Adevar Labs

Prepared for:

M0 Foundation

October 30, 2026

Issuance V2 SVM Audit Report

Version: after fix review

Table of Contents

1. Executive Summary 4

About M0 Issuance V2 4

Audit Summary 4

Overall Security Posture 4

Launch Recommendations 5

Audit Scope 6

2. Assumptions and Considerations 7

Audit Assumptions 7

Centralization Risks 7

Trust Assumptions 7

3. Severity Definitions 8

Impact 8

Likelihood 8

Severity Classification Matrix 8

4. Findings 10

M01: adopt_stablecoin accepts Confidential Transfer and Metadata Pointer states that completion refuses 10

M02: claim_accrued mints a holder's burned amount as yield on an adopted mint without Permissioned Burn 13

L01: The documented transfer-hook list bound is twice the real ceiling, and exceeding it bricks every transfer 16

L02: A mint paused with a renounced pause authority completes adoption and can never be resumed 18

5. Enhancement Opportunities 20

E01: Aligning the migration guide with update_token_metadata will make clear that custom metadata keys cannot be repaired after adoption 20

© 2026 Adevar Labs. All rights reserved.

E02: Funding Token-2022's padded length will keep write_token_metadata from failing at the 355-byte Multisig boundary 22

E03: Letting the admin revoke keys and cancel mints while paused will improve pause_gateway 23

E04: The CLI launches every stablecoin with an unbounded mint rate limit 25

E05: Optional delay on admin rotation will give the issuer time to cancel a hostile rotation 26

E06: Enforcing distinct proposer and executor keys in execute_mint will improve issuance safety 28

About Us 31

About Adevar Labs 31

Audit Methodology 31

Confidentiality Notice 33

Legal Disclaimer 33

© 2026 Adevar Labs. All rights reserved.

1. Executive Summary

About M0 Issuance V2

M0 Issuance V2 on Solana is the M0 Issuer Gateway, a native program that each issuer forks, builds and deploys under its own keys. A stablecoin is a Token-2022 mint whose authorities are all held by a program PDA, which signs only after the gateway's checks pass. The gateway adds what Token-2022 cannot express: a mint delay, a rate limit, a circuit breaker, roles scoped to one mint, freeze and thaw routing through Token ACL, and Revenue Direct rewards. One deployment lets an issuer manage its whole fleet of stablecoins from a single program. It can also adopt live V1 M extension mints. The **moonpay** branch adds **eject_authority**, which moves one authority off the PDA to a key the issuer names. This report covers the SVM program. The EVM contracts are covered in a separate report.

Audit Summary

The table below summarizes identified vulnerabilities by severity and remediation status.

Severity	Count	Fixed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	2	2	0
Low	2	2	0

Enhancement opportunities: 6

Overall Security Posture

Account validation is thorough, every role grant is scoped to one mint, and Token-2022 enforces pause and freeze on every holder operation, so the gateway stays out of the transfer path. We found no Critical or High issues: 2 Medium and 2 Low issues, mostly in edge cases of V1 adoption, and 6 enhancement opportunities.

© 2026 Adevar Labs. All rights reserved.

Before Fix Review Summary

The 2 Medium issues both concern adopted mints. On a mint without Permissioned Burn, holders can burn directly through Token-2022, which lowers supply but not the gateway's principal, so **claim_accrued** mints the burned amount to the rewards recipient. **adopt_stablecoin** also accepts Confidential Transfer and Metadata Pointer states that **complete_adoption** refuses, so after the V1 handoff such a mint can never finish adoption. The 2 Low issues are a mint paused with a renounced pause authority, which completes adoption and can never be resumed, and a documented transfer-hook account limit twice the real ceiling, past which every transfer fails. The enhancements suggest distinct proposer and executor keys, an optional delay on admin rotation, explicit rate limits in the CLI, incident tools that work while the gateway is paused, a metadata rent top-up that matches Token-2022's padded length, and a migration guide that matches what **update_token_metadata** can repair.

After Fix Review Summary

M0 fixed all four issues. Adoption now refuses a stuck Confidential Transfer or Metadata Pointer authority and a renounced pause or Scaled UI authority, and completion resumes a mint paused during the window. The gateway tracks the supply it last saw, so burns made outside it are no longer paid out as yield. Arming a transfer hook now refuses lists longer than the measured limit of 15 entries.

M0 also implemented five of the six enhancements: a fixed admin rotation delay chosen at deployment, a check that stops a non-admin proposer from executing its own mint, admin incident actions while the gateway is paused, a metadata rent top-up that covers Token-2022's padding, and a migration guide that matches what **update_token_metadata** can repair. The CLI's unbounded rate-limit default is acknowledged.

We then reviewed M0's final commit, **2c8e101**, which adds a refactor of account validation and state loading (PR #68) and license headers (PR #69). We found no regressions and no new issues: every fix is still in place, and each account check is the same as before or stricter. Two notes for future changes. **Account::load** now returns a copy, so a write to that copy compiles but is lost, and the gate's copy of a **Stablecoin** goes stale if the account is written later in the same instruction. Neither happens in the current code, and returning a read-only wrapper from **load**, with no reads of the gate's copy after a **load_mut**, would keep it that way. The IDL also still maps error codes 3 and 4 to the account checks, which now return Solana's built-in errors.

