HACKEN

# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Router Protocol
**Date**: May 30th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io



## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for Router Protocol. |
| **Approved By** | Evgeniy Bezuglyi \| SC Department Head at Hacken OU |
| **Type** | SDK Library |
| **Platform** | EVM |
| **Language** | Solidity |
| **Methods** | Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| **Website** | https://www.routerprotocol.com/ |
| **Timeline** | 06.05.2022 - 30.05.2022 |
| **Changelog** | 23.05.2022 - Initial Review<br>30.05.2022 - Second Review |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Introduction

Hacken OÜ (Consultant) was contracted by Router Protocol (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

## Scope

The scope of the project is smart contracts in the repository:

### Initial review scope

**Repository:**
https://github.com/router-protocol/router-crosstalk
**Commit:**
2429c9e20c3c2517d77cd559cc60273d2002ea04
**Technical Documentation:** No
**JS tests:** Yes
**Contracts:**
nonupgradeable/testnets/RouterCrossTalkMumbai.sol
(sha3: 484555cc1daed4063e0c9bea9564696f59da172fd332eab19e8eaf33)
nonupgradeable/RouterCrossTalk.sol
(sha3: 2b9f5d0996da0bcd087f1ae4847b7d33a7c5a5b3d2c1da5d05c9b83b)
upgradable/RouterCrossTalkUpgradeable.sol
(sha3: dc7085fee13a4bca81e34d3f59f9fc7f9ecbfb216c9f78f5027f60f0)

### Second review scope

**Repository:**
https://github.com/router-protocol/router-crosstalk
**Commit:**
479e3c10823d41f6d337f4ccbc6ef8af9232fd82
**Technical Documentation:** No
**JS tests:** Yes
**Contracts:**
RouterCrossTalkUpgradeable.sol
(sha3: 52572b13a1b62394de5d35ba6e29e4bd3106f5ba4dfd5b3bc3d4bc36)
RouterCrossTalk.sol
(sha3: 21f1c98958a0396d008a0a6d04e93b94767d7d00ce1fafc36bebb754)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations. |
| Low | Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Executive Summary

The score measurement details can be found in the corresponding section of the methodology.

## Documentation quality

The Customer provided neither functional requirements nor technical requirements. The total Documentation Quality score is **0** out of **10**.

## Code quality

The total CodeQuality score is **5** out of **10**. No NatSpec in the code. No unit tests were provided. Not following code-style guidelines.

## Architecture quality

The architecture quality score is **8** out of **10**. The architecture overall is clear but the functionality is duplicated.

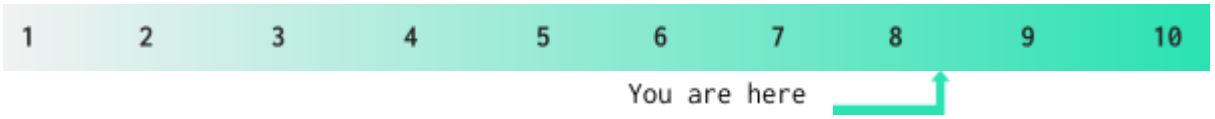
## Security score

As a result of the audit, security engineers found **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

## Summary

According to the assessment, the Customer's smart contract has the following score: **8.3**



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are considered:

| Item | Type | Description | Status |
|---|---|---|---|
| **Default Visibility** | SWC-100<br>SWC-108 | Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously. | Passed |
| **Integer Overflow and Underflow** | SWC-101 | If unchecked math is used, all math operations should be safe from overflows and underflows. | Passed |
| **Outdated Compiler Version** | SWC-102 | It is recommended to use a recent version of the Solidity compiler. | Passed |
| **Floating Pragma** | SWC-103 | Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. | Failed |
| **Unchecked Call Return Value** | SWC-104 | The return value of a message call should be checked. | Not Relevant |
| **Access Control & Authorization** | CWE-284 | Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users. | Passed |
| **SELFDESTRUCT Instruction** | SWC-106 | The contract should not be destroyed until it has funds belonging to users. | Not Relevant |
| **Check-Effect-Interaction** | SWC-107 | Check-Effect-Interaction pattern should be followed if the code performs ANY external call. | Passed |
| **Uninitialized Storage Pointer** | SWC-109 | Storage type should be set explicitly if the compiler version is < 0.5.0. | Not Relevant |
| **Assert Violation** | SWC-110 | Properly functioning code should never reach a failing assert statement. | Not Relevant |
| **Deprecated Solidity Functions** | SWC-111 | Deprecated built-in functions should never be used. | Passed |
| **Delegatecall to Untrusted Callee** | SWC-112 | Delegatecalls should only be allowed to trusted addresses. | Passed |
| **DoS (Denial of Service)** | SWC-113<br>SWC-128 | Execution of the code should never be blocked by a specific contract state unless it is required. | Passed |

