Zellic

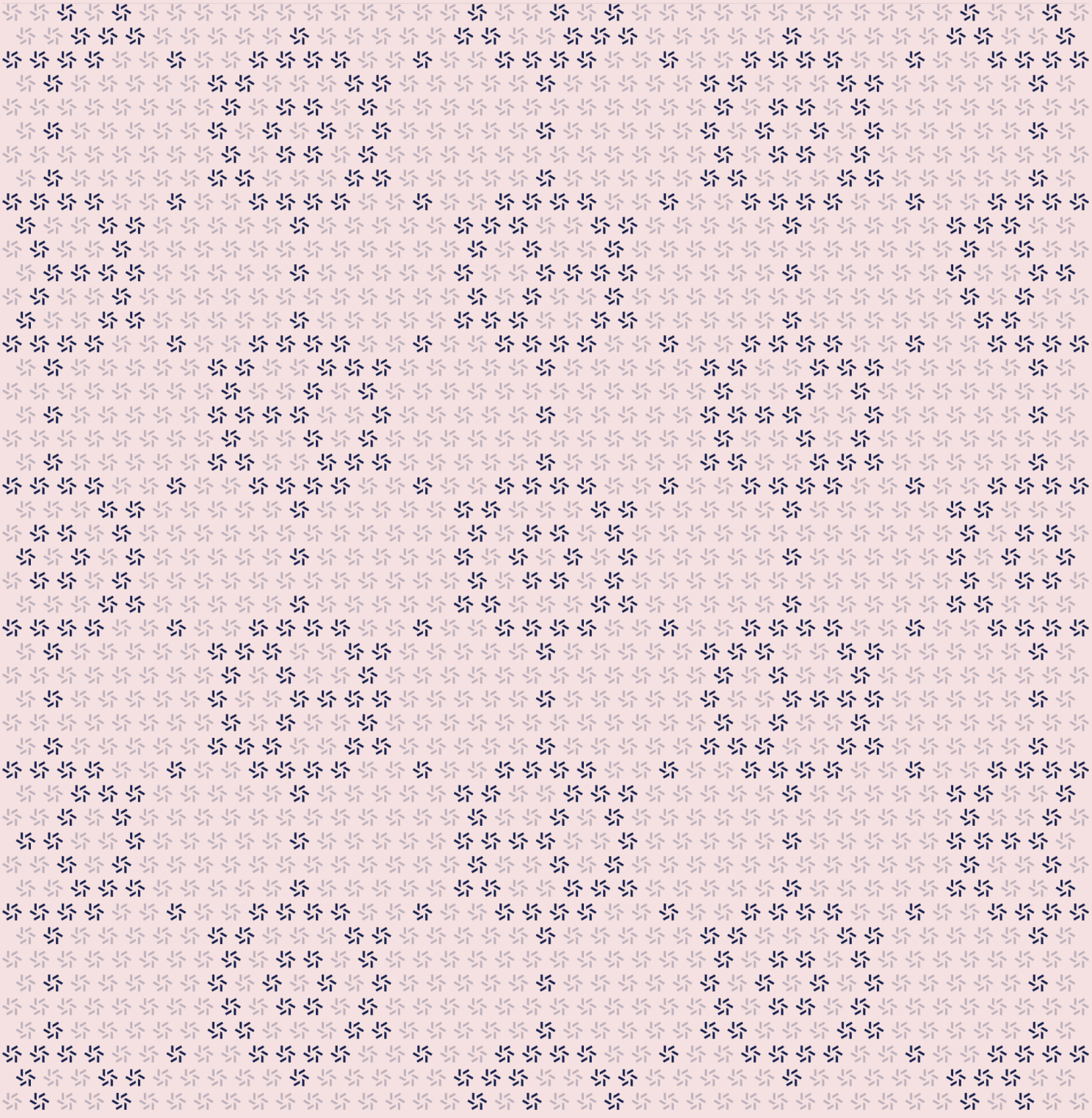
Prepared for
Ryan Zarick
Isaac Zhang
LayerZero Labs

