Zellic

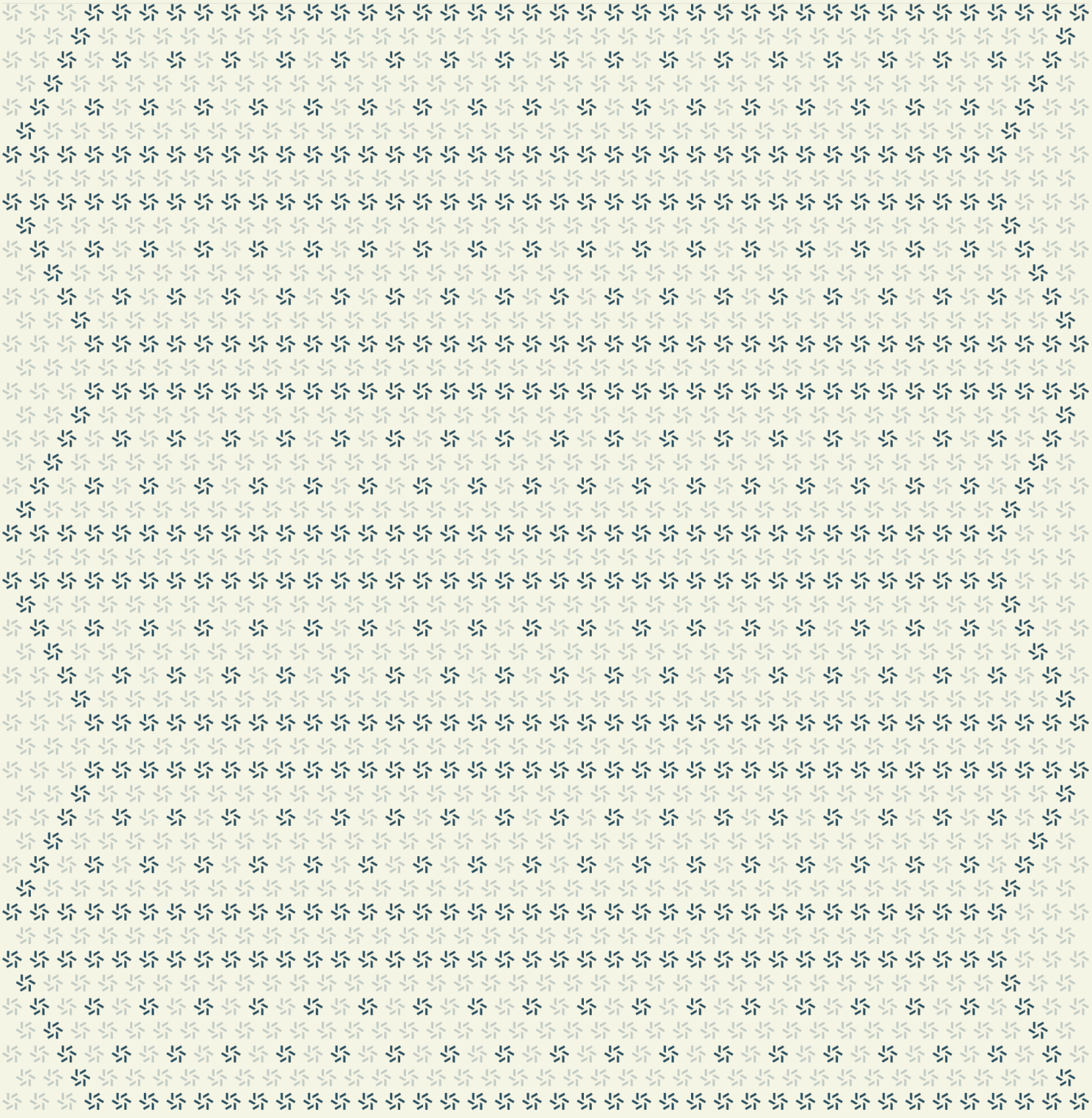
Prepared for
Zach Petersen
Emily Pirnack
Paxos Trust Company, LLC

Prepared by
Filippo Cremonese
Zellic

February 16, 2024

Mint Controller

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

Paxos Trust Company, LLC contributed the following description of the program subject of this review:

> The `minter_controller` program is being created to support Paxos warm minting capabilities. This program adds transaction controls to Solana mint authorities for the token-2022 program. The transaction controls include rate limitings and whitelisting. The program requires the token mint authority to be a spl token multisig.