Launch Recommendations

Before the fix review, we recommended:

1. Keep the gateway's principal in step with supply on adopted mints without Permissioned Burn, or refuse such mints at adoption.
2. Make **adopt_stablecoin** refuse every state that **complete_adoption** refuses, including a paused mint whose pause authority is vacant.
3. Correct the documented transfer-hook account limit to the measured ceiling, and enforce it.

After the fix review, all three are implemented.

© 2026 Adevar Labs. All rights reserved.

Before launch, we recommend:

- Deploy a build of the reviewed commit **2c8e101**, or of a later commit that has been reviewed.
- Set a non-zero admin rotation delay in **initialize_issuer_config**. It cannot be changed afterwards, and **0** disables it.
- Pass explicit rate-limit and mint-delay values when launching a stablecoin with the CLI, instead of its unbounded defaults.
- Check the mint's custom metadata keys before **complete_adoption**, as the migration guide now says.

After launch, alert on admin proposals and rotations, role grants, **eject_authority** and any authority change on a gateway mint, and changes to rate limits and the earner rate. After each adoption, reconcile the gateway's principal and index against the mint's supply.

Audit Scope

- **Repository:** https://github.com/m0-platform/issuance-v2 (branch **moonpay**, merged into **main** with the fixes)
- **Before Fix Review Commit Hash:** **3ac60b772e4985141bc627e7f11b034fc94d190e**
- **After Fix Review Commit Hash:** **2c8e10109204d77e4602372cc0b13f8741cebe83**
- **Files/Modules in Scope:**
  - **svm/programs/issuer-gateway/src**
  - **svm/crates/state/src**
- **Exclusions:**
  - The rest of **svm/**: the SDK and interface crates, the CLI, the JavaScript client, runbooks, scripts and test programs
  - Token-2022, Token ACL and Token ACL Gate, reviewed only in how the gateway uses them
- **Extended Scope:**
  - **Pull Request:** https://github.com/m0-platform/solana-m-extensions/pull/73
  - **Commit Hash:** **2e19f9ba03b0f0c7ce33a131af4f8dd5d1fc264b**
  - **Added Files/Modules:** **programs/m_ext/src/instructions/handoff.rs**, with its entry point, error codes and **handoff** feature wiring in **programs/m_ext/src/lib.rs**, **programs/m_ext/src/instructions/mod.rs**, **programs/m_ext/src/errors.rs** and **programs/m_ext/Cargo.toml**

All remediation pull requests are merged into **main**: #46 directly, and #50 to #57 through PR #49. We reviewed the fixes at **c2410b44a1bc6f9e313010460a5bd6ec49f83f33**, the last commit of that series, and then the final **main** commit **2c8e10109204d77e4602372cc0b13f8741cebe83**, which adds the refactor in PR #68 and license headers in PR #69.

© 2026 Adevar Labs. All rights reserved.

2. Assumptions and Considerations

Audit Assumptions

I. Issuer builds and keys: Each issuer builds with its own **INITIALIZER** key and program id (the key in the repository is a development placeholder), deploys a build of an audited commit, and holds the admin, the upgrade authority and the program id keypair in multisigs.

II. V1 cutover: The V1 program that runs **handoff** is built from solana-m-extensions PR #73 at **437f087** with the no-yield and handoff features, and its upgrade authority deploys that build only for the cutover. The documented cutover procedure stops V1 wraps and unwraps before adoption, so the supply does not change. It moves every remaining authority to the PDA and renounces none of them. The operator manually checks the mint metadata before **complete_adoption**.

III. V1 freeze state: Any holder V1 blocked is frozen on the mint before the cutover completes.

IV. Token ACL and transfer hooks: Token ACL and Token ACL Gate are the canonical Solana Foundation deployments. An issuer that arms a transfer hook deploys and vets the hook program first.

V. Authority ejection: An issuer that ejects an authority accepts what it breaks: minting outside the rate limit and circuit breaker, yield accounting that no longer matches supply, and gateway controls that stop working. MoonPay accepted this trade-off.

VI. Confidential transfer approval: Approval covers a token account, not its holder. Token-2022 keeps the approval when the account changes owner, and a holder can always hand over the account's key, so the issuer does not rely on approval to keep confidential transfers with the holder it reviewed.

Centralization Risks

- The issuer admin passes every role check without a grant. It can mint through the proposal flow, seize, pause, grant and revoke roles, and eject authorities.
- The issuer's upgrade authority can replace the program for every stablecoin in the deployment at once.
- After **eject_authority**, the named key holds that Token-2022 authority directly, outside every gateway check.

Trust Assumptions

- Off-chain reserves back the supply 1:1, and the issuer alone is responsible for them.
- The issuer, M0 and the partner project a coin is issued for do not collude.
- Token-2022 enforces pause and the frozen bit on every transfer, including transfers that never touch the gateway.
- The V1 $M swept during the handoff goes to an account M0 controls.

© 2026 Adevar Labs. All rights reserved.

3. Severity Definitions

Each issue identified in this report is assigned a severity level based on two dimensions: **Impact** and **Likelihood**. These dimensions help convey both the potential consequences of a vulnerability and how likely a vulnerability is to be discovered and exploited in the real world.

