Zellic

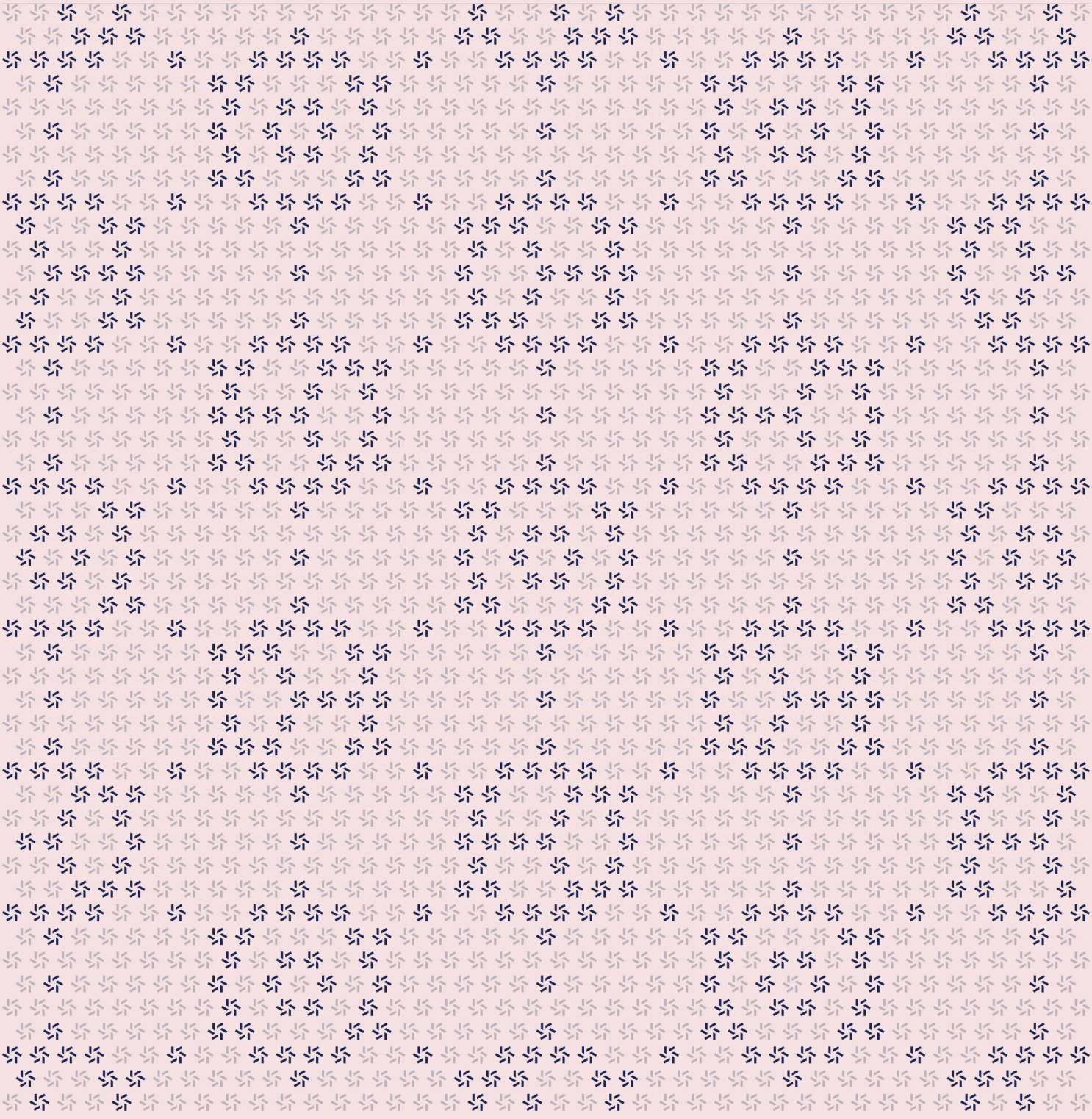
Prepared for
LayerZero Labs

Prepared by
Qingying Jie
Nan Wang
Zellic

May 19, 2025

TON OFT

Ton Application Diff 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 on May 19th, 2025. LayerZero shared a diff of the changes made to ethena-oft to transition it from, an Ethena-specific OFT to a generic implementation. During this engagement, Zellic reviewed TON 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:

- Are the bytecode changes limited only to the storage-object name change from "EthenaOFT" to "OFT"?
- Does the new version maintain functional consistency with the original Ethena OFT?

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 TON OFT files, we discovered one finding, which was informational in nature.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	1

2. Introduction

2.1. About TON OFT

LayerZero Labs contributed the following description of TON 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 files.

