Zellic

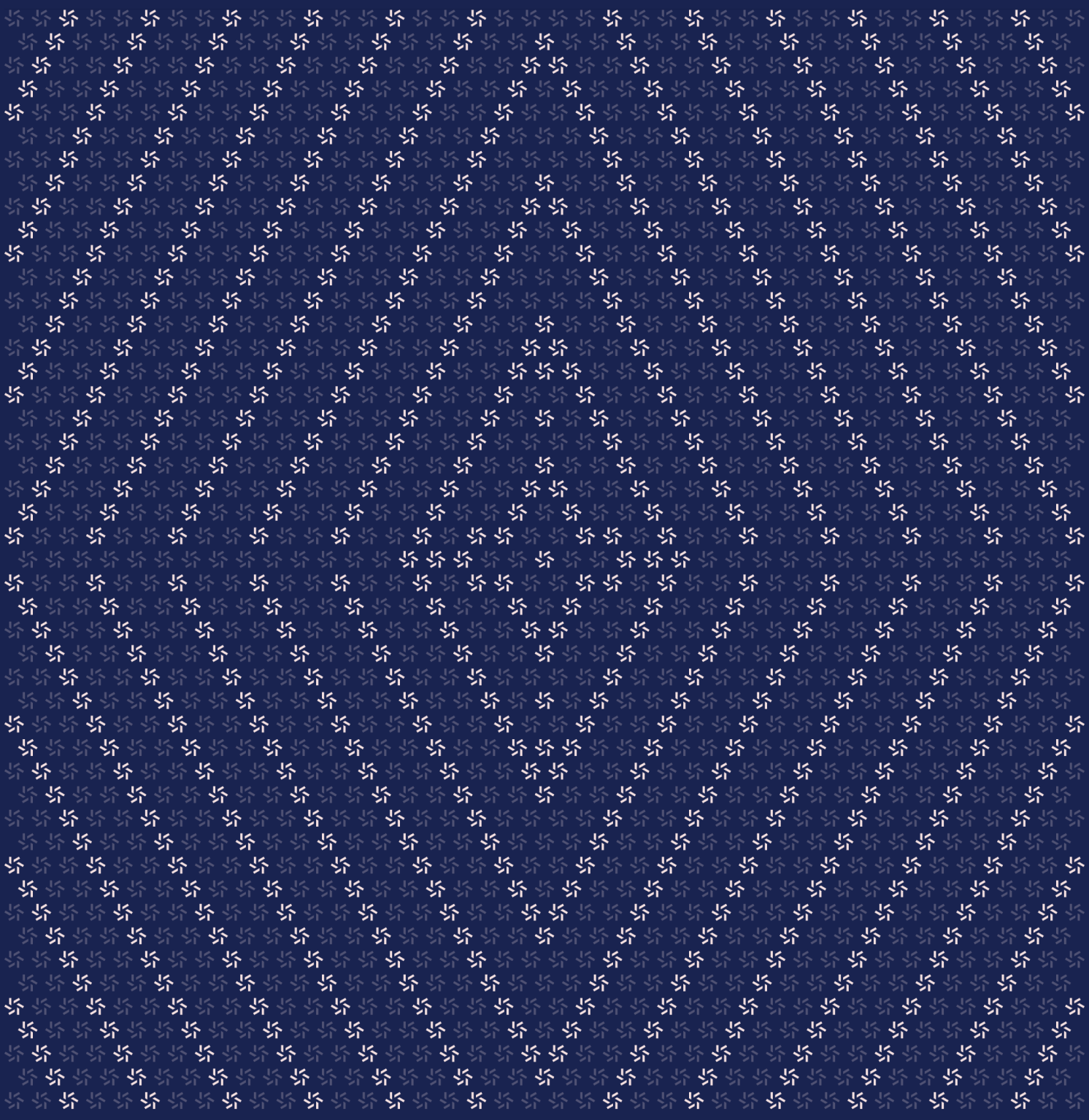
Prepared for
Ryan Zarick
Isaac Zhang
LayerZero Labs

Prepared by
Jasraj Bedi
Aaron Esau
Quentin Lemauf
Zellic

September 23, 2025

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. Overview

1.1. Executive Summary

Zellic conducted a security assessment for LayerZero Labs from May 12th to May 30th, 2024, as well as from September 11th to September 16th, 2025. During this engagement, Zellic reviewed LayerZero OApp & OFT's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.2. 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?

1.3. 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

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide.

1.4. 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 the benefit of LayerZero Labs in the Discussion section (3. ↗).