Note: Enhancements represent non-blocking improvements, typically usability, observability, or defense-in-depth tweaks that do not pose an immediate asset risk but would improve the product's reliability and/or user experience if implemented.

Impact

Impact reflects the potential consequences of the issue, particularly on **project funds**, **user funds**, and the **availability or integrity** of the protocol.

- **High Impact**: Successful exploitation could result in a complete loss of user or protocol funds, disruption of core protocol functionality, or permanent loss of control over critical components.
- **Medium Impact**: Exploitation could cause significant disruption or partial loss of funds, but not a total compromise. May impact some users or non-core functionality.
- **Low Impact**: The issue has minor or negligible consequences. It may affect edge cases, expose metadata, or degrade performance slightly without putting funds or core logic at serious risk.

Likelihood

Likelihood reflects how easy a vulnerability is to discover and exploit by an attacker, as well as how economically attractive the exploit is to an attacker.

- **High Likelihood**: The vulnerability is trivially exploitable. This means it can be exploited by a wide range of actors without privileged access rights, with minimal capital requirements and low financial risks.
- **Medium Likelihood**: This type of vulnerability can be found and exploited with moderate effort. It might require a significant capital investment, but with manageable financial risk.
- **Low Likelihood**: Exploitation of these vulnerabilities is often technically infeasible or requires highly specialized conditions. They may require extraordinary effort or a significant financial risk for an attacker, with a high chance of failure and minimal potential return.

Severity Classification Matrix

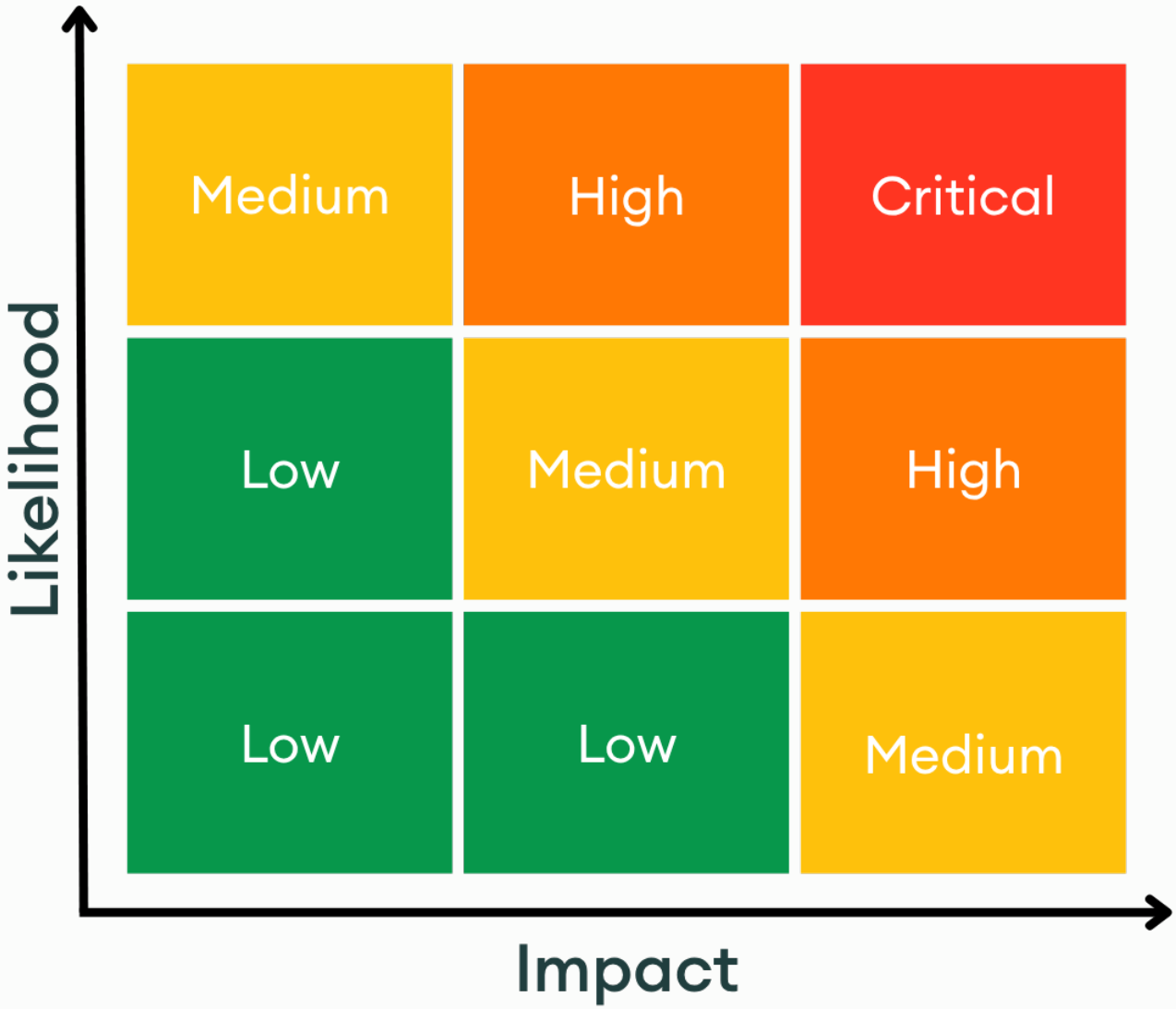
By combining **Impact** and **Likelihood**, we assign a severity level using the matrix below:

- **Critical**: High impact + high likelihood (e.g. a bug that could allow anyone to drain a substantial amount of protocol funds with minimal effort)
- **High**: High impact with medium likelihood, or medium impact with high likelihood
- **Medium**: Moderate impact and/or discoverability

© 2026 Adevar Labs. All rights reserved.

