0xCommit

Security Audit Report

DZap

Version: Final

Date: 18th Aug 2025

Table of Contents

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.

Introduction

Purpose of this report

0xCommit has been engaged by **DZap -Contracts** to perform a security audit of several Solana Programs 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 solana program 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 GitHub repositories:

Version	Commit ID
Final	52a0a7a814a1d8a6ecd4973290e39a8f0f6f2969

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.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
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. This category may also include opinionated recommendations that the project team might not share.

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
4. Report preparation

Summary of Findings

Sr. No.	Description	Severity	Status
1	Reentrancy in BridgeFacet with External Calls	Critical	Resolved
2	Fee Calculation Overflow Risk	Medium	Acknowledged
3	Insufficient Event Emission	Low	Resolved
4	Missing Zero Address Checks	High	Resolved
5	Inconsistent Error Handling	Medium	Resolved

Detailed Findings

Reentrancy in BridgeFacet with External Calls

Location: `contracts/Bridge/Facets/BridgeFacet.sol`

Description: Multiple bridge functions execute external calls without reentrancy protection, particularly in swap execution before bridge operations.

Impact: Reentrancy attacks could drain funds or manipulate bridge state.

Recommendation:

1. Add reentrancy guards to all external-facing functions
2. Follow checks-effects-interactions pattern
3. Use OpenZeppelin's ReentrancyGuard

Resolved

Medium Severity Issues

Fee Calculation Overflow Risk

Location: `contracts/Shared/Libraries/LibBridge.sol`

Description: Fee calculations don't check for overflow in token amounts.

Impact: Incorrect fee calculations, potential fund loss.

Recommendation:

```
// Use SafeMath or Solidity 0.8+ automatic checks
uint256 feeAmount = (amount * feePercent) / 10000;
require(feeAmount <= amount, "Fee overflow");
```

Resolved

Low Severity Issues

Insufficient Event Emission

Location: Multiple contracts

Description: Critical operations missing event emissions.

Following are the locations where Event emission is missing -

- DiamondInit.initialize()
- Token Wrapper functions
- Fee Collection operations
- Refund operations
- Dust Sweeping operations
- LibAsset Transfer Functions
- Adapter updates in libbridge
- WithdrawFacuet.executeCallandWithdraw()
- Accesscontrol operations
- Dlamondcut operations
- Permit Operation
- Failed Swap Handing

Impact: Difficult to track protocol state, reduced transparency.

Recommendation: Add comprehensive event logging.

Resolved

Missing Zero Address Checks

Location: Various functions

Description: Some functions don't validate against zero addresses.

Following are the locations where Address checks are missing -

- OwnershipFacet.transferOwnership()
- DirectTransferAdapter.bridgeViaTransfer()
- GenericBridgeAdapter.bridgeViaGeneric()
- LibAsset.transferFromERC20()
- AccessManagerFacet.setCanExecute()

Recommendation: Add comprehensive input validation.

Resolved

Inconsistent Error Handling

Location: Throughout codebase

Description: Mix of required statements and custom errors.

Recommendation: Standardize on custom errors for gas efficiency.

Resolved