Security Review

Cantina Managed

SOLANA FOUNDATION: TOKEN-ACL-GATE

Cantina Managed review by:
R0bert, Lead Security Researcher
Sujith Somraaj, Lead Security Researcher

CANTINA

August 14, 2026

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Medium Risk . . . 4
  - 3.1.1 Mutable-owner token accounts cannot be permissionlessly frozen . . . 4
- 3.2 Low Risk . . . 4
  - 3.2.1 Stale gate policies survive a Token ACL config reset . . . 4
  - 3.2.2 TypeScript freeze setup silently installs an empty policy . . . 5
  - 3.2.3 Raw IDL silently generates a fail-open thaw setup . . . 6
  - 3.2.4 Deleting a list invalidates active mint configurations . . . 6
  - 3.2.5 CLI create-list and add-wallet always panic before RPC . . . 7
  - 3.2.6 Build succeeds with a stale or missing IDL . . . 7
  - 3.2.7 Stale MintConfigs overwrite gate policy . . . 8
  - 3.2.8 Initialized default bypasses Allow policies . . . 9
  - 3.2.9 Owner rotation lets a blocklisted account evade freezing . . . 11
- 3.3 Informational . . . 11
  - 3.3.1 Upgrade invalidates existing mode-1 lists . . . 11
  - 3.3.2 Legacy count drift requires migration census only . . . 12
  - 3.3.3 Immutable-owner parser trusts malformed TLV headers . . . 13
  - 3.3.4 Generated gate client forwards unnecessary signer privilege . . . 13
  - 3.3.5 Gate errors collide with Token ACL errors and are absent from the generated clients 14
  - 3.3.6 Gate dispatch accepts non-standard discriminator aliases . . . 15
  - 3.3.7 Checkout path newlines inject Cargo directives . . . 15
  - 3.3.8 Gate validation is not bound to the supplied owner . . . 16
  - 3.3.9 Unsafe state casts lack layout guarantees . . . 17
  - 3.3.10 Mutable account-data reference is retained across CPIs . . . 17
  - 3.3.11 `README` documents obsolete modes and incorrect instruction discriminators . . . 17
  - 3.3.12 Tests may execute a stale checked-in SBF binary . . . 17
  - 3.3.13 Existing mint adoption needs account migration . . . 18

1 Introduction

1.1 About Cantina

Cantina is a security platform that pairs a world-class network of security researchers with agentic AI to help teams building onchain find and fix real risk. Through managed reviews, competitions, and bounties, Cantina secures protocols before vulnerabilities can be exploited.
Learn more at cantina.security

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

The Solana Foundation is a non-profit foundation based in Zug, Switzerland, dedicated to the decentralization, adoption, and security of the Solana ecosystem.

From Jul 21st to Jul 23rd the Cantina team conducted a review of token-acl-gate on commit hash 2b99db71. The team identified a total of **23** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	1	1	0
Low Risk	9	6	3
Gas Optimizations	0	0	0
Informational	13	10	3
Total	**23**	**17**	**6**

2.1 Scope

The security review had the following components in scope for token-acl-gate on commit hash 2b99db71:

```
program
├── build.rs
└── src
    ├── error.rs
    ├── lib.rs
    ├── instructions
    │   ├── add_wallet.rs
    │   ├── can_freeze_permissionless.rs
    │   ├── can_thaw_permissionless.rs
    │   ├── create_list.rs
    │   ├── delete_list.rs
    │   ├── mod.rs
    │   ├── remove_wallet.rs
    │   ├── setup_extra_metas.rs
    │   └── setup_freeze_extra_metas.rs
    └── state
        ├── list_config.rs
        ├── mod.rs
        └── wallet_entry.rs
```

3 Findings

3.1 Medium Risk

3.1.1 Mutable-owner token accounts cannot be permissionlessly frozen

Severity: Medium Risk.

Context: can_freeze_permissionless.rs#L45-L47.

Description: `CanFreezePermissionless::process` rejects every token account that lacks the optional Token-2022 `ImmutableOwner` extension before it evaluates any configured list. This prerequisite is necessary for permissionless thaw because an owner that can change after authorization could escape the policy attached to the old owner. It is not necessary for freeze. Token ACL 0.2.3 unpacks the token account and binds the supplied owner to `token_account.base.owner`, then invokes the gate and the Token-2022 freeze in one atomic instruction while holding the token account writable. Token-2022 also refuses account-owner changes after an account is frozen.

A 165-byte Token-2022 account without extensions is nevertheless a valid token account. The strongest reproduced path does not depend on existing-mint migration: under a mint whose current default is `Frozen`, a base-layout account can be created frozen, thawed by the issuer through Token ACL's permissioned path and later become subject to permissionless block-list enforcement. Token ACL's config creation does not prove that all token accounts that may later need freezing carry `ImmutableOwner`.

Consequently, a blocked holder with an initialized mutable-owner account cannot be frozen through the permissionless enforcement path. The gate returns `ImmutableOwnerExtensionMissing`, Token ACL rolls the transaction back before its freeze CPI and the account remains able to transfer its balance. The proof of concept uses one block-list entry for two accounts owned by the same wallet: the owner's associated token account with `ImmutableOwner` freezes successfully, while its valid base-layout account rejects the freeze and transfers its token after the failed enforcement attempt.

The issuer can still use Token ACL's permissioned freeze as a recovery path, so the bypass is not necessarily permanent. It nevertheless defeats the permissionless block-list guarantee until the issuer notices and intervenes.

Recommendation: Do not require `ImmutableOwner` in `can_freeze_permissionless`; retain the check in `can_thaw_permissionless`. The current Token ACL owner comparison and immediate Token-2022 freeze CPI provide the binding needed during the atomic freeze operation and a successfully frozen account cannot change its owner. If this gate must independently authenticate the Token ACL context in the future, validate the caller or transient flag and parse the token account rather than using `ImmutableOwner` as a freeze prerequisite.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix. The immutable-owner prerequisite was removed from the freeze handler.