- **Low**: Minimal impact or unlikely to be exploited

This structured approach helps teams prioritize fixes and mitigate the most dangerous threats first.



Severity Matrix

© 2026 Adevar Labs. All rights reserved.

4. Findings

M01: adopt_stablecoin accepts Confidential Transfer and Metadata Pointer states that completion refuses



LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/lifecycle/adopt_stablecoin.rs:L84-L132

RELEVANT SNIPPET

>_ adopt_stablecoin.rs — RUST

```rust
/// Check the mint's extensions against [`ADOPTABLE_EXTENSIONS`] and return
/// their bitmap. Each refusal names a state completion could never repair.
fn check_extensions(
    mint: &StateWithExtensions<'_, Mint>,
    mint_data: &[u8],
    stablecoin_pda: &Address,
) -> Result<u16, ProgramError> {
    let mut listed = [ExtensionType::Uninitialized; MAX_EXTENSIONS];
    let count =
        ListedState::<ListedMint>::from_bytes(mint_data)?.write_extension_types(&mut listed)?;
    let mut present = 0u16;
    for ext_type in &listed[..count] {
        // A closeable mint could be closed at zero supply and its address reused.
        if *ext_type == ExtensionType::MintCloseAuthority {
            return Err(GatewayError::MintCloseAuthorityPresent.into());
        }
        let bit = ADOPTABLE_EXTENSIONS
            .iter()
            .position(|t| t == ext_type)
            .ok_or(GatewayError::UnsupportedExtension)?;
        present |= 1 << bit;
    }
    // A live hook would run an unknown program on every holder transfer.
    if mint
        .get_extension::<TransferHook>()
        .is_ok_and(|hook| hook.program_id.as_ref().is_some())
    {
        return Err(GatewayError::TransferHookActive.into());
    }
    // The delegate can move balances during the pending window, so it must
```

© 2026 Adevar Labs. All rights reserved.

>_ adopt_stablecoin.rs RUST

```rust
    // already be the stablecoin PDA, or nobody.
    if mint
        .get_extension::<PermanentDelegate>()
        .is_ok_and(|pd| pd.delegate.as_ref().is_some_and(|d| d != stablecoin_pda))
    {
        return Err(GatewayError::PermanentDelegateStillExternal.into());
    }
    // A cleared burn authority reopens holder self-burns for good.
    if mint
        .get_extension::<PermissionedBurnConfig>()
        .is_ok_and(|burn| burn.authority.as_ref().is_none())
    {
        return Err(GatewayError::BurnAuthorityRevoked.into());
    }
    // Renounced metadata can never be repaired.
    let metadata_authority = token_metadata_update_authority(mint)?;
    if metadata_authority.is_some_and(|a| a.as_ref().is_none()) {
        return Err(GatewayError::MetadataAuthorityRevoked.into());
    }
    Ok(present)
}

/// TokenMetadata's update authority, or `None` without TokenMetadata. Only the
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/lifecycle/complete_adoption.rs:L169-L177
- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/lifecycle/complete_adoption.rs:L186-L193

DESCRIPTION

adopt_stablecoin runs before the V1 **handoff** so it can refuse any mint that completion could never finish. Its module header (L8-L10) explains why: "If authorities moved first and adoption then rejected the mint, the mint authority would sit at a PDA no handler ever signs with."

complete_adoption refuses two states that **check_extensions** does not verify:

- A Confidential Transfer policy with auto-approve or an auditor key under an empty authority, which fails with **ConfidentialTransferPolicyStuck**
- A Metadata Pointer that does not point at the mint, under an empty authority, which fails with **MetadataPointerStuck**

Token-2022 never restores a cleared authority, so both errors are permanent for that mint. The V1 CLI does not create either state, but the authority holder can reach one at any time before adoption by changing the setting and then calling **SetAuthority** with **None** and Token-2022 allows both. **check_extensions** accepts these mints because it reads neither extension's state. The Transfer Hook one does check it: **adopt_stablecoin** refuses a live hook, so **TransferHookStuck** can only come from a change during the pending window. The two other settings have no matching check.

The issuer only sees the failure later: **plan_resets** returns **AdoptionIncomplete** until the mint and freeze authorities equal the PDA, so the admin cannot observe the stuck error before **handoff**.
After cutover, every **complete_adoption** call fails:

© 2026 Adevar Labs. All rights reserved.

- **abort_adoption** closes the stablecoin account and moves no authority.
- A second **adopt_stablecoin** succeeds because the mint still passes **check_extensions**, and completion fails again for the same reason.
- Every other handler, **eject_authority** included, goes through **verify_stablecoin**, which refuses a pending account and cannot load a closed one.

RECOMMENDATION

adopt_stablecoin should refuse a mint that **complete_adoption** would refuse. Add two checks to **check_extensions**:

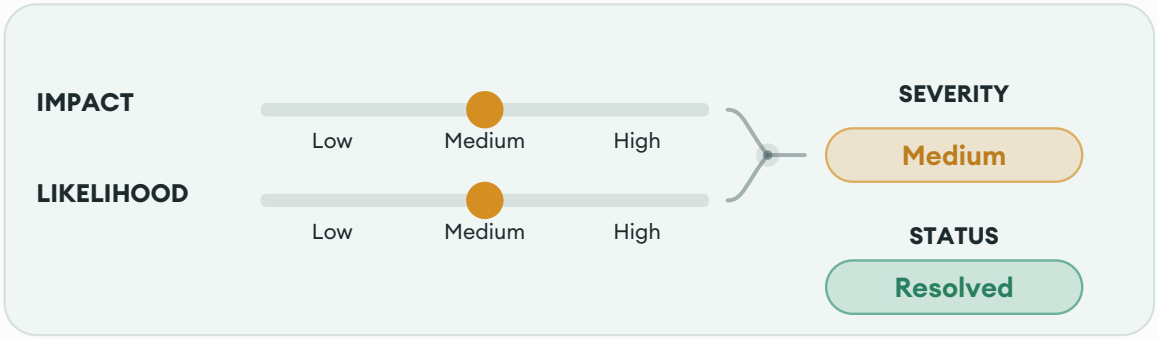
- Refuse with **ConfidentialTransferPolicyStuck** when the Confidential Transfer authority is empty and auto-approve is on or an auditor key is set.
- Refuse with **MetadataPointerStuck** when the Metadata Pointer authority is empty and the pointer does not point at the mint.

FIX REVIEW NOTES

Fixed according to the recommendation: **adopt_stablecoin** now refuses a stuck Confidential Transfer or Metadata Pointer authority before the handoff.

© 2026 Adevar Labs. All rights reserved.

M02: claim_accrued mints a holder's burned amount as yield on an adopted mint without Permissioned Burn



LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/util/index_math.rs:L170-L172

RELEVANT SNIPPET

>_ index_math.rs — RUST

