Zellic

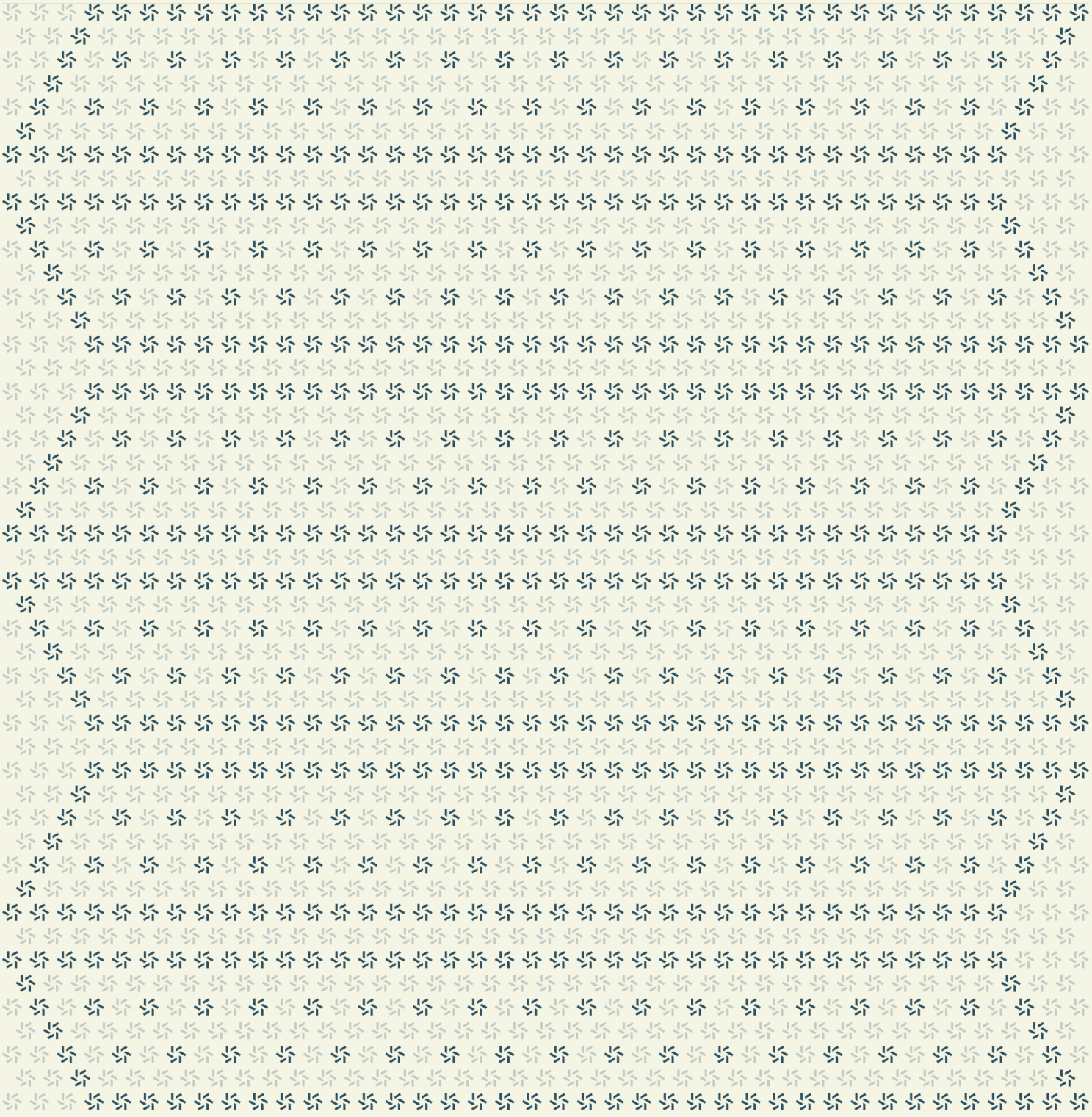
**Prepared for**
Fernando Otero
Anza Technology, Inc.