3.2 Low Risk

3.2.1 Stale gate policies survive a Token ACL config reset

Severity: Low Risk.

Context: setup_extra_metas.rs#L96-L100.

Description: The thaw and freeze extra-metas PDAs are derived only from a variant seed and the mint address. Their TLV data stores list-resolution rules, but no Token ACL `MintConfig`, freeze authority or configuration generation. The gate also has no instruction that invalidates these accounts when Token ACL deletes the corresponding mint config.

Token ACL can delete its canonical `MintConfig`, return the Token-2022 freeze authority to another key and later recreate a fresh config at the same address. If the successor selects this gate and enables a permissionless operation before replacing its metadata, Token ACL silently resolves the prior

configuration's policy. An independent list operator trusted only by the old authority therefore retains policy influence after the reset.

This affects both controls. In the thaw proof of concept, the successor recreated the config, enabled permissionless thaw without calling `SetupExtraMetas`, funded an account and deliberately froze it. The old Allow-list operator then added the holder, permissionlessly thawed the account through the inherited metadata and the holder transferred its token. In the freeze proof of concept, an old Block-list operator added a new holder and caused its issuer-thawed account to freeze. The former mint authority cannot actively rewrite the successor's metadata; the capability survives through the unchanged PDA and the still-live delegated list.

The successor must choose the same gate, enable the affected permissionless operation and omit a fresh metadata rewrite. The current authority can recover by replacing the inherited state. These conditions reduce likelihood, but an inherited thaw policy can release deliberately frozen assets and restore transferability. This is an access-control bypass under a supported configuration-reset sequence.

POC file: 8d0da5285aaaeb759a9b3d2c8aa5a5ff.

Recommendation: Bind both metadata variants to a non-reusable Token ACL configuration generation, then require permissionless evaluation to prove that the presented generation is current. This requires a coordinated Token ACL/interface change because the current callback does not carry a generation identifier.

Alternatively, make Token ACL's config deletion invalidate both metadata PDAs. Do not allow permissionless thaw or freeze on a recreated config until its current authority installs fresh metadata. Until the lifecycle is enforced on chain, deployment tooling should overwrite both metadata variants after every config deletion or authority handoff and before enabling either permissionless operation.

Solana Foundation: TokenACL implies a certain trust between MintConfig authority and the ListConfig authority. On a worst case scenario, a MintConfig authority can overwrite the configured stale thaw/freeze lists that were previously setup immediately after a MintConfig reset.

Cantina Managed: Acknowledged.

3.2.2 TypeScript freeze setup silently installs an empty policy

Severity: Low Risk.

Context: codama.ts#L25-L62.

Description: The SDK generator gives `setupExtraMetas` an `addresses` argument whose values become remaining accounts. It does not configure the same argument for `setupFreezeExtraMetas`. Consequently, the generated TypeScript `getSetupFreezeExtraMetasInstruction` constructor accepts no list input and emits only its six fixed accounts.

Both setup opcodes call `SetupExtraMetas::process`, which treats every account after the fixed six as a policy list. A freeze setup built by the TypeScript constructor therefore succeeds and writes a valid freeze metadata account containing zero entries. `CanFreezePermissionless::process` returns `AccountAllowed` when no configured list approves a freeze, so no wallet can be permissionlessly frozen under this configuration.

An operator using the published TypeScript constructor can silently install an empty freeze policy instead of the intended allow or block lists. For example, a wallet present in the intended block list remains thawed until the operator reconfigures the mint with the Rust SDK, the CLI or a manually extended TypeScript instruction. No attacker action is required. The persistent security-control misconfiguration makes this more than a transient client error.

Recommendation: Add a `setupFreezeExtraMetas` entry to the Codama instruction visitor. It should mirror the `addresses` argument and read-only remaining-account mapping of `setupExtraMetas`. Regenerate the TypeScript client and add a parity test that requires the freeze setup constructor to append every supplied policy list.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix. The generated TypeScript freeze constructor now requires and appends addresses as remaining accounts in `setupFreezeExtraMetas.ts`.

3.2.3 Raw IDL silently generates a fail-open thaw setup

Severity: Low Risk.

Context: build.rs#L17-L28, mod.rs#L49-L100.

Description: The four instructions in `mod.rs` that consume dynamic policy accounts describe those accounts only in Rust comments. `build.rs` writes Codama's unprocessed output directly to `idl/token_acl_gate_program.json`, where `setupExtraMetas`, `setupFreezeExtraMetas`, `canThawPermissionless` and `canFreezePermissionless` all lack `remainingAccounts`. A standard Codama TypeScript client generated directly from that committed IDL consequently exposes no policy-account argument and emits only the six fixed accounts for every one of those instructions.

The repository's own `codama.ts` process hides part of the defect by adding dynamic accounts to the thaw instructions in memory before rendering the checked-in clients. It does not update the committed IDL, so downstream users that treat that IDL as the program's portable interface still generate the incomplete constructors. This is distinct from the shipped TypeScript freeze-client omission: even `setupExtraMetas`, which works in the shipped TypeScript SDK, is incomplete when regenerated from the raw IDL.

The thaw setup failure is fail-open. `SetupExtraMetas::process` accepts zero remaining accounts, successfully initializes a valid zero-item metadata account and returns success. Token ACL then resolves no list/entry pairs, so `CanThawPermissionless::process` reaches its unconditional success result. An operator who intended to install an allow list through a raw-IDL-generated client instead installs an empty policy under which every immutable-owner token account can be permissionlessly thawed.

The official checked-in TypeScript client patches thaw setup and the Rust builder lets callers append arbitrary remaining accounts, so affected operators must generate or validate a client directly from the committed IDL. This dependency on a non-canonical client-generation path limits the finding to low severity despite the fail-open policy result.

