SECURITY ASSESSMENT

Accretion

Provided by Accretion Labs Pte Ltd. for Solana Foundation
January 19, 2026

Solana Foundation

A25SFR2

AUDITORS

Role	Name
Lead Auditor	Robert Reith (robert@accretion.xyz)
Auditor	Mahdi Rostami (mahdi@accretion.xyz)

CLIENT

Solana Foundation (https://solana.org) engaged Accretion to conduct a security assessment of Token ACL, a permissioned token standard that enables allow/block listing.

ENGAGEMENT TIMELINE

09 Oct — Project Kickoff
Initial planning and scope definition

09 Oct — Assessment Begins
Security review and testing phase

17 Nov — Review Fixes
Security recommendations are given and implemented

19 Jan — Project Completion
Report delivery and on-chain confirmation

AUDITED CODE

Program 1

ProgramID: TACLkU6CiCdkQN2MjoyDkVg2yAH9zkxiHDsiztQ52TP

Repository: https://github.com/solana-foundation/token-acl

ASSESSMENT

The security assessment of the Token ACL Solana program revealed a focused set of issues in a compact codebase. We identified 5 security findings: 1 high, 2 medium, and 2 low severity. All findings were promptly fixed by the team, resulting in a clean final state with no outstanding issues.

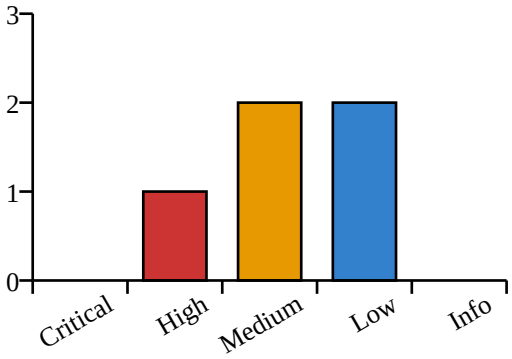
CODE ASSESSMENT

The code implemented a token access control layer for managing mint configurations and permissions. The program was relatively small in scope but had critical gaps in ownership validation. The team was highly responsive, addressing all identified issues including the high-severity ownership check bypass.

KEY FINDINGS

Our top findings included a high-severity issue where program ownership of the Mint Config was not validated, allowing potential spoofing. Medium-severity issues covered denial of service through account balance manipulation and unauthorized mint configuration creation. Low-severity findings addressed an incorrect idempotent check causing early returns and inability for mint config authorities to reclaim rent when using the close mint extension.

SEVERITY DISTRIBUTION



ENGAGEMENT SCOPE

The scope of this security assessment was a full review of the following items:

Item 1: Token ACL

Link: https://github.com/solana-foundation/token-acl
Commit: b14f6b128af3f38aefcc552929eec1ae789d9bcf
Program ID: TACLkU6CiCdkQN2MjoyDkVg2yAH9zkxiHDsiztQ52TP
Audit Result:

- **Audited Commit:** 7f4238515c82aa8b0b43e590eb95d621ba406fd5
- **Build Hash:** unverified
- **Status:** unverified
- **Comment:** not verified

ISSUES SUMMARY

ID	TITLE	SEVERITY	STATUS
ACC-H1	Not checking program ownership of Mint Config	HIGH	FIXED
ACC-M1	Unauthorized Mint Configuration Creation	MEDIUM	FIXED
ACC-M2	Denial of Service via Account Balance Manipulation	MEDIUM	FIXED
ACC-L1	Mint Config Authority Unable to Close Config and Reclaim Rent When Close Mint Extension Is Used	LOW	FIXED
ACC-L2	Incorrect Idempotent Check Causes Early Return	LOW	FIXED

DETAILED ISSUES

ACC-H1 Not checking program ownership of Mint Config

HIGH FIXED

Description

We found that whenever the Mint Config is used, its program ownership or address isn't explicitly checked. The program primarily relies on implicit checks, such as writing data to the account, or using it as a signing authority. However in some cases, such as when creating a new config, this isn't enough.

