Prepared for
Ryan Zarick
Isaac Zhang
LayerZero Labs

Prepared by
Jasraj Bedi
Aaron Esau
Zellic

June 12, 2024

LayerZero OApp & OFT

Smart Contract Security Assessment

Contents

About Zellic

Zellic is a vulnerability research firm with deep expertise in blockchain security. We specialize in EVM, Move (Aptos and Sui), and Solana as well as Cairo, NEAR, and Cosmos. We review L1s and L2s, cross-chain protocols, wallets and applied cryptography, zero-knowledge circuits, web applications, and more.

Prior to Zellic, we founded the #1 CTF (competitive hacking) team ↗ worldwide in 2020, 2021, and 2023. Our engineers bring a rich set of skills and backgrounds, including cryptography, web security, mobile security, low-level exploitation, and finance. Our background in traditional information security and competitive hacking has enabled us to consistently discover hidden vulnerabilities and develop novel security research, earning us the reputation as the go-to security firm for teams whose rate of innovation outpaces the existing security landscape.

For more on Zellic's ongoing security research initiatives, check out our website zellic.io ↗ and follow @zellic_io ↗ on Twitter. If you are interested in partnering with Zellic, contact us at hello@zellic.io ↗.

1. Executive Summary

Zellic conducted a security assessment for LayerZero Labs from May 12th to May 30, 2024. During this engagement, Zellic reviewed LayerZero OApp & OFT's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.1. Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Can an OApp be blocked?
- Can an OFT be blocked?
- Is it possible to forge arbitrary messages?
- Is the OApp and OFT validation robust?

1.2. Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Front-end components
- Infrastructure relating to the project
- Key custody

1.3. Results

During our assessment on the scoped LayerZero OApp & OFT contracts, there were no security vulnerabilities discovered.

Additionally, Zellic recorded its notes and observations from the assessment for LayerZero Labs's benefit in the Discussion section (3. ↗) at the end of the document.

2. Introduction

2.1. About LayerZero OApp & OFT

LayerZero Labs contributed the following description of LayerZero OApp & OFT:

> LayerZero is a generic messaging protocol. OFT and OApp are libraries built on top of LayerZero for developers to leverage when building on LayerZero

2.2. Methodology

During a security assessment, Zellic works through standard phases of security auditing, including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application.

Breakdown of Finding Impacts

Impact Level	Count
🟥 Critical	0
🟧 High	0
🟨 Medium	0
🟩 Low	0
◼ Informational	0

We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

2.3. Scope

The engagement involved a review of the following targets:

LayerZero OApp & OFT Contracts

Repository	https://github.com/LayerZero-Labs/monorepo ↗
Version	monorepo: `8e818f95b26ddd16e34c289de15cd44e86198480`
Programs	• packages/layerzero-v2/evm/oapp/contracts/oapp/OApp.sol • packages/layerzero-v2/evm/oapp/contracts/oapp/OAppCore.sol • packages/layerzero-v2/evm/oapp/contracts/oapp/OAppReceiver.sol • packages/layerzero-v2/evm/oapp/contracts/oapp/OAppSender.sol • packages/layerzero-v2/evm/oapp/contracts/oft/OFT.sol • packages/layerzero-v2/evm/oapp/contracts/oapp/OFTAdapter.sol • packages/layerzero-v2/evm/oapp/contracts/oapp/OFTCore.sol • packages/layerzero-v2/evm/oapp/contracts/oapp/OFtPrecrime.sol
Type	Solidity
Platform	EVM-compatible

2.4. Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of two person-weeks. The assessment was conducted over the course of three calendar weeks.

Contact Information

The following project manager was associated with the engagement:

Jasraj Bedi
CTO
jazzy@zellic.io ↗

The following consultants were engaged to conduct the assessment:

Jasraj Bedi
Co-Founder
jazzy@zellic.io ↗

Aaron Esau
Engineer
aaron@zellic.io ↗

2.5. Project Timeline

The key dates of the engagement are detailed below.

May 12, 2024	Start of primary review period
May 30, 2024	End of primary review period

3. Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security-related, and do not convey that we are suggesting a code change.

3.1. Cautions for OApp developers

It is important that OApp developers consider the following:

- **Only override `_lzReceive` — not `lzReceive`.** Note that overriding `_lzReceive` (i.e. not `lzReceive`) would not impact functionality (i.e., it would still pass tests) and would produce no warning but would skip the following critical security checks:
    - `require(address(endpoint) == msg.sender, "OApp: endpoint only");`
    - `assertRemoteAddress(_srcEid, _srcAddress);`
    - `_acceptNonce(_srcEid, _srcAddress, _nonce);`
- **Messages are not ordered by default.** Delivery of messages is ordered, but execution is not.

3.2. Cautions for OFT developers

- **Truncation of dust** The amounts being transferred cross-chain are truncated to the SharedDecimals amount of six. This means there may be dust leftover after transfers that won't be transferrable cross-chain.

4. Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

4.1. Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.