Prepared by
Aaron Esau
Nan Wang
Zellic

January 16, 2025

Ethena OFT

TON Application 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 January 2nd to January 15th, 2025. During this engagement, Zellic reviewed Ethena 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 only designated administrators invoke critical opcodes such as `freeze`, `mint`, `burn`, or `call_to`?
- Are the permission checks for each opcode invocation appropriate, or could there be any unauthorized or out-of-scope operations?
- Does the rate-limiting mechanism adequately protect against excessive net outflows or DOS conditions?
- Could the modifications to the Jetton contract break compatibility with the original Jetton standard?
- Could the modifications to the Jetton contract result in unexpected fund lockups or losses?

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 Ethena OFT contracts, there were no security vulnerabilities discovered.

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 Ethena OFT

LayerZero Labs contributed the following description of Ethena OFT:

> LayerZero is a technology that enables applications to move data across blockchains, uniquely supporting censorship-resistant messages and permissionless development through immutable smart contracts.

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:

Ethena OFT Contracts

Type	FunC
Platform	TON
Target	LayerZero TON Ethena OFT OApp
Repository	https://github.com/LayerZero-Labs/ethena-usde-internal ↗
Version	`9174f785403ddae564bda355755dd7217065f4f3`

Programs	`src/handler.fc`
	`src/main.fc`
	`src/ethenaOFT/*.fc`
	`src/oApp/*.fc`
	`src/token/*`
	`src/tokenAdmin/*`
	`structs/*`

2.4. Project Overview

Zellic was contracted to perform a security assessment for a total of 8 person-days. The assessment was conducted by two consultants over the course of two calendar weeks.

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

The following consultants were engaged to conduct the assessment:

Aaron Esau
Engineer
aaron@zellic.io

Nan Wang
Engineer
nan@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

January 2, 2025	Start of primary review period
January 8, 2025	Kick-off call
January 15, 2025	End of primary review period
January 21, 2025	Scope changed from commit `fa07cd97` to `9174f785`

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. Rate limit increases even after pausing

Note that pausing the protocol does not stop the rate limit from increasing. While this is not a security issue in itself, note that immediately after unpausing, users will be able to send OFTs containing more value than may be expected.

We suggest modifying the `pause` functionality to immediately refill, and then modify `unpause` to reset the `lastRefill` timestamp to the current time.

4. Threat Model

This provides a threat model description of the assessed smart contracts. As time permitted, we analyzed each entity's capability in the contracts and created a written threat model highlighting aspects such as modifications to the standard Jetton contract and the design of tokenAdmin's management capabilities.

Not all functions in the audit scope may have been modeled. The presence of absence of a threat model in this section does not necessarily suggest that a function is safe.

4.1. EthenaJetton

EthenaJetton is a modified version of the standard Jetton contract, designed to support specific features required by the protocol.

Extensions of EthenaJetton to the standard Jetton

EthenaJetton introduces the following key modifications (among others) to the standard Jetton contract, preserving compatibility while enhancing support for cross-chain scenarios — specifically the Wallet → Minter burn flow and the cross-chain contract (EthenaOFT) → Minter mint flow:

1. **Allowing Wallets to directly transfer to the Minter (equivalent to a burn)**

   In a standard Jetton, a user's Wallet typically can only transfer tokens to other Wallets. EthenaJetton, however, permits the Wallet to set `to_owner_address` to the Minter itself, still using the standard `op::internal_transfer` message.

   When the Minter receives this transfer from a Wallet — where the target address is the Minter — it treats the action as a burn by doing `total_supply -= jetton_amount;`. Meanwhile, if `forward_ton_amount > 0`, the original `forward_payload` is preserved and forwarded to the `mint_to_authority`.

   As a result, users can complete a burn simply by having their Wallet send a `transfer` to the Minter.

2. **Introducing the `mint_to_authority` field**

   The `mint_to_authority` is an immutable (unless upgraded) special field, usually pointing to the EthenaOFT contract address. It holds the same minting privileges as `admin_address`: whenever EthenaOFT intends to distribute tokens on TON, it can directly call `op::mint`.