We were asked to review a minor patch to Mint Controller, intended to allow minting fractional token amounts.

1.1. Scope

The engagement involved a review of the following targets:

Mint Controller Programs

Type	Rust
Platform	Solana
Target	solana-programs
Repository	https://github.com/paxosglobal/solana-programs ↗
Version	`93ceaddea60981ccf5e41ae059acd19f412a9cab`
Programs	`minter-controller/src/instructions/mint_token.rs` `minter-controller/src/lib.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 ↗

The following consultant was engaged to conduct the assessment:

Filippo Cremonese
Engineer
fcremo@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.

2. Patch Review

The objective of this engagement was the review of any changes introduced between commit `b1ba5ccf` and commit `93ceadde` (the most recent commit available at the time of the review). The full list of commits included in this range is as follows:

- `1cc0b0e9`: Add audit reports
  - This commit added PDF audit reports and did not modify any code
- `f3d7abbb`: Update mint token instruction
- `b919d59f`: Remove unused payer from read function
- `93ceadde`: Revert "Remove unused payer from read function"
  - This commit fully reverted the previous one, undoing its changes

2.1. Updates to `MintToken`

Commit `f3d7abbb` updated the `MintToken` instruction to use the spl-token `MintToChecked` instruction instead of the `MintTo` instruction when performing the CPI call to mint tokens. `MintToChecked` differs from `MintTo` in that it receives a `decimal` argument from the caller. The spl-token program checks that the `decimal` argument matches the configuration of the mint. This feature is intended to increase the likelihood of detecting a bug or human error when handling the decimal precision of a token, which could result in minting the wrong order of magnitude of tokens.

As part of the changes introduced to use `MintToChecked`, the `MintToken` instruction handler was modified to remove the scaling of the minted amount.

```
pub fn mint_token(ctx: Context<MintToken>, amount: u64) -> Result<()> {
pub fn mint_token(ctx: Context<MintToken>, amount: u64, decimals: u8) ->
    Result<()> {
     //Checks rate limit
     ctx.accounts.minter.rate_limit.check_limit(amount)?;

     let minter_authority_key = ctx.accounts.minter_authority.key();
     let mint_account_key = ctx.accounts.mint_account.key();
     // PDA signer seeds
     let signer_seeds: &[&[&[u8]]] = &[&[b"minter",
    minter_authority_key.as_ref(), mint_account_key.as_ref(),
    &[ctx.accounts.minter.bump]]];

    // Invoke the mint_to instruction on the token program
    // Invoke the mint_to_checked instruction on the token program
     // Anchor does not implement the token multisig so we have to do it here
    manually.
    let ix = spl_token_2022::instruction::mint_to(
    let ix = spl_token_2022::instruction::mint_to_checked(
         ctx.accounts.token_program.to_account_info().key,
         ctx.accounts.mint_account.to_account_info().key,
```

```
            ctx.accounts.associated_token_account.to_account_info().key,
            ctx.accounts.mint_multisig.to_account_info().key,
            &[ctx.accounts.minter.to_account_info().key],
        amount * 10u64.pow(ctx.accounts.mint_account.decimals as u32),
        amount,
        decimals
    )?;
```

This change is significant, because the both the checked and unchecked variants of the `MintTo` instruction mint the exact amount of non-normalized units of tokens received as input. The `decimal` argument is not involved in determining the amount of minted tokens. The caller must therefore supply the correctly scaled amount.

Paxos Trust Company, LLC confirmed this change is intentional and intended to allow minting fractional amounts of the minted token.