Recommendation: Make the committed IDL the complete, canonical interface. Encode machine-readable remaining-account definitions for all four dynamic instructions or serialize the post-visitor Codama tree back to `idl/token_acl_gate_program.json` before any client is rendered. Do not rely on repository-local, in-memory visitors to repair a portable ABI artifact.

Consider also adding a CI test that generates a fresh client from the committed IDL and requires callers to be able to append up to five list accounts to both setup instructions and flattened list/entry pairs to both gate instructions. Add an end-to-end regression test in which an intended allow list rejects an unlisted wallet; if a zero-list thaw policy is meant to be supported, require operators to select it explicitly rather than obtaining it from an omitted client field.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix. The committed IDL now contains remaining-account definitions for all four dynamic instructions.

3.2.4 Deleting a list invalidates active mint configurations

Severity: Low Risk.

Context: delete_list.rs#L45-L56.

Description: `DeleteList::process` treats a zero `wallets_count` as the only lifecycle condition before closing `list_config`. It does not account for thaw or freeze extra-metas accounts that still contain the list public key. Those accounts continue to resolve the closed address, but both gate validators require that address to remain owned by this program and decode as `ListConfig`. Consequently, every permissionless thaw that includes the stale list fails with `InvalidListConfig`.

Permissionless freeze also fails whenever evaluation reaches the stale list; a preceding list that already authorizes freeze can short-circuit first, making otherwise equivalent multi-list configurations order-sensitive. A single-list configuration always fails until the mint authority replaces the metadata or the list authority recreates the list.

Recreation does not fully restore the old policy. `create_list` can initialize the same PDA with a different mode because the address commits to the authority and seed, not the mode. The list authority can even place `delete_list` and `create_list` in one transaction, atomically replacing the mode without a missing-account interval. Existing mint metadata then starts using the replacement mode without approval from the mint authority. A third-party list operator can therefore interrupt subscribed mints or replace the interpretation of an empty list.

Recommendation: Prevent list closure while subscriptions exist. Track each `(list, mint, thaw-or-freeze)` subscription when extra metas are configured and require the subscription count to be zero in `delete_list`. Because any mint authority can currently reference a public list, registration must require list-authority approval or use expiring leases; otherwise an arbitrary mint could subscribe solely to prevent deletion. Add a retirement step so mint authorities can migrate before rent is reclaimed. Bind each subscription to an immutable list generation and expected mode so recreating the same PDA cannot silently replace the policy. If a safe bilateral lifecycle is not added, remove `delete_list` and leave published list accounts initialized.

Solana Foundation: The protocol assumes a certain level of trust between issuers and list controllers. Even if this trust is broken, the issuer can immediately change the configuration to remove the dependency on the deleted list.

Cantina Managed: Acknowledged.

3.2.5 CLI create-list and add-wallet always panic before RPC

Severity: Low Risk.

Context: main.rs#L128-L133.

Description: `process_create_list` sets the authority, seed, mode and list account on `CreateListBuilder`, but it never sets the mandatory payer account. `process_add_wallet` makes the same omission at `cli/src/main.rs:192-199`. Both generated builders finalize the instruction with `self.payer.expect("payer is not set")`. Therefore, `create-list` and `add-wallet` panic before the CLI requests a blockhash or submits a transaction.

An operator cannot create a policy list or add a wallet through the official CLI. The on-chain instructions and generated SDK remain usable, so an operator can switch clients or patch the CLI. No attacker action is required. This is a low-severity integration failure rather than an on-chain vulnerability.

Recommendation: Set `.payer(payer.pubkey())` in both functions before calling `.instruction()`. Keep a binary regression test for `create-list` and `add-wallet` that uses a dead localhost RPC endpoint. After the fix, both commands should reach a clean RPC error instead of panicking. Consider changing generated builders to return a descriptive `Result` for missing fields instead of panicking in production callers.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.2.6 Build succeeds with a stale or missing IDL

Severity: Low Risk.

Context: build.rs#L7-L11.

Description: `main` limits Cargo's change detection to `src/`, then converts every error returned by `generate_idl` into a warning. The generated file is written to the repository-level `idl/` directory

instead of Cargo's tracked `OUT_DIR`. Consequently, a successful Cargo invocation does not prove that `idl/token_acl_gate_program.json` exists or matches the program that was compiled.

Two independent cases reproduce this condition. After one successful build, deleting the IDL and rerunning the documented `cargo check --features idl` command does not execute the cached build script, so Cargo exits successfully and leaves the IDL missing. Separately, changing the compiled program ID while adding a Codama directive that Codama cannot resolve makes IDL generation fail. Cargo still exits with status zero, while the old IDL retains the previous program ID.

`codama.ts` treats this file as the source for the committed Rust and TypeScript clients. The release workflows build and publish those clients without failing on IDL warnings or checking that regenerated artifacts have no diff. A release can therefore pair a new on-chain program with clients that use an old program ID, discriminator, account order or data layout. The most likely result is failed or misdirected transactions. A reused discriminator whose behavior changed could instead make an outdated client request an unintended operation. The on-chain binary itself does not consume the IDL.

The direct `fs::write` also truncates the destination before the write completes. An I/O failure or concurrent reader can therefore leave malformed JSON and the same warning-only handling still reports a successful build.

Recommendation: Move checked-in IDL generation out of `build.rs` and into a dedicated command that returns a non-zero exit status on any parsing, serialization or write error. Generate into a temporary file, validate it and atomically rename it into `idl/` only after success. Add a CI check that regenerates the IDL and both SDKs, then fails if `git diff --exit-code` reports any change.