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	0

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. 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.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped contracts itself. These observations — found in the Discussion (3. ↗) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3. Scope

The engagement involved a review of the following targets:

LayerZero OApp & OFT Contracts

Type	Solidity
Platform	EVM-compatible
Target	devtools
Repository	https://github.com/LayerZero-Labs/devtools ↗
Version	`6c87e0d9e463539ef7215a31838a7007f25fef15`

2.4. Project Overview

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

Additionally, Zellic provided a secondary review of the contracts, covering changes to the codebase since the initial assessment (which was on monorepo ↗ commit `8e818f95`). Details of the latest scope are in the table in section 2.3. ↗.

Contact Information

The following project managers were associated with the engagement:

Jacob Goreski
Engagement Manager
jacob@zellic.io

Chad McDonald
Engagement Manager
chad@zellic.io

Pedro Moura
Engagement Manager
pedro@zellic.io

The following consultants were engaged to conduct the assessment:

Jasraj Bedi
CTO and Co-founder
jazzy@zellic.io

Aaron Esau
Engineer
aaron@zellic.io

Quentin Lemauf
Engineer
quentin@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
September 11, 2025	Start of secondary review period
September 16, 2025	End of secondary 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 developers

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.

OFT developers

It is important that OFT developers consider the following:

- **Truncation of dust.** The amounts being transferred cross-chain are truncated to the SharedDecimals amount of six. This means there may be dust left over after transfers that will not be transferable cross-chain.

4. System Design

This provides a description of the high-level components of the system and how they interact, including details like a function's externally controllable inputs and how an attacker could leverage each input to cause harm or which invariants or constraints of the system are critical and must always be upheld.

Not all components in the audit scope may have been modeled. The absence of a component in this section does not necessarily suggest that it is safe.

4.1. OApp

LayerZero Labs has the ability to change configuration at any time. However, note that the OApps have the ability to create configuration settings that always override the default configuration.

Types of configuration

LayerZero V2 provides the following general types of configuration to OApps:

- **Messaging libraries.** OApps' messaging library choices are restricted to those approved by LayerZero Labs. Note that messaging libraries may have specific configuration options such as the number of inbound confirmations required to deliver a message.
    - *Send library.* This library is responsible for fee calculation and interacting with off-chain entities.
    - *Receive library.* This library is responsible for message validation (i.e., ensuring all proper verifiers approve of the message).
- **Decentralized Verifier Networks.** In the receive library, the verifiers can be configured to some extent depending on the library implementation. Decentralized Verifier Networks (DVNs) are responsible for ensuring packets are valid, but because they are abstracted, the logic or process for determining what constitutes a valid packet is up to the verifier (i.e., it is configurable). In all receive libraries as of the time of this writing, the OApp's delegate may manually configure the DVNs to whatever they desire.
    - *Required DVNs.* All of these types of configured DVNs must approve of the packet, or the packet will be rejected.
    - *Optional DVNs.* A quorum of these types of configured DVNs must approve of the packet, or the packet will be rejected. Note that the quorum number is also configurable by the OApp.
- **Number of confirmations.** The number of block confirmations before a message is considered final.

Recommendations to OApp deployers

Based on these abilities of LayerZero Labs, if an OApp does not desire a specific custom configuration, we recommend fixing the configuration to the current default configuration at the time of deployment.

Note that the purpose of LayerZero Labs's ability to change the default configuration is to make the protocol future-proof without upgradability. As time passes, vulnerabilities or functionality bugs could potentially be discovered in the messaging libraries. Similarly, new technology may be developed to increase the efficiency of verification (i.e., gas savings).

To that end, we recommend that OApp deployers evaluate their present configuration and LayerZero Labs's new default configuration whenever their default configuration changes.

Roles

The **OApp owner** has full control over the OApp configuration because it can set the OApp delegate. Additionally, it can set the allowed peers, meaning it could potentially allow a malicious peer to send arbitrary messages to it. It is essential that the OApp owner is a trusted entity and is protected behind a multi-sig or DAO.

The **OApp delegate** can change the OApp configuration, including the messaging libraries, DVNs, and number of confirmations. These are security-critical settings, so it is essential that the OApp delegate is also trusted and protected.

4.2. OFT

The OFT is ultimately still an OApp, and thus, the OApp's threat model applies to it.

Roles

Like in the OApp, the **OFT owner** is all powerful and is the same as the OApp owner. In addition to the abilities documented above, the OFT owner can change the inspector.

The **message inspector** can block or allow outbound messages based on arbitrary logic, if it is set by the OFT owner. If the inspector is malicious, at worst, it could block all messages from being sent.

Lastly, **users** have the ability to send tokens cross-chain using the `send` function. They are subject to the configured fees and limits, which are customizable for each OFT. They may also execute additional logic on the destination chain using the `composeMsg` parameter. This is standard functionality for many LayerZero OApps.

5. Assessment Results

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

5.1. 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.