Location

https://github.com/solana-foundation/token-acl/blob/141865422f4398df9a50397e943ce18ed93cfebc/program/src/instructions/create_config.rs#L122-L123

Relevant Code

```
/// create_config.rs L122-L123
        let (_, config_bump) =
            Pubkey::find_program_address(&[MintConfig::SEED_PREFIX, mint.key.as_ref()], &crate::ID);
```

Mitigation Suggestion

In every instruction that uses the Mint Config account, confirm the address and program ownership status before using the account.

Remediation

Fixed in commits `cf2d2653c97f10f933d8baf1e12682dab2a75f20`, and `5d79e669e3af9301d1f0b9d10597ebd8ec6c9d7a`.

ACC-M1 Unauthorized Mint Configuration Creation

MEDIUM FIXED

Description

We found that in `create_config`, anyone could create a configuration for any mint and set themselves as the authority. In addition, if the program does not change the freeze authority in `create_config`, there is no way to properly use this standard and the program.

Location

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/create_config.rs#L82-L93

Relevant Code

```
if freeze_authority == *self.authority.key {
            // we can cpi to change freeze authority right away
            let ix = spl_token_2022::instruction::set_authority(
                self.token_program.key,
                self.mint.key,
                Some(self.mint_config.key),
                AuthorityType::FreezeAccount,
                self.authority.key,
                &[],
            )?;
            invoke(&ix, &[self.mint.clone(), self.authority.clone()])?;
        }
```

Mitigation Suggestion

Only allow the freeze authority to create a mint configuration.

Remediation

Fixed in `https://github.com/solana-foundation/token-acl/pull/7`.

ACC-M2 Denial of Service via Account Balance Manipulation

MEDIUM FIXED

Description

We found that an attacker could send some lamports to certain accounts (such as `flag_account` or `mint_config`) and cause a denial of service (DoS) in processes like Permission-less Thaw and Freeze or `create_config`. This happens because `create_account` reverts if the target account already has a positive balance. You can see this behavior in the system processor implementation [here](https://github.com/anza-xyz/agave/blob/5bfe52efb95017b2cbdf00dde1445a6c23d0161d/programs/system/src/system_processor.rs#L160-L161).

Location

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/create_config.rs#L41-L47

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/freeze_permissionless.rs#L66-L72

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/thaw_permissionless.rs#L66-L72

Relevant Code

```
let ix = solana_system_interface::instruction::create_account(
            self.payer.key,
            self.mint_config.key,
            lamports,
            MintConfig::LEN as u64,
            &crate::ID,
        );

        invoke_signed(
            &ix,
            &[self.payer.clone(), self.mint_config.clone()],
            &[&seeds],
        )?;
```

```
        let ix = solana_system_interface::instruction::create_account(
            self.authority.key,
            self.flag_account.key,
            0,
            1 as u64,
            &crate::ID,
        );

        invoke_signed(
            &ix,
            &[self.authority.clone(), self.flag_account.clone()],
            &[&seeds],
        )?;
```

Mitigation Suggestion

Instead of using `create_account`, calculate the required lamports, transfer them to the target account (such as `flag_account` or `mint_config`), and then perform `allocate` and `assign` operations manually.

Remediation

Fixed in `https://github.com/solana-foundation/token-acl/pull/8` and `https://github.com/solana-foundation/token-acl/pull/9`.

ACC-L1 Mint Config Authority Unable to Close Config and Reclaim Rent When Close Mint Extension Is Used

LOW FIXED

Description

We found that when a user tries to call `delete_config` to reclaim rent, the function sets the freeze authority to a new authority.

The issue is that it does not check whether the mint account still exists or if it was closed and recreated. In such cases, the user may be unable to close the mint config and reclaim rent, and the new authority may not be able to create a mint config afterward.