If generation must remain in `build.rs`, propagate the error from `main` instead of printing only a warning. Do not rely on Cargo's build-script cache as an artifact freshness check; add a separate test that generates into a temporary directory and compares the result byte-for-byte with the committed IDL.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.2.7 Stale MintConfigs overwrite gate policy

Severity: Low Risk.

Context: setup_extra_metas.rs#L96-L115.

Description: `SetupExtraMetas::process` derives the canonical thaw or freeze metadata PDA from the mint, then authorizes the write with the supplied Token ACL `MintConfig`. It checks the record's stored mint, authority and gating program. It never verifies that the record itself is the canonical Token ACL PDA for that mint or that the mint still names it as freeze authority. `SetupExtraMetas::try_from` checks only that Token ACL owns the record. Consequently, a genuine stale same-mint record gives its old stored authority write access to the current mint's canonical gate policy.

Older untagged Token ACL code under the current program ID could create this state. One historical commit allowed the real mint freeze authority to create a second, noncanonical `MintConfig` for the same mint. That noncanonical record could then hand the mint's Token-2022 freeze authority to the canonical config while remaining open under its original stored authority. An even earlier historical commit allowed an unrelated signer to create an alternate same-mint record without changing the victim mint's freeze authority. Executable upgrades do not migrate or close either record, so current gate setup continues to accept any stale records that still exist.

Both historical forms are sufficient to change live behavior after upgrade. A stale authority can replace an empty Allow policy with empty Block and change a real canonical-config Token-2022 thaw from failure to success. The symmetric rewrite can change a real freeze from failure to success. Supplying the stale config directly to Token ACL still fails at the final Token-2022 signer check and rolls the transaction back, but that check does not protect the separate metadata write followed by an ordinary canonical-config call.

Concrete example:

- Mint `M` has the real, current Token ACL config `C` : `C = canonical MintConfig PDA for mint M` .
- Token-2022 currently recognizes `C` as the mint freeze authority.
- The current mint authority configures thaw policy normally, for example: only allowlisted users can thaw.
- There is also an old stale Token ACL config `S` for the same mint: `S.mint = M` , `S.freeze_authority = old_authority` and `S.gating_program = gate` .
- `S` is not the live Token-2022 freeze authority anymore. The live authority is `C` .
- The old authority calls `SetupExtraMetas` and passes stale config `S` .
- The gate accepts it because `S` is Token-ACL-owned and its stored fields look valid.
- The gate then overwrites the canonical mint metadata PDA: `extra_metas = PDA(thaw_metadata, M)` .
- The live mint's thaw policy has now been changed by the old authority.
- Later, a normal user calls permissionless thaw through the real canonical config `C` .
- Token ACL uses the overwritten metadata and invokes the gate.
- Token-2022 accepts the final thaw because the call is using the real canonical config `C` .

This finding requires a deployment that ran an affected untagged historical revision and retained an alternate record. A finalized census found no such record on the observed mainnet or devnet state and mainnet's reconstructed initial binary already contained the creation fix. These conditions reduce likelihood, but an affected deployment gives a former authority or outsider active control over thaw and freeze policy. This differs from the existing config-reset finding, where unchanged metadata and old list delegates survive passively; here a second Token ACL-owned authorization record can actively overwrite the successor's metadata.

A live `MintConfig` census makes the current public exposure explicit. The check compared every exact-size Token-ACL-owned `MintConfig` account against the canonical PDA derived from its stored mint, `PDA("MINT_CONFIG", mint)` . A noncanonical result would be the stale same-mint authorization record required for this finding. None were found on the public snapshots checked: | Cluster | Finalized slot | MintConfig accounts checked | Noncanonical stale configs | | --- | ---: | ---: | ---: | | mainnet-beta | 434506701 | 17 | 0 | | devnet | 478083653 | 21492 | 0 |.

Therefore, the issue is not currently exploitable on those public snapshots. It remains a deployment-history risk for private, pruned or custom deployments that may have run affected older Token ACL code and retained an alternate config.

Recommendation: Derive the expected Token ACL config with `[b"MINT_CONFIG", mint]` under the Token ACL program ID and require `token_acl_mint_config.key()` to equal it before accepting either setup instruction. If available without weakening compatibility, also verify that Token-2022 currently names that config as the mint's freeze authority.

Before deploying the check to a cluster that may have run an affected revision, enumerate every Token-ACL-owned 100-byte discriminator-1 record and compare its key with the PDA derived from its stored mint. Close or migrate every mismatch.

Solana Foundation: Fixed in PR 21.

Cantina Managed: PR 21 adds a canonical PDA derivation check to `SetupExtraMetas` , ensuring that only the authoritative `MintConfig` for a given mint can overwrite gate policy. No noncanonical stale configs were found on the observed public snapshots at the time of the fix; the check is a preventative hardening measure.

3.2.8 Initialized default bypasses Allow policies

Severity: Low Risk.

Context: setup_extra_metas.rs#L102-L130.

Description: `SetupExtraMetas::process` binds the supplied Token ACL `MintConfig` to the mint, authority and gate, then validates each list account. It never reads the mint's current Token-2022 `DefaultAccountState`. Token ACL 0.2.3 likewise checks only the mint compatibility it requires for setup. A mint authority can therefore delegate freeze authority and install thaw metadata containing an `Allow` list while newly created token accounts still default to `Initialized`.

Thaw policy	Mint `DefaultAccountState`	Pre-existing `Initialized` accounts owned by unlisted wallets	Assessment	Consequence
Contains any `Allow` list, including mixed policies	`Initialized`	Any	**Migration / unsafe operational state**	New unlisted accounts begin transferable and never require a gated thaw.
Contains any `Allow` list, including mixed policies	`Frozen`	None	**Compatible configuration**	New unlisted accounts remain frozen until the gate authorizes thawing.
Contains any `Allow` list, including mixed policies	`Frozen`	One or more	**Existing-account migration scope**	The default protects new accounts; handling old initialized accounts is outside this finding.
No `Allow` list	`Initialized` or `Frozen`	Any	**Outside this finding**	The required default depends on the intended block-list or reactive-freeze behavior.