```
168:
169: /// `present(principal) − supply`, floored at zero.
170: pub fn rewards_owed(principal: u64, index: u64, supply: u64) -> Result<u64, ProgramErr
       or> {
171:     Ok(principal_to_amount_down(principal, index)?.saturating_sub(supply))
172: }
173:
174: #[cfg(test)]
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/lifecycle/adopt_stablecoin.rs:L120-L125

DESCRIPTION

rewards_owed returns **principal_of_supply * latest_index / 1e12 - supply**. **supply** comes live from the mint, **principal_of_supply** from the **Stablecoin** PDA. A supply change must move both.

1. **adopt_stablecoin** seeds **principal_of_supply = supply** and **latest_index = INDEX_SCALE**.
2. **check_extensions** probes **PermissionedBurnConfig** with **is_ok_and**, which is **false** when the extension is absent. **complete_adoption**'s **require_pda** passes the same absence.
3. A holder calls Token-2022 **Burn** directly, and **supply** drops.
4. **burn** alone lowers **principal_of_supply**. Its **PermissionedBurn** CPI cannot run on this mint, so the field never moves.

The terms diverge permanently. **claim_accrued** reads the gap as accrued yield and mints it to **rewards_recipient**. It calls **update_index**, not **update_index_and_consume**, so the bucket is never debited. At **earner_rate_bps = 0** the mint equals the burn exactly, because **principal_to_amount_down** short-circuits on an identity index.

Nothing repairs the field: **abort_adoption** needs **AdoptionPending**, and **adjust_stablecoin** has no principal setter.

svm/docs/005-migration.md marks Permissioned Burn "missing, forever" on the reference mint, and names self-burn as the holder redemption path.

© 2026 Adevar Labs. All rights reserved.

PROOF OF CONCEPT

One adopted mint with:
- 6 decimals, no Permissioned Burn extension
- Supply and **principal_of_supply** both 1,000,000.000000
- **latest_index** = 1e12, **earner_rate_bps** = 0
- **rate_limit_capacity** = 100.000000, fully available

What happens then:
1. A holder sends Token-2022 **Burn** for 400.000000. Supply falls to 999,600.000000; **principal_of_supply** stays at 1,000,000.000000.
2. A **Claimer** calls **claim_accrued**. **rewards_owed(1_000_000_000_000, 1_000_000_000_000, 999_600_000_000)** returns **400_000_000**.
3. **MintTo** issues 400.000000 to **rewards_recipient**, four times the bucket's whole capacity, and **rate_limit_remaining** stays at 100.000000.

Result: supply returns to 1,000,000.000000 against reserves of 999,600.000000. The loop repeats per redemption.

Same sequence on a registered mint: Token-2022 rejects the holder's **Burn**, because **register_stablecoin** places **PermissionedBurn** in the unconditional floor extension set.

RECOMMENDATION

Record the supply the gateway last saw, and treat a drop below it as a burn. Add **last_seen_supply: u64** to **Stablecoin**, seeded to 0 at registration and to the adopted supply at adoption.