Location

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/delete_config.rs#L46-L58

Relevant Code

```
let ix = spl_token_2022::instruction::set_authority(
            self.token_program.key,
            self.mint.key,
            Some(&new_freeze_authority),
            AuthorityType::FreezeAccount,
            self.mint_config.key,
            &[],
        )?;
        invoke_signed(
            &ix,
            &[self.mint.clone(), self.mint_config.clone()],
            &[&seeds],
        )?;
```

Mitigation Suggestion

Before calling the token program, check if the mint account exists and ensure it has not been closed or had its freeze authority changed. If it has, skip calling the token program.

Remediation

Fixed in `https://github.com/solana-foundation/token-acl/pull/6`.

ACC-L2 Incorrect Idempotent Check Causes Early Return

LOW FIXED

Description

We found that the current idempotent feature checks the account state too early and then returns. Idempotent behavior should skip only the final state change when the state is already the desired one. It should not skip earlier validation or access-control steps.

It prevents other programs from relying on our validations, which can cause downstream issues.

Location

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/freeze_permissionless.rs#L42-L46

https://github.com/solana-foundation/token-acl/blob/b14f6b128af3f38aefcc552929eec1ae789d9bcf/program/src/instructions/thaw_permissionless.rs#L42-L46

Relevant Code

```
if is_idempotent {
            let ta_data = self.token_account.data.borrow();
            let ta = StateWithExtensions::<spl_token_2022::state::Account>::unpack(&ta_data)?;
            if ta.base.state != AccountState::Frozen {
                return Ok(());
```

Mitigation Suggestion

Move the idempotent check to the last step, right before calling the token program.

Remediation

Fixed in `https://github.com/solana-foundation/token-acl/pull/10/commits/8578a08ff09609115ecc7863d8e0190b57934014`.

APPENDIX

Vulnerability Classification

We rate our issues according to the following scale. Informational issues are reported informally to the developers and are not included in this report.

Severity	Description
Critical	Vulnerabilities that can be easily exploited and result in loss of user funds, or directly violate the protocol's integrity. Immediate action is required.
High	Vulnerabilities that can lead to loss of user funds under non-trivial preconditions, loss of fees, or permanent denial of service that requires a program upgrade. These issues require attention and should be resolved in the short term.
Medium	Vulnerabilities that may be more difficult to exploit but could still lead to some compromise of the system's functionality. For example, partial denial of service attacks, or such attacks that do not require a program upgrade to resolve, but may require manual intervention. These issues should be addressed as part of the normal development cycle.
Low	Vulnerabilities that have a minimal impact on the system's operations and can be fixed over time. These issues may include inconsistencies in state, or require such high capital investments that they are not exploitable profitably.
Informational	Findings that do not pose an immediate risk but could affect the system's efficiency, maintainability, or best practices.

Audit Methodology

Accretion is a boutique security auditor specializing in Solana's ecosystem. We employ a customized approach for each client, strategically allocating our resources to maximize code review effectiveness. Our auditors dedicate substantial time to developing a comprehensive understanding of each program under review, examining design decisions, expected and edge-case behaviors, invariants, optimizations, and data structures, while meticulously verifying mathematical correctness—all within the context of the developers' intentions.

Our audit scope extends beyond on-chain components to include associated infrastructure, such as user interfaces and supporting systems. Every audit encompasses both a holistic protocol design review and detailed line-by-line code analysis.

During our assessment, we focus on identifying:
• Solana-specific vulnerabilities
• Access control issues
• Arithmetic errors and precision loss
• Race conditions and MEV opportunities
• Logic errors and edge cases
• Performance optimization opportunities
• Invariant violations
• Account confusion vulnerabilities
• Authority check omissions
• Token22 implementation risks and SPL-related pitfalls
• Deviations from best practices

Our approach transcends conventional vulnerability classifications. We continuously conduct ecosystem-wide security research to identify and mitigate emerging threat vectors, ensuring our audits remain at the forefront of Solana security practices.