Token-2022 copies the mint's default state into each new token account. An account created as `Initialized` does not need permissionless thaw: Token ACL's idempotent path returns before calling the gate and ordinary transfers do not invoke Token ACL or this program. An allowlisted holder can therefore create and fund an unlisted wallet's canonical `ImmutableOwner` ATA in one transaction, after which the unlisted owner can transfer the tokens onward without a gate decision.

This relaxed setup path is intentional for existing-mint ACL migration and migration itself is out of scope. The remaining issue is a configuration/documentation hazard: an `Allow` policy can appear installed while it is not yet enforcing holder restrictions for newly created accounts. Reaching the failure requires a trusted issuer or integration to keep the mint default at `Initialized`; the documented workflow and observed live configurations use `Frozen`.

Recommendation: Keep the permissive setup path if it is required for migration, but make the inactive state explicit. Documentation, CLI and SDK setup flows should warn or require an explicit migra-

tion/unsafe flag when thaw metadata contains an `Allow` list and the mint default is not currently `Frozen`. Status or preflight tooling should tell issuers that the allow-list does not restrict newly created accounts until `DefaultAccountState` is set to `Frozen`.

If a strict on-chain mode is later added, reject mixed policies containing an `Allow` list unless the current mint default is `Frozen`.

Solana Foundation: Fixed in PR 21.

Cantina Managed: This issue is sufficiently mitigated through documentation and CLI warnings; SDK enforcement would remain advisory and bypassable via generated clients or the IDL.

3.2.9 Owner rotation lets a blocklisted account evade freezing

Severity: Low Risk.

Context: *(No context files were provided by the reviewer).*

Description: PR 21 removes the `ImmutableOwner` requirement from `CanFreezePermissionless` so that valid Token-2022 accounts with mutable owners can be frozen. However, the freeze metadata still identifies the holder by reading the current owner from bytes 32-64 of the token account. The owner is mutable on precisely the accounts that the PR newly allows.

For example, Alice owns an initialized token account and is added to a blocklist. Before the permissionless freeze executes, Alice calls Token-2022 `SetAuthority(AccountOwner)` and assigns the account to a new key. This does not change the token-account address, its initialized state or its balance.

A freeze instruction resolved before the owner change fails because Token ACL correctly detects that Alice is no longer the account owner. Resolving the instruction again does not help. The resolver now looks for a blocklist entry for the new owner. That entry does not exist, so `CanFreezePermissionless` returns `AccountAllowed`. The new owner can then transfer the account's balance.

This behavior was introduced by PR 21. At the pre-PR commit 2b99db7, the same account is rejected with `ImmutableOwnerExtensionMissing` before blocklist evaluation. At PR head commit 3797e1a, an unchanged mutable-owner account freezes successfully, while an account that changes owner evades both the old and newly resolved freeze instructions.

The impact is limited by the required ordering window. The issuer can also use its permissioned freeze authority.

Recommendation: Do not simply restore the freeze-side `ImmutableOwner` check, because that would reintroduce the issue PR 21 intended to fix.

Instead, ensure that a block decision remains attached to the same token-account address after its owner changes. One option is to store an account-level deny marker and consult it during permissionless freeze. If enforcement must remain owner-based, require `ImmutableOwner` before an account can become initialized and provide a migration procedure for existing mutable-owner accounts.

Operator tooling should also support adding the blocklist entry and freezing the target account in one transaction. This reduces the ordering window, but it does not replace the on-chain identity fix.

Solana Foundation: This is a timing attack that is impossible to avoid to a point. The flow here assumes that Alice transfers ownership of Token Account to Bob between being added to the blocklist and being permissionlessly frozen. Even if an account level deny marker is created, they can directly transfer the funds by coordinating with Bob. Adding a deny marker on account level would potentially balloon blocking costs for issuers. Ultimately this is a limitation when migrating existing mints to token-acl or when using initialized by default that should simply be documented. Fixing it by code introduces further friction to the nominal case of DefaultAccountState set to Frozen.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Upgrade invalidates existing mode-1 lists

Severity: Informational.

Context: list_config.rs#L25-L30.

Description: `ListConfig::get_mode` now maps raw mode byte `1` to `Unused`, but the released v0.2 program interpreted that same persistent byte as the supported `AllowAllEoas` mode. Program upgrades do not rewrite existing account data. The current `create_list` also rejects byte `1` and both permissionless validators reject `Unused` as an invalid list. There is no migration instruction or versioned decoder for accounts created by v0.2.

Consequently, upgrading the program without first migrating every mode-1 list makes permissionless thaw and freeze operations that reference one of those lists fail. Affected token accounts remain frozen until the mint or list authority replaces the legacy configuration. If the required authority is unavailable, the lock can be permanent.

A read-only snapshot on 2026-07-21 found seven canonical mode-1 lists on devnet, three of which were still referenced by extra-metas accounts. Mainnet was already running the March 2026 v0.2 deployment, but the snapshot found no mode-1 list accounts on mainnet at that slot. Therefore, the mainnet-specific exposure was not observed at that time. The upgrade risk remains real for any target cluster: if a mode-1 list exists and is still referenced when the current decoder is deployed, that list reference will be bricked until it is migrated or replaced.

Recommendation: Add and execute an explicit migration for every existing mode-1 list before disabling its old behavior. Keep the old mode readable during the migration window or add a migration instruction that lets the list authority select a supported replacement mode. The deployment procedure should also enumerate affected list accounts and their extra-account-meta subscribers, then block the upgrade until none remain.

Solana Foundation: There are currently no mode-1 lists, the upgrade doesn't impact existing operations.

Cantina Managed: Acknowledged.

3.3.2 Legacy count drift requires migration census only

Severity: Informational.