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.

2.3. Scope

The engagement involved a review of the following targets:

TON OFT Files

Type	FunC
Platform	TON
Target	Diff of changes made to ethena-oft
Version	`sha256:` `690bd2ab2bfc4b1d2432c0b3d76e11208036d5ab00f23f3ba0a5c8eb4fae14c1`
Programs	`diff_output3.diff`

2.4. Project Overview

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

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:

Qingying Jie
Engineer
qingying@zellic.io

Nan Wang
Engineer
nan@zellic.io

2.5. Project Timeline

The key dates of the engagement are detailed below.

May 19, 2025	Start of primary review period
May 19, 2025	End of primary review period

3. Detailed Findings

3.1. File-path case-sensitivity issue in import statements

Target	xaut-ton-everdawn/xaut-ton/src/TokenAdmin/storage.fc		
Category	Code Maturity	**Severity**	Informational
Likelihood	N/A	**Impact**	Informational

Description

During the transition from EthenaOFT to the generic OFT implementation, the directory-naming convention was changed from lowercase (`token`) to uppercase first letter (`Token`). However, in the file xaut-ton-everdawn/xaut-ton/src/TokenAdmin/storage.fc, an import statement still references the old lowercase path format,

```
#include "../token/op-codes.fc"
```

when it should be this:

```
#include "../Token/op-codes.fc"
```

Impact

This issue causes compilation failures on case-sensitive file systems like Linux, although it works on macOS due to its case-insensitive nature. This inconsistency creates development-environment differences that can be problematic.

If undetected, it could delay deployment and violate the principle of platform-agnostic code design.

Recommendations

We recommend the following.

1. Update the import statement in TokenAdmin/storage.fc to use the correct case-sensitive path:

   ```
   #include "../Token/op-codes.fc"
   ```

2. Implement a cross-platform testing strategy that includes compilation verification on both case-sensitive and case-insensitive file systems.

Remediation

This issue has been acknowledged by LayerZero Labs, and fixes were implemented in the following commits:

- 76533271
- bbf1a008

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. Component: OFT

Description

This component is a standardized implementation of the transition from Ethena OFT to a generic OFT, involving a major restructuring of naming conventions and directory organization. The core change involves replacing the specific name "Ethena" with the more generic "OFT" and reorganizing the entire project structure.

The main changes are as follows:

- **Top-level directory structure**
  - The original `ethena-oft/ethenaoft-ton/...` was migrated to the new namespace `xaut-ton-everdawn/xaut-ton/...`.
  - The new project hierarchy was unified — `src/BamOFT` (contract code), `src/Token`, `src/TokenAdmin`, `src/modules` (common modules), and `src/oftStorages` (unified view for testing/deployment).
- **File/directory renaming**
  - Generic "OFT" naming was introduced, removing the "Ethena" prefix — `ethenaOFT/...` → `modules/OFT/...`.
  - For `token/...` → `Token/...`, the first letter was capitalized for style consistency.
  - For `tokenAdmin/...` → `TokenAdmin/...`, the first letter was capitalized for style consistency.
  - Contract entry point src/main.fc was moved to src/BamOFT/main.fc with path depth corrections.
- **Internal string replacements**
  - All constants, comments, and function names containing "EthenaOFT" were replaced with "OFT".
  - For `Ethena` → `OFT`, events, error codes, and comments were renamed accordingly.
- **Core storage structure**
  - The ethenaoft-ton/src/ethenaOFT/storage.fc was moved to modules/OFT/storage.fc, rewriting the object name as `OFT::NAME` with additional comments.

- All getters/setters now use the `OFT::` prefix.

Invariants

- Code functionality remains unchanged; the new version provides the exact same functionality as the original Ethena OFT.
- All cross-chain logic, security checks, and permission controls remain unchanged.
- Storage-format changes are limited to object-name changes from "EthenaOFT" to "OFT".
- Project directory structure changes do not affect actual deployment and operation.

Attack surface

These changes primarily involve naming conventions and directory-structure reorganization. The attack surface focuses on the following points:

- **Storage-variable renaming**, ensuring that storage key name changes from "EthenaOFT" to "OFT" do not cause existing data loss or conflicts.
- **Path depth changes**, verifying that increasing the depth of include paths (`../` → `../../`//`../../../`) does not introduce incorrect dependencies or version mismatches.
- **Code-migration completeness**, ensuring all files are correctly migrated without omitting any critical components.
- **Naming consistency**, checking that all EthenaOFT references have been replaced with OFT to avoid inconsistencies from partial updates.

5. Assessment Results

During our assessment on the scoped TON OFT files, we discovered one finding, which was informational in nature.

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.