayBefore, uint256 _delayAfter);
```

CVF-40. INFO

- **Category** Suboptimal
- **Source** Governance.sol

Recommendation This unction would be simpler if an operation would have a separate state property along with a timestamp.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit.*

```
105 105 function getOperationState(bytes32 _id) public view returns (
        ↪ OperationState) {
```

CVF-41. FIXED

- **Category** Procedural
- **Source** Governance.sol

Recommendation These commented out functions should be removed.

```
142 142 // function scheduleShadow(bytes32 _id, uint256 _delay) external
        ↪ onlyOwner {
```

```
186 186 // function executeInstant(Operation calldata _operation) external
        ↪ payable onlySecurityCouncil {
```

CVF-42. INFO

- **Category** Suboptimal
- **Source** Governance.sol

Description Cancelling an operation by deleting its timestamp hides original timestamps of cancelled operations.

Recommendation Consider using a separate state property to mark an operation as cancelled.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit.*

```
156 156 delete timestamps[_id];
```

CVF-43. INFO

- **Category** Unclear behavior
- **Source** Governance.sol

Description This event is emitted even if nothing actually changed.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit.*

```
250 250 emit ChangeMinDelay(minDelay, _newDelay);
```

```
257 257 emit ChangeSecurityCouncil(securityCouncil, _newSecurityCouncil);
```

CVF-44. INFO

- **Category** Procedural
- **Source** Governance.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *We do not plan to modify this code. This code is copied from zksync's Governance. The audit report is from https://blog.openzeppelin.com/december-diff-and-governance-audit.*

```
262 262 receive() external payable {}
```

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