Context: delete_list.rs#L45-L56, remove_wallet.rs#L58-L61.

Description: The current program no longer allows a list authority to remove a wallet entry from a different list. `RemoveWallet::try_from` now verifies that the supplied `WalletEntry` belongs to the supplied `ListConfig` before `RemoveWallet::process` closes the entry and decrements `wallets_count`. Therefore, new count drift is not reachable through the current instruction set.

A pre-remediation binary could have created inconsistent legacy state by authenticating against one list while closing an entry from another list. If such state already existed, normal lifecycle calls would preserve the offset. For example, an overcounted list with stored count `1` and zero live entries cannot remove the missing entry. It also cannot be deleted because `DeleteList::process` trusts the stored count. Re-adding and removing the same wallet changes the count from `1` to `2` and back to `1`, so ordinary add/remove operations do not repair the old mismatch.

A finalized live-account census on 2026-07-22 found no affected state on the public clusters checked:

Cluster	Finalized slot	List accounts	Wallet entries	Count mismatches	Orphan entries
mainnet-beta	434462215	8	79	0	0
devnet	478032749	21409	74	0	0

This should be treated as a theoretical migration warning rather than an active vulnerability. It only matters for a deployment that previously ran the vulnerable legacy binary and already accepted a cross-list removal before the current guard was added.

Recommendation: Keep a migration census step for deployments that may have run the vulnerable legacy binary. Inventory every 74-byte list and 65-byte entry, group entries by their embedded list key and compare the result with `wallets_count`. If a mismatch is found, apply a controlled migration that writes the verified count before relying on deletion. No emergency remediation is indicated for the public mainnet-beta or devnet snapshots above because no mismatched live lists were observed.

Solana Foundation: No mismatch were found for live lists, no action required.

Cantina Managed: Acknowledged.

3.3.3 Immutable-owner parser trusts malformed TLV headers

Severity: Informational.

Context: mod.rs#L115-L131.

Description: `has_immutable_owner_extension` enters the loop whenever at least one byte remains, but then slices two bytes for the type and four bytes for the complete header without checking those bounds. It also returns `true` as soon as the first two bytes equal type 7. Therefore, it does not require the ImmutableOwner length field to exist or equal zero. Finally, it continues scanning after extension type 0 even though Token-2022 treats `Uninitialized` as the end of populated TLV data.

These differences produce fail-stop, resource-exhaustion and false-positive results. Token-2022 can initialize overallocated accounts with one, two or three unused TLV bytes, but the gate panics while examining each of those valid accounts instead of reporting a missing ImmutableOwner extension. The scan also treats zero-filled unused space as a sequence of four-byte `Uninitialized` records. A 64 KiB unused region therefore consumes the full 200,000-unit transaction budget instead of stopping at its first type-zero entry. A truncated type-7 header, a nonzero-length type-7 record and a type-7 header placed after an `Uninitialized` entry have the opposite result: Token-2022's typed lookup rejects them, while both gate instructions accept them as proof of immutable ownership.

The canonical Token ACL caller later invokes Token-2022 for the actual freeze or thaw. Supported Token-2022 instructions also do not provide a way to forge the false-positive layouts in a genuine token account. Consequently, the malformed-header cases do not currently bypass token state enforcement and the gate itself does not mutate state. The concern is still security-relevant because a safe public helper can panic on a Token-2022-created account and its `true` result is not equivalent to the ownership guarantee claimed by its callers. Any additional integration that treats gate success as an authorization result without repeating Token-2022 validation would inherit the false positive.

Recommendation: Use Token-2022's typed extension parser to query `ImmutableOwner` or reproduce all of its structural checks locally. Before every header read, verify that the required bytes remain. Stop on `Uninitialized`, reject unknown or truncated entries, check that each value ends within the account data and require the ImmutableOwner entry to have its canonical zero length. Return a `Result<bool, ProgramError>` so malformed data is a controlled error rather than a program panic. Validate the token account's owner, initialized base state and account-type byte before treating a positive result as an ownership guarantee.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.4 Generated gate client forwards unnecessary signer privilege

Severity: Informational.

Context: mod.rs#L81-L94.

Description: The Codama definitions mark `authority` as a signer for both permissionless gate instructions. This causes the generated Rust, TypeScript and CPI builders to set `is_signer = true` for account zero. The Token ACL interface defines the same account as a non-signer and the canonical Token ACL helper deliberately invokes the gate without forwarding the caller's signature. Neither local gate handler checks or needs signer status.

Therefore, an integration that uses the generated gate CPI client instead of `token-acl-interface` grants the gate more caller privilege than the standard intends. The current handlers are read-only, perform no CPI and receive the fixed policy accounts as read-only, so this does not create a present fund-loss mechanism. The practical concern is compatibility and least privilege: the published client asks callers to provide a signature that the Token ACL interface does not require. A conforming proxy that only has the caller's public key, not the caller's signature, cannot use this client as-is. If the upgradeable gate later adds signer-sensitive behavior, integrations that rely on the generated CPI client would already be forwarding unnecessary signer authority to it.

Recommendation: Remove `signer` from the `authority` Codama annotation on `CanThawPermissionless` and `CanFreezePermissionless`, then regenerate the IDL and both clients. Consider naming the account `caller` to match its actual role. Add a regression test that compares the generated account metas with `token-acl-interface` and requires account zero to be read-only and non-signing for both instructions.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.5 Gate errors collide with Token ACL errors and are absent from the generated clients

Severity: Informational.

Context: error.rs#L3-L28.

Description: `ABLError` relies on implicit, declaration-order discriminants and converts them directly into `ProgramError::Custom`. The first ten values are `0` through `9`, which overlap every custom error currently declared by the Token ACL program that invokes this gate. For example, the gate returns code `2` for `AccountBlocked`, while Token ACL assigns code `2` to `InvalidTokenProgram`. Token ACL propagates the gate error unchanged, so the transaction-level error is attached to the outer Token ACL instruction and can be decoded as the unrelated Token ACL error.