   When the Minter processes a burn-like message from the Wallet (i.e., a transfer with the Minter as the recipient) that includes a `forward_ton_amount`, it constructs an `op::transfer_notification` and forwards it — along with the relevant payload — to `mint_to_authority`, enabling EthenaOFT to receive and handle any additional logic related to the burn event.

3. **Cross-chain integration**

In the Wallet → Minter burn flow, to burn / transfer out tokens on TON, a user can simply send a transfer from the Wallet to the Minter, accomplishing a burn and optionally passing a `forward_payload` on to EthenaOFT.

In the EthenaOFT → Minter mint flow, when tokens arrive from another chain and need to be minted on TON, EthenaOFT (as `mint_to_authority`) can directly call `op::mint` on the Minter, without having to change `admin_address`.

Through these mechanisms, Wallet, Minter, and EthenaOFT coordinate seamlessly to preserve the core Jetton standard interface and simultaneously fulfill cross-chain mint/burn requirements.

4.2. TokenAdmin

TokenAdmin serves as a proxy contract that centralizes the management of the Minter contract's opcodes (such as `mint` and `change_admin`). It assigns a separate admin address to each opcode, and a globally privileged SuperAdmin can at any time replace or revoke any admin. Before any internal message is forwarded to the Minter contract, the TokenAdmin first checks whether the caller has permission for the corresponding opcode and then sanitizes the input, thus enforcing stricter permission control and validation.

The SuperAdmin transition follows a two-step flow (`setTentativeSuperAdmin` and `claimSuperAdmin`). If, in the future, there is a need to completely remove the SuperAdmin role, one can deploy a specialized proxy contract that is only able to execute `claimSuperAdmin` to effectively burn that privilege.

This design enables quick admin rotation to mitigate compromised keys, provides flexibility for upgrades or freezes, and more.

The capabilities of each entity are as follows.

SuperAdmin

`TokenAdmin::OP::SET_TENTATIVE_SUPER_ADMIN`

A SuperAdmin can invoke this opcode to designate a new Tentative SuperAdmin (a two-step flow: first set then let the other party `CLAIM_SUPER_ADMIN`).

`TokenAdmin::OP::CLAIM_TON`

A SuperAdmin can use this opcode to withdraw a specified amount of TON from the contract balance.

`TokenAdmin::OP::SET_ADMIN`

This sets (or replaces) the admin address for a particular opcode. For example, if opcode `op::mint` is assigned to address A, then A can subsequently execute a mint via `CALL_CONTRACT`.

TentativeSuperAdmin

`TokenAdmin::OP::CLAIM_SUPER_ADMIN`

After the current SuperAdmin has called `SET_TENTATIVE_SUPER_ADMIN`, the Tentative SuperAdmin may use this opcode to claim the SuperAdmin role and become the new SuperAdmin.

opcodeAdmin

`TokenAdmin::OP::CALL_CONTRACT`

This opcode allows an admin to initiate a child opcode call to the Minter contract through the TokenAdmin contract. During `CALL_CONTRACT`, TokenAdmin checks permissions to verify whether the current caller is the opcodeAdmin for that opcode. If the check fails, an `OnlyAdmin` error is thrown. If it succeeds, TokenAdmin constructs and sends a request to the Minter contract — first sanitizing the incoming message body to ensure it is a valid command package and then calling `buildMessageForOpcode` to generate the final message body.

The child opcodes may be

- `op::mint`
- `op::change_admin`
- `op::claim_admin`
- `op::upgrade`
- `op::change_metadata_uri`
- `op::transfer`
- `op::burn`
- `op::set_status`
- `op::freeze`

Please refer to the Jetton contract's threat model in section 4.1. ↗ for details regarding the functionality of some of these. The only opcode requiring special mention is `op::freeze`, which is effectively `call_to → set_status 0x03`, allowing a special role to perform blacklisting (freeze) but not to remove it (unfreeze).

5. Assessment Results

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

During our assessment on the scoped Ethena 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.