// Security Assessment

07.08.2025 - 07.09.2025

RLC Multichain Bridge

iExec

HALBORN

RLC Multichain Bridge - iExec

Prepared by: HALBORN

Last Updated 08/14/2025

Date of Engagement: July 8th, 2025 - July 9th, 2025

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	0	0	0	0	2

TABLE OF CONTENTS

1. Introduction

`iExec` engaged `Halborn` to conduct a security assessment on their smart contracts beginning on July 8th, 2025 and ending on July 9th, 2025. The security assessment was scoped to the smart contracts provided to Halborn. Commit hashes and further details can be found in the Scope section of this report.

The `RLC Multichain Bridge` codebase in scope consists of a few different smart contracts, responsible for bridging tokens across chains, using LayerZero.

2. Assessment Summary

`Halborn` was provided 2 days for the engagement and assigned a full-time security engineer to review the security of the smart contracts in scope.

The purpose of the assessment is to:

- Identify potential security issues within the smart contracts.
- Ensure that smart contract functionality operates as intended.

In summary, `Halborn` did not identify any significant security risks. However, some enhancements were recommended to improve code clarity, consistency, or maintainability. These were partially addressed by the `iExec team`:

- `Consider using a fixed pragma version.`
- `Consider fixing the typographical errors in the codebase.`

3. Caveats

After the initial assessment, the `iExec` team reported a role management issue affecting the `IexecLayerZeroBridge` file. Halborn performed a post-assessment review at pull request PR77 to verify the fixes, and confirmed that the issue has been addressed.

4. Test Approach And Methodology

`Halborn` performed a manual review of the code. Manual testing is great to uncover flaws in logic, process, and implementation.

The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture, purpose and use of the platform.
- Smart contract manual code review and walkthrough to identify any logic issue.
- Thorough assessment of safety and usage of critical Solidity variables and functions in scope that could led to arithmetic related vulnerabilities.

5. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

5.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

5.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

5.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

6. SCOPE

REPOSITORY

(a) Repository: rlc-multichain

(b) Assessed Commit ID: b5e157f

(c) Items in scope:

- RLCCrosschainToken.sol
- RLCLiquidityUnifier.sol
- DualPausableUpgradeable.sol
- IexecLayerZeroBridge.sol
- IIexecLayerZeroBridge.sol
- IRLCLiquidityUnifier.sol
- ITokenSpender.sol

Out-of-Scope: External dependencies and economic attacks.

REMEDIATION COMMIT ID:

- 82c8c45

Out-of-Scope: New features/implementations after the remediation commit IDs.

7. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
FLOATING PRAGMA	INFORMATIONAL	ACKNOWLEDGED - 08/13/2025
TYPOGRAPHICAL ERROR	INFORMATIONAL	SOLVED - 07/15/2025

8. FINDINGS & TECH DETAILS

8.1 FLOATING PRAGMA

// INFORMATIONAL

Description

Currently, all of the in-scope contracts have a floating pragma version. This is generally considered a bad practice and instead, a fixed pragma version should be used.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Consider using a fixed pragma version.

Remediation Comment

ACKNOWLEDGED: The `iExec` team made a business decision to acknowledge this finding and not alter the contracts.

8.2 TYPOGRAPHICAL ERROR

// INFORMATIONAL

Description

`IexecLayerZeroBridge` and `DualPausableUpgradeable` use the word `outbount` instead of `outbound` in a few places.

BVSS

AO:A/AC:L/AX:L/R:N/S:U/C:N/A:N/I:N/D:N/Y:N (0.0)

Recommendation

Consider fixing the typographical errors.

Remediation Comment

SOLVED: The `iExec` team solved this finding in the specified commit by following the mentioned recommendation.

Remediation Hash

https://github.com/iExecBlockchainComputing/rlc-multichain/commit/82c8c45f11a8928bb400c7d5059f3e7f7d1c1e0c

---