The same enum is not marked up as a Codama error definition. Consequently, `idl/token_acl_gate_program.json` contains an empty `program.errors` array and the generated Rust error module contains no error types. A client that only consumes the shipped IDL or generated SDK cannot identify `AccountBlocked`, `AccountAllowed` or malformed-account failures. Monitoring and automated incident handling can therefore classify an ACL policy rejection as a configuration fault. In addition, inserting or reordering a variant would silently renumber later errors and change the public error ABI.

The pinned `token-acl-interface` 0.2.3 library also exposes a `ProgramError` conversion that assumes every custom code belongs to `AccountResolutionError`, then unwraps that decode. None of the gate's codes 0 through 17 belongs to that high-numbered namespace. Therefore, a consumer that sends a propagated gate error through this public conversion panics instead of preserving the unknown custom error. The canonical on-chain caller and the repository's instruction builders do not currently convert execution failures this way, so this is just an additional client-integration hazard.

Recommendation: Treat the numeric mapping as a public ABI. Assign every variant an explicit discriminant and add a regression test for all values. Preserve the existing values if compatibility with deployed clients is required; otherwise reserve a versioned, non-overlapping range for gate errors. Add Codama error metadata with a message and matching explicit code for every variant, regenerate the IDL and SDKs and test that `program.errors` plus the Rust and TypeScript decoders contain all variants. Clients that decode nested CPI failures should use the failing program id from execution logs instead of assuming that the outer instruction owns the custom code. The interface conversion should retain an unrecognized `ProgramError::Custom` value instead of unwrapping an `AccountResolutionError` decode.

Solana Foundation: Fixed in PR 21 and PR 38.

Cantina Managed: PR 21 assigns gate errors explicit non-overlapping codes (`4096` – `4113`) and adds generated decoders in `error.rs`. PR 38 fixes the `token-acl-interface` library so that unrecognized custom error codes are preserved rather than causing a panic.

3.3.6 Gate dispatch accepts non-standard discriminator aliases

Severity: Informational.

Context: mod.rs#L76-L100, lib.rs#L30-L40.

Description: The Token ACL interface defines `CanThawPermissionless` and `CanFreezePermissionless` with eight-byte discriminators. Its builders emit those eight bytes and its decoder requires and compares the complete eight-byte prefix (while currently ignoring any later bytes). The `Instruction` enum records only the first byte of each discriminator. `process_instruction` likewise reads only byte zero, then sends every matching byte string to the same handler. Consequently, `[8]`, `[8, 0, 0, 0, 0, 0, 0, 0]` and the canonical thaw discriminator are indistinguishable to this program. The same ambiguity exists for every byte string beginning with `214` on freeze.

The canonical Token ACL 0.2.3 caller constructs the correct eight-byte values and both gate handlers are read-only. Therefore, the aliases do not currently bypass a freeze or thaw policy. However, the gate does not provide the instruction domain separation promised by the interface. A future interface instruction with the same first byte can be interpreted as today's thaw or freeze check. The Codama IDL and generated clients also encode the non-standard one-byte form, which makes the published gate ABI disagree with the interface it implements.

Recommendation: Compare the complete eight-byte discriminator before dispatching either interface instruction and reject one-byte, truncated and wrong-suffix encodings. Decide explicitly whether bytes after that prefix are permitted and keep the runtime consistent with the interface decoder. Keep the private management instructions on their existing one-byte opcodes if compatibility requires it, but represent the two Token ACL interface instructions with their complete discriminators in the IDL and generated clients.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.7 Checkout path newlines inject Cargo directives

Severity: Informational.

Context: build.rs#L30.

Description: `generate_idl` inserts `idl_path.display()` directly into a `cargo:warning=` line. Cargo treats each newline-delimited line beginning with `cargo:` as a build instruction, but Unix filenames may contain newline characters. A newline in the checkout directory can therefore terminate the warning and introduce a second Cargo directive.

This was reproduced with an otherwise unchanged checkout inside a directory named with embedded `cargo:rustc-cfg=buildrs_path_injected` and `cargo:rustc-env=BUILD_RS_PATH_INJECTED=YES` lines. Cargo accepted both lines. It then invoked `rustc` with `--cfg buildrs_path_injected` and with `BUILD_RS_PATH_INJECTED=YES` in the compiler environment.

An attacker must be able to choose the directory in which an otherwise trusted copy of this repository is built. The documented GitHub workflows do not expose that control. However, a build service or local automation that accepts an untrusted workspace name can have compiler configuration, compiler environment variables or linker arguments injected without any source change. The current program does not condition behavior on the demonstrated custom configuration, so this is a build-integrity hardening issue rather than a direct on-chain vulnerability.

The error warning on line 10 should be treated the same way because some error strings may contain filesystem paths.

Recommendation: Do not place unsanitized paths or errors in Cargo instruction output. Remove the success warning or write it to stderr. If a Cargo warning is retained, replace carriage returns, newlines and other control characters with printable escaped forms before interpolation. Apply the same sanitization to generation errors.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.8 Gate validation is not bound to the supplied owner

Severity: Informational.

Context: can_thaw_permissionless.rs#L48.

Description: Both gate handlers accept an `owner` account, but neither uses it to validate the supplied `WalletEntry`.

In the thaw handler, owner is explicitly unused:

```
fn validate_thaw_list(
      list: &AccountInfo,
      _owner: &AccountInfo,
      wallet_entry: &AccountInfo,
  )
```

For an Allow list, the handler only checks that the wallet entry is owned by the gate program and that `wallet_entry.list_config` matches the supplied list. It never checks `wallet_entry.wallet_address`.