RUST

```rust
fn reconcile_supply(sc: &mut Stablecoin, supply: u64, index: u64) -> Result<(), ProgramError>
  {
    if supply < sc.last_seen_supply {
        let burned = sc.last_seen_supply - supply;
        sc.principal_of_supply = sc
            .principal_of_supply
            .saturating_sub(amount_to_principal_up(burned, index)?);
    }
    sc.last_seen_supply = supply;
    Ok(())
}
```

Call it before the CPI in **claim_accrued**, **execute_mint**, **distribute_rewards** and **burn**, then stamp the live supply after. A rise leaves the principal alone, since only supply with principal behind it earns. This changes the account layout, so bump **ACCOUNT_VERSION**.

A burn outside the gateway then earns accrual only until the next gateway call, which is zero at **earner_rate_bps = 0**.

If that is too large, have **claim_accrued** refuse an adopted mint without the extension, and pay the partner through **distribute_rewards**. Either fix keeps such mints adoptable, which the reference mint depends on, since a mint can never gain an extension.

© 2026 Adevar Labs. All rights reserved.

FIX REVIEW NOTES

Fixed: the gateway now records the supply it last saw and debits principal for anything burned outside it, so an outside burn is no longer paid out as yield. The burned amount only accrues until the next gateway call.

© 2026 Adevar Labs. All rights reserved.

L01: The documented transfer-hook list bound is twice the real ceiling, and exceeding it bricks every transfer



LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/compliance/force_transfer.rs:L112-L115

RELEVANT SNIPPET

>_ force_transfer.rs — RUST

```
110: }
111:
112: /// Room for a live hook's account demands: the hook program, its
113: /// extra-account-metas PDA, and up to thirty list entries. No-alloc CPI needs a
114: /// fixed bound, so a hook demanding more must be refused when it is vetted.
115: const MAX_HOOK_ACCOUNTS: usize = 32;
116:
117: /// Zero-copy instruction payload for `ForceTransfer` (8 bytes).
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/docs/002-compliance.md:L37-L39

DESCRIPTION

MAX_HOOK_ACCOUNTS is 32, and both its doc comment and **docs/002-compliance.md** tell the operator to vet a hook's list against 30 entries. The measured ceiling is 15.

At 16 entries Token-2022's **add_to_cpi_instruction** exhausts its 32 KiB bump heap. **request_heap_frame** does not move it, because **HEAP_LENGTH** is a compile-time constant in the allocator.

Every transfer then fails, not only seizure: a plain owner **TransferChecked** hits the same wall. The documented dormant-toggle recovery restores seizure and leaves holder transfers bricked. The exact ceiling tracks the deployed Token-2022 build.

RECOMMENDATION

Validate the hook's **["extra-account-metas", mint]** list when **adjust_stablecoin** arms the program, and refuse a count at or above the measured ceiling. Correct the bound in the doc comment and in **docs/002-compliance.md**. The hook's own authority can still grow the list afterwards, so this bounds the state at arming only.

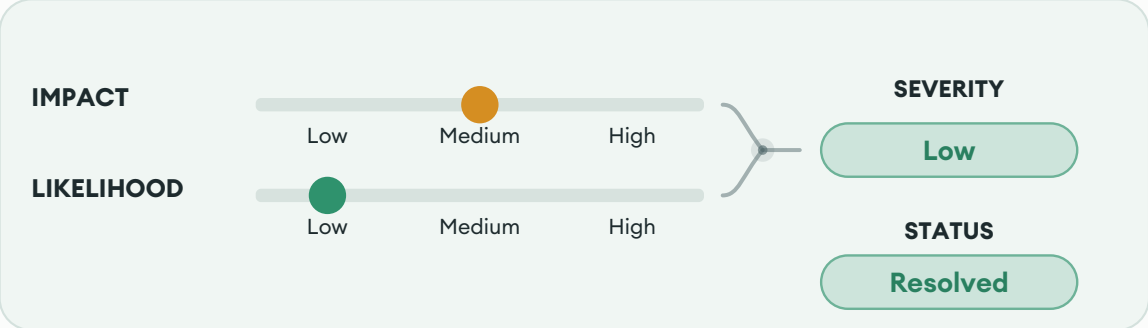
© 2026 Adevar Labs. All rights reserved.

FIX REVIEW NOTES

Fixed: arming a transfer hook now refuses an extra-accounts list longer than 15 entries, the limit we measured.

© 2026 Adevar Labs. All rights reserved.

L02: A mint paused with a renounced pause authority completes adoption and can never be resumed



LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/lifecycle/complete_adoption.rs:L90-L99

RELEVANT SNIPPET

>_ complete_adoption.rs — RUST

```
88:     mint.get_extension::<ScaledUiAmountConfig>()
89:         .ok()
90:         .map(|e| &e.authority),
91:     mint.get_extension::<PausableConfig>()
92:         .ok()
93:         .map(|e| &e.authority),
94: ];
95: for slot in pda_or_vacant.into_iter().flatten() {
96:     if slot.as_ref().is_some_and(|k| k != stablecoin) {
97:         return Err(GatewayError::AuthorityRetained.into());
98:     }
99: }
100: let burn = mint
101:     .get_extension::<PermissionedBurnConfig>()
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/lifecycle/complete_adoption.rs:L142-L147

DESCRIPTION

complete_adoption refuses a setting left off baseline when its authority is vacant. **needs_reset** applies that to the confidential-transfer policy, the transfer hook and the metadata pointer. **PausableConfig.paused** is read by neither arm, and the Pausable slot sits in **pda_or_vacant**, where **is_some_and** passes a vacant authority.

