LayerZero OFT

Audit

Presented by:

OtterSec contact@osec.io

Robert Chen r@osec.io

Naveen Kumar J naina@osec.io

Contents

Appendices

01 | Executive Summary

Overview

LayerZero engaged OtterSec to perform an assessment of `oft/v2` from their `solidity-examples` program under our retainer. This assessment was conducted between November 11th and November 15th, 2022.

Critical vulnerabilities were communicated to the team prior to the delivery of the report to speed up remediation. After delivering our audit report, we worked closely with the team to streamline patches and confirm remediation. We delivered final confirmation of the patches November 17th, 2022.

02 | Scope

The source code was delivered to us in a git repository at github.com/LayerZero-Labs/solidity-examples. This audit was performed against commit `9d9bf12`.

A brief description of the programs is as follows.

Name	Description
solidity-examples	Solidity code examples building on top of LayerZero

03 | Findings

Overall, we report 1 findings.

We split the findings into **vulnerabilities** and **general findings**. Vulnerabilities have an immediate impact and should be remediated as soon as possible. General findings don't have an immediate impact but will help mitigate future vulnerabilities.

04 | General Findings

Here we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent antipatterns and could lead to security issues in the future.

ID	Description
OS-LZR-SUG-00	Reorder OFT payload fields to mitigate the impact of variable length decoding

OS-LZR-SUG-00 | Reorganize Payload Fields

Description

Currently, the OFT payload is encoded via

```
eturn abi.encodePacked(
            PT_SEND_AND_CALL,
            _toAddress,
            _amountSD,
            _addressToBytes32(_from),
            _dstGasForCall,
            _payload
        );
```

Note that the security of this encoding depends on the correct size of `_toAddress`. Otherwise, the critical field `_amountSD` could get confused with bytes of the `_from` field.

Remediation

As a defense in-depth measure, it could make sense to reorder the fields. More specifically, putting `_amountSD` first in the encoding will likely mitigate future variable length encoding attacks.

A | Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings can be found in the General Findings section.

Critical	Vulnerabilities that immediately lead to loss of user funds with minimal preconditions Examples: • Misconfigured authority or access control validation • Improperly designed economic incentives leading to loss of funds
High	Vulnerabilities that could lead to loss of user funds but are potentially difficult to exploit. Examples: • Loss of funds requiring specific victim interactions • Exploitation involving high capital requirement with respect to payout
Medium	Vulnerabilities that could lead to denial of service scenarios or degraded usability. Examples: • Malicious input that causes computational limit exhaustion • Forced exceptions in normal user flow
Low	Low probability vulnerabilities which could still be exploitable but require extenuating circumstances or undue risk. Examples: • Oracle manipulation with large capital requirements and multiple transactions
Informational	Best practices to mitigate future security risks. These are classified as general findings. Examples: • Explicit assertion of critical internal invariants • Improved input validation