The freeze handler has the same issue: `owner` is not passed to `validate_freeze_list`, and an initialized wallet entry is considered present based only on its `list_config`.

As a result, a caller that constructs the CPI manually can substitute another wallet's entry. For example, if Alice is allowlisted and Bob is not, the caller can pass Bob as owner and Alice's entry as the remaining account. The thaw gate returns Ok because Alice's entry belongs to the correct list, even though it has no relationship to Bob.

The inverse cases are also affected. Where the policy expects an entry to be absent, the caller can supply an unrelated empty system account and have it treated as a missing entry. Checking `wallet_entry.wallet_address` alone would therefore not address every case.

The canonical Token ACL flow is not affected because its TLV resolver derives the expected wallet-entry address before invoking the gate. The issue affects integrations that call the gate directly and interpret Ok as an authorization result.

Recommendation: Validate the resolved accounts inside the gate instead of relying solely on the caller.

Use `ExtraAccountMetaList::check_account_infos` with the canonical extra-metas account and the appropriate thaw or freeze discriminator. This verifies that the remaining accounts match the configured metadata and that dynamically resolved wallet-entry addresses are correct.

Alternatively, derive every expected wallet-entry PDA from:

```
["wallet_entry", list.key(), owner.key()]
```

and require the supplied account to match it. For initialized entries, also verify:

```
wallet_entry.list_config == list.key()
wallet_entry.wallet_address == owner.key()
```

The gate should additionally derive and validate the operation-specific extra-metas PDA so a caller cannot supply an arbitrary policy configuration.

Solana Foundation: Given that the current gates are read-only, a malformed request from a different program has no impact on the program data. Its difficult to justify an additional ~20% CU overhead

on re-calculating PDAs and additional checks when this should be up to the caller to properly supply these inputs.

Cantina Managed: Acknowledged.

3.3.9 Unsafe state casts lack layout guarantees

Severity: Informational.

Context: mod.rs#L67, mod.rs#L82.

Description: `load_unchecked` casts raw account bytes to generic state after checking only a manually maintained length. Current layouts are safe, but future aligned or non-POD fields could introduce undefined behavior.

Recommendation: Require `Pod + Zeroable`, use checked `bytemuck` loaders, and add compile-time size/alignment assertions.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.10 Mutable account-data reference is retained across CPIs

Severity: Informational.

Context: add_wallet.rs#L28-L29, add_wallet.rs#L75.

Description: `AddWallet` holds a mutable `ListConfig` reference while executing multiple System Program CPIs, complicating borrow and aliasing safety.

Recommendation: Validate and release the borrow before CPIs, then reborrow the list afterward to increment `wallets_count`.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.11 `README` documents obsolete modes and incorrect instruction discriminators

Severity: Informational.

Context: *(No context files were provided by the reviewer)*.

Description: The `README` presents `AllowAllEoas` as supported and lists opcodes that no longer match the program, potentially causing incorrect integrations (or) confusion among integrators.

Recommendation: Update the mode and instruction tables from the current source and verify them automatically during CI.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.

3.3.12 Tests may execute a stale checked-in SBF binary

Severity: Informational.

Context: program_test.rs#L41-L44.

Description: The LiteSVM harness loads a fixture binary without verifying that it was built from the current program source, allowing tests to pass against outdated code.

Recommendation: Build the SBF artifact before tests or compare its hash against a fresh build and fail CI on mismatch.

Solana Foundation: Fixed in PR 21. The assumption here would be that if someone has a fresh clone, the binary would be in sync with the codebase, and when they do changes they would at least do a build for rustc checks. On a worst case someone writes code → runs tests without fresh build → opens PR, the CI should catch it then and force them to update the binary on the PR branch.

Cantina Managed: PR 21 fixes the CI path so that a fresh SBF binary is built and copied before tests run. A local `cargo test` without a prior SBF build can still execute the stale checked-in fixture; the Solana Foundation considers this acceptable because CI will enforce a fresh build before merge.

3.3.13 Existing mint adoption needs account migration

Severity: Informational.

Context: setup_extra_metas.rs#L102-L130.

Description: `SetupExtraMetas::process` can configure Token ACL allow-list metadata for a mint after checking only the supplied Token ACL `MintConfig` and list accounts. It does not require any proof that already-existing token accounts for the mint have been frozen or migrated before the policy metadata is accepted.

This is not a Token-2022 bug. Token-2022 behaves as designed: updates to `DefaultAccountState` only affect accounts initialized after the update. An issuer can switch a mint's current default to `Frozen`, enable Token ACL with an `Allow` list and still leave older unlisted accounts in the `Initialized` state. Those accounts are already usable, so they do not need a successful gated thaw. Ordinary Token-2022 transfers do not invoke Token ACL or this gate, so the holder can continue transferring after adoption.

The local test demonstrated this exact migration state: after adoption, a fresh unlisted ATA is created `Frozen`, proving that the current mint default is safe for new accounts, while the older unlisted ATA remains `Initialized`. A permissionless thaw for the older holder is rejected by the gate, but a normal transfer from that account succeeds without any Token ACL or gate invocation.

Based on the protocol-team clarification, the relaxed setup path exists so preexisting mints can migrate to ACL and the migration itself is out of scope for this program. Under that stated scope, this is an operational/documentation boundary: ACL adoption only restricts accounts that are frozen before they need a gated thaw and issuers must handle any older initialized accounts separately before relying on ACL for those balances.

Recommendation: Document existing-mint adoption as a migration. Before relying on an allow-list policy for preexisting balances, issuers should enumerate token accounts for the mint and freeze or migrate every initialized account whose owner is not allowed. The setup guide, CLI and SDK should state clearly that `DefaultAccountState::Frozen` protects only accounts created after the update and does not retroactively freeze old accounts.

Solana Foundation: Fixed in PR 21.

Cantina Managed: Verified fix.