A mint paused during the pending window with its pause authority renounced therefore completes as **Adopted**. Token-2022 requires a present authority in **process_toggle_pause** and in **SetAuthority**'s **Pause** arm, so **resume** can never be signed and every transfer, mint and burn fails permanently.

© 2026 Adevar Labs. All rights reserved.

RECOMMENDATION

Add a fourth **needs_reset** call for the paused bit with its own **PausableStuck** variant, and move **PausableConfig** from **pda_or_vacant** into the **require_pda** set. Apply the same move to **ScaledUiAmountConfig**, whose multiplier is read by neither arm either.

FIX REVIEW NOTES

Fixed: adoption now refuses a renounced Pausable or Scaled UI authority, and **complete_adoption** resumes a mint paused during the window.

© 2026 Adevar Labs. All rights reserved.

5. Enhancement Opportunities

E01: Aligning the migration guide with update_token_metadata will make clear that custom metadata keys cannot be repaired after adoption

LOCATION

issuance-v2/3ac60b77/svm/docs/005-migration.md:L169-L172

RELEVANT SNIPPET

>_ 005-migration.md MARKDOWN

```
167: the metadata forever, so a name spoofed during the window could never be
168: repaired (`MetadataAuthorityRevoked`). Token-2022 can reinstate neither, so
169: both `adopt_stablecoin` and `complete_adoption` refuse them. Metadata fields
170: rewritten during the window remain possible: with the update authority at the
171: PDA they are repairable through `update_token_metadata`, but eyeball the
172: metadata before you run `complete_adoption`.
173: Completion also resets the Confidential Transfer policy to the registered
174: baseline: manual approval, no auditor. The outgoing key could have poisoned
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/metadata/mod.rs:L18-L20
- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/metadata/update_token_metadata.rs:L173-L182

DESCRIPTION

005-migration.md says that metadata fields rewritten during the adoption window are repairable through update_token_metadata once the update authority is at the PDA. That holds for the name, symbol and URI only. update_token_metadata sends one UpdateField CPI for each of those three fields, and no handler sends UpdateField with a custom key or RemoveKey.

Token-2022 metadata also carries additional_metadata, a list of custom key/value pairs that Token-2022 also calls fields. The outgoing authority can add or change them during the window, and V1 can leave some behind from before adoption. Once complete_adoption puts the update authority at the PDA, nothing can change or remove them. No V1 or V2 token writes custom keys today, so current mints are not affected, but an operator following the guide could skip checking custom keys before completion, expecting to fix them afterwards.

POTENTIAL BENEFIT

Operators check custom keys before complete_adoption, while they can still be changed.

RECOMMENDATION

State in 005-migration.md that only the name, symbol and URI can be repaired after adoption, and that custom keys need to be checked before complete_adoption. Alternatively, let update_token_metadata update and remove custom keys.

© 2026 Adevar Labs. All rights reserved.

FIX REVIEW NOTES

Fixed according to the recommendation: the migration guide now says that only the name, symbol and URI can be repaired after adoption, and that custom keys must be checked before **complete_adoption**.

© 2026 Adevar Labs. All rights reserved.

E02: Funding Token-2022's padded length will keep write_token_metadata from failing at the 355-byte Multisig boundary

LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/metadata/write_token_metadata.rs: L145-L147

RELEVANT SNIPPET

>_ write_token_metadata.rs RUST

```rust
let uri_len = check_null_padded(&uri)?;

let grown_len =
    ctx.mint.data_len() + TLV_HEADER_LEN + metadata_value_len(name_len, symbol_len, uri_len);
let required = Rent::get()?.try_minimum_balance(grown_len)?;
let top_up = required.saturating_sub(ctx.mint.lamports());
if top_up > 0 {
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/metadata/update_token_metadata.rs:L159-L162

DESCRIPTION

write_token_metadata tops up the mint's rent for data_len + TLV header + metadata value. Token-2022 does not always allocate that length. When the result would be exactly 355 bytes, the size of a legacy Multisig account, it adds 2 bytes of padding so the two account types cannot be confused, and allocates 357. The top-up then falls 2 bytes short (13,920 lamports at the default rent rate), the runtime rejects the mint as no longer rent-exempt, and the whole instruction rolls back. Sending the same call again fails the same way.

It takes one exact length. An adopted mint whose only extension is a Metadata Pointer occupies 234 bytes, and a metadata value of 117 bytes (an 8-byte name, a 4-byte symbol and a 25-byte URI) brings it to 355. update_token_metadata funds a looser bound, which usually covers the padding, but it can hit the same boundary when its allowance is under 2 bytes.

POTENTIAL BENEFIT

Metadata writes succeed at every length, without the admin over-funding the mint or changing the metadata to step around the boundary.

RECOMMENDATION

Size the top-up with Token-2022's own calculation, try_get_new_account_len_for_variable_len_extension, and use the same length in update_token_metadata.

