



Security Assessment Report



Dzap IO

Dzap IO core contracts Audit

Version: Final ▾

Date: 12 Jun 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
1. Upgrade solidity compiler version	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 **Dzap IO** to perform a security audit of several bridge 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).

Codebases Submitted for the Audit

The audit has been performed on the following commits:

Github Link: <https://github.com/DZapIO/DZapCoreContracts/>

Version	Commit hash
Initial	6cbac236221eb77f735a176c064c9301b3111290
Final	aa0c00bfa5ac6c5f799feb4911a7a2193d8ad1cf

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 Dzap bridging contracts utilize a diamond proxy pattern, allowing the admin to whitelist bridges and decentralized exchanges (DEXes). The team has put significant effort into ensuring the contracts are gas-efficient.

Summary of Findings

Sr. No.	Description	Severity	Status
1	Upgrade solidity compiler version	Informational ▾	Acknowl... ▾

Detailed Findings

1. Upgrade solidity compiler version

Severity: **Informational** ▾

Description

Using Solidity compiler version 0.8.19 may expose the project to potential vulnerabilities due to missing bug fixes in later versions. Regular updates and improvements are made in each new release of the Solidity compiler.

Remediation

To ensure the project benefits from the latest security enhancements and bug fixes, it is recommended to upgrade to Solidity compiler version 0.8.22 or later.

Status

Acknowledged ▾