



Security Assessment Report

Router Protocol

ERC20 contract Audit

Version: Final ▾

Date: 15 Jul 2024



Table of Contents

Table of Contents	1
License	2
Disclaimer	3
Introduction	4
Codebases Submitted for the Audit	5
How to Read This Report	6
Overview	7
Methodology	7
Functionality Overview	7
Summary of Findings	8
Detailed Findings	9

License

THIS WORK IS LICENSED UNDER A [CREATIVE COMMONS ATTRIBUTION-NODERIVATIVES 4.0 INTERNATIONAL LICENSE](#).

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

COPYRIGHT OF THIS REPORT REMAINS WITH THE AUTHOR.

Introduction

Purpose of this report

0xCommit has been engaged by **Router Protocol** to perform a security audit of several Token Migration contract components.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behaviour.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following commits:

Github Link:

<https://github.com/router-protocol/asset-forwarder-contracts/blob/depositPause/evm/contracts/UpgradeableERC20.sol>

Version	Commit hash
Initial	feb379f4964b9d685bf49af305982f56c0b00d27
Final	6b4dec003910e00ebfa4e4a504bfad0172a61ce6

Network	Address
Ethereum mainnet	0x60F67E1015b3f069DD4358a78c38f83fE3a667A9
Polygon	0x93890f346C5D02C3863a06657bc72555dC72c527

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
High	An attacker can successfully execute an attack that clearly results in operational issues for the service. This also includes any value loss of unclaimed funds permanently or temporary.
Medium	The service may be susceptible to an attacker carrying out an unintentional action, which could potentially disrupt its operation. Nonetheless, certain limitations exist that make it difficult for the attack to be successful.
Low	The service may be vulnerable to an attacker executing an unintended action, but the impact of the action is negligible or the likelihood of the attack succeeding is very low and there is no loss of value.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary

The status of an issue can be one of the following: **Pending**, **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Overview

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
 - a. Race condition analysis
 - b. Under-/overflow issues
 - c. Key management vulnerabilities
 - d. Access Control Issues
 - e. Boundary Analysis
4. Report preparation

Functionality Overview

The token migration contract works to migrate from one token to a newer version by allowing users to deposit old tokens and receive the new tokens.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Open zeppelin packages are not hardcoded	Low ▾	Resolved ▾

Detailed Findings

1. Open-Zeppelin packages are not hardcoded

Severity: Low ▾

Description

The package version used for making upgradeable contracts is not hardcoded. It needs to be defined which version of dependency needs to be used for current implementation.

Remediation

Remove “^” from open zeppelin upgradeable contracts dependency in package.json to make sure the versions of the package are fixed for the upgradeable contracts.

Status

Resolved ▾