FIX REVIEW NOTES

Fixed: the metadata write and update paths now fund the 2 bytes of padding Token-2022 adds at 355 bytes, so they no longer fail the rent check.

© 2026 Adevar Labs. All rights reserved.

E03: Letting the admin revoke keys and cancel mints while paused will improve pause_gateway

LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/gate.rs:L37

RELEVANT SNIPPET

>_ gate.rs — RUST

```rust
35:  let stablecoin_bump = verify_stablecoin(stablecoin, mint)?;
36:  let deployment = Deployment::verify(issuer_config)?;
37:  deployment.require_live()?;
38:
39:  let admin = deployment.admin;
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/gate.rs:L68

DESCRIPTION

The spec names **pause_gateway** as the response to "a compromised operational key" (spec). While the gateway is paused, **Gate::require** and **Gate::require_admin** call **require_live**, which reverts with **IssuerPaused**. So the admin cannot call **revoke_roles**, **close_role_grant**, **cancel_mint** or **trip_breaker** during the pause.

>_ 000-spec.md — MARKDOWN

```
793: including callers who bypass our program. The breaker and `pause_gateway`
794: are safety actions against ourselves: they stop only our own instructions, for
795: a compromised operational key or a reserve mismatch. The breaker exists
796: because pausing punishes holders; tripping it stops supply growth while
797: holders keep transacting, the right shape for most issuance incidents.
```

To remove a stolen **MintProposer** or **MintExecutor** key, the admin must first call **resume_gateway**. If the admin sends the resume and the clean-up as two transactions, the stolen key can call **execute_mint** between them on any proposal that matured during the pause. The admin can close this gap only by putting resume and clean-up in one transaction, and no doc or runbook says to do this.

The EVM gateway already allows these steps while paused. **cancelMint** and **pauseMint** carry no **whenNotPaused** modifier, and OpenZeppelin's **revokeRole** is not pausable. So on EVM the issuer can cancel proposals, stop minting and revoke a key without unpausing.

POTENTIAL BENEFIT

The admin removes a stolen key and its queued mints during the pause, so **resume_gateway** no longer reopens minting to that key.

RECOMMENDATION

Let the admin call **revoke_roles**, **close_role_grant**, **cancel_mint** and **trip_breaker** while the gateway is paused, as the EVM gateway already does. Keep every other instruction, and every delegated key, blocked during a pause. Then list these four instructions as **pause_gateway** exceptions in the spec.

© 2026 Adevar Labs. All rights reserved.

Alternatively, the admin can send **resume_gateway** and the clean-up instructions in one transaction, so no other transaction lands between them. If M0 keeps the current code, the spec should describe this procedure.

FIX REVIEW NOTES

Fixed according to the recommendation: while the gateway is paused, the admin can now revoke roles and cancel mints.

© 2026 Adevar Labs. All rights reserved.

E04: The CLI launches every stablecoin with an unbounded mint rate limit

LOCATION

issuance-v2/3ac60b77/svm/cli/src/cli.rs:L161-L170

RELEVANT SNIPPET

>_ cli.rs RUST

```rust
pub currency: String,
/// Seconds between a mint proposal and execution
#[arg(long, default_value_t = 0)]
pub mint_delay: i64,
/// Mint volume available at any instant, in base units; u64::MAX = effectively unlimited
#[arg(long, default_value_t = u64::MAX)]
pub rate_limit_capacity: u64,
/// Rate spent mint capacity replenishes at, in base units per second; 0 = no refill
#[arg(long, default_value_t = 0)]
pub rate_limit_refill_per_second: u64,
/// Annual earner rate in basis points (max 10000); 0 = streaming off
#[arg(long, default_value_t = 0)]
pub earner_rate_bps: u16,
/// Where Revenue Direct rewards are paid; defaults to the signer
```

DESCRIPTION

igw stablecoin register, **launch** and **adopt** default **rate_limit_capacity** to **u64::MAX** and **mint_delay** to 0, with no refill, and the bucket is born full. **docs/000-spec.md** says that at **earner_rate_bps = 0**, also the default, the bucket "becomes the only on-chain bound on payouts".

The Rust SDK ships the same values behind **#[cfg(feature = "test-utils")]**, headed "Test-only. Production callers name every field". The documented launch one-liner passes none of these flags, and **igw doctor** never loads a **Stablecoin** account.

The CLI is outside the audited scope. We note this because the documented launch command relies on these defaults to configure the on-chain rate limit.

POTENTIAL BENEFIT

An operator following the documented launch command gets a rate limit that binds, instead of silently opting out of the control the specification names as the only bound on payouts.

RECOMMENDATION

Make **--rate-limit-capacity** and **--mint-delay** required on **register**, **launch** and **adopt**, or keep the defaults and refuse **u64::MAX** without an explicit acknowledgement flag. Add a **doctor** check that loads the **Stablecoin** account and reports a capacity of **u64::MAX**.

DEVELOPER RESPONSE

Acknowledged. The CLI still defaults to an unbounded rate limit and no mint delay.

© 2026 Adevar Labs. All rights reserved.

E05: Optional delay on admin rotation will give the issuer time to cancel a hostile rotation

LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/admin/propose_admin.rs:L57-L81

RELEVANT SNIPPET

>_