| | | | |
|---|---|---|---|
| **Race Conditions** | SWC-114 | Race Conditions and Transactions Order Dependency should not be possible. | Passed |
| **Authorization through tx.origin** | SWC-115 | tx.origin should not be used for authorization. | Passed |
| **Block values as a proxy for time** | SWC-116 | Block numbers should not be used for time calculations. | Passed |
| **Signature Unique Id** | SWC-117 SWC-121 SWC-122 | Signed messages should always have a unique id. A transaction hash should not be used as a unique id. | Passed |
| **Shadowing State Variable** | SWC-119 | State variables should not be shadowed. | Passed |
| **Weak Sources of Randomness** | SWC-120 | Random values should never be generated from Chain Attributes. | Passed |
| **Incorrect Inheritance Order** | SWC-125 | When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. | Passed |
| **Calls Only to Trusted Addresses** | EEA-Level-2 SWC-126 | All external calls should be performed only to trusted addresses. | Passed |
| **Presence of unused variables** | SWC-131 | The code should not contain unused variables if this is not justified by design. | Passed |
| **EIP standards violation** | EIP | EIP standards should not be violated. | Not Relevant |
| **Assets integrity** | **Custom** | Funds are protected and cannot be withdrawn without proper permissions. | Passed |
| **User Balances manipulation** | **Custom** | Contract owners or any other third party should not be able to access funds belonging to users. | Passed |
| **Data Consistency** | **Custom** | Smart contract data should be consistent all over the data flow. | Passed |
| **Flashloan Attack** | **Custom** | When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used. | Not Relevant |
| **Token Supply manipulation** | **Custom** | Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer. | Passed |

| | | | |
|---|---|---|---|
| **Gas Limit and Loops** | **Custom** | Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit. | Passed |
| **Style guide violation** | **Custom** | Style guides and best practices should be followed. | Failed |
| **Requirements Compliance** | **Custom** | The code should be compliant with requirements provided by the Customer, | Passed |
| **Repository Consistency** | **Custom** | The repository should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code. | Passed |
| **Tests Coverage** | **Custom** | The code should be covered with unit tests. Tests coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested. | Passed |

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

## System Overview

**Router Crosstalk** is an SDK library to work with the Router Bridge contract, which contains the following contracts:

- RouterCrossTalk.sol — an abstract contract that allows communicating with `iGenericHandler` contracts programmatically.
  It has the following attributes:
  - handler: the `iGenericHandler` contract to communicate with;
  - linkSetter: Address of linker;
  - feeToken: Token used in the `genericDeposit` function;
  - Chain2Addr: The mapping of link addresses per chain id.
- RouterCrossTalkUpgradeable.sol — an upgradable abstract contract that allows communicating with `iGenericHandler` contracts programmatically.
  It has the following attributes:
  - handler: the `iGenericHandler` contract to communicate with;
  - linkSetter: Address of linker;
  - feeToken: Token used in the `genericDeposit` function;
  - Chain2Addr: The mapping of link addresses per chain id.

### Privileged roles

- The only privileged role for all contracts is the `handler`. Only the `handler` role is allowed to call the `routerSync` function.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Findings

## Critical

No critical severity issues were found.

## High

### 1. Tests failing

The only test provided is failing.

**Scope**: test

**Recommendation**: ensure tests are running successfully and the code coverage is not less than 95%.

**Status:** Fixed (Revised Commit: 479e3c1)

## Medium

No medium severity issues were found.

## Low

### 1. Visibility is not declared explicitly

The visibility of variables should be declared explicitly. While the implicit visibility declaration is “internal” that could be not self-explanatory to some reviewers or developers.

**Contracts**: RouterCrossTalkMumbai.sol, RouterCrossTalk.sol, RouterCrossTalkUpgradeable.sol

**Variable**: handler

**Recommendation**: declare the visibility of state variables explicitly.

**Status:** Fixed (Revised Commit: 479e3c1)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

# Disclaimers

## Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

## Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited smart contracts.