**Prepared by**
Maik Robert
Avraham Weinstock
Zellic

**October 13, 2025**

# p-token

## Solana Program Patch Review



# 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. Introduction

We were asked to review three different pull requests to p-token, which added a new instruction to p-token as well as improved CU usage by implementing a custom entry point as well as simplifying some math operations.

## 1.1. Scope

The engagement involved a review of the following targets:

### p-token Modules

| | |
|---|---|
| **Type** | Rust |
| **Platform** | Solana |
| **Target** | token |
| **Repository** | https://github.com/solana-program/token/ ↗ |
| **Version** | `ae4630eade05ee3176acb814e72ec1bb2d9a356e` |
| **Programs** | `p-interface/src/instruction.rs`<br>`p-token/src/entrypoint.rs`<br>`p-token/src/processor/batch.rs`<br>`p-token/src/processor/mod.rs`<br>`p-token/src/processor/unwrap_lamports.rs` |

| | |
|---|---|
| **Target** | token |
| **Repository** | https://github.com/solana-program/token/ ↗ |
| **Version** | `2769f3bac969e56e2221dea2371bfec1775397ed` |
| **Programs** | `p-token/src/entrypoint.rs` |

| | |
|---|---|
| **Target** | token |
| **Repository** | https://github.com/solana-program/token/ ↗ |
| **Version** | `9949dee8619b5d51ee3e1c9e23311e1bdd3c2cfd` |
| **Programs** | `p-token/src/processor/close_account.rs`<br>`p-token/src/processor/shared/burn.rs`<br>`p-token/src/processor/withdraw_excess_lamports.rs` |

## 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:

**Maik Robert**
Engineer
maik@zellic.io ↗

**Avraham Weinstock**
Engineer
avi@zellic.io ↗

## 1.2. 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.

## 1.3. Project Timeline

The key dates of the engagement are detailed below.

| | |
|---|---|
| **October 7, 2025** | Start of primary review period |
| **October 9, 2025** | End of primary review period |

# 2. Patch Review

As no issues have been identified, the purpose of this section is to document the changes made by each individual pull request that was part of this patch review.

## 2.1. PR 83: Simplify math operations

This patch replaces several occurrences of `checked_add` / `checked_sub` with unchecked equivalents.

In `process_close_account` and `process_withdraw_excess_lamports`, the destination account balance cannot overflow because the total supply is guaranteed not to exceed $2^{64} - 1$. In `process_burn`, the supply cannot underflow since the supply exceeds the sum of the amounts.

## 2.2. PR 85: Add custom entry point

This patch defines an `entrypoint` function manually instead of using Pinocchio's `program_entrypoint!` macro to wrap `process_instruction`. It includes a fast path that handles the `TransferChecked` and `Transfer` instructions, before falling back to calling `process_instruction` in the same way as the previous `program_entrypoint!`.

On the fast path, it checks first for the number of accounts (which is four for `TransferChecked` and three for `Transfer`) and validates the account layout in memory before checking the instruction discriminators. It then casts the account data to the corresponding types and calls the underlying `process_transfer_checked` or `process_transfer` functions.

## 2.3. PR 87: Add `unwrap_lamports` instruction

This patch adds an `unwrap_lamports` instruction that withdraws lamports from a native token account. It is handled by its discriminator 45 in `inner_process_remaining_instruction`.

If `unwrap_lamports` is used in a `batch` instruction, `process_batch` will ensure that the source account to `unwrap_lamports` is owned by the token program.

In `process_unwrap_lamports`, it checks that the source account is a native account, that the source account signed the transaction, and that the source account has sufficient balance for the transfer. If the destination account differs from the source account, the source account's balance is decremented and the destination account's balance is incremented by the amount of the transfer with unchecked operations that cannot overflow due to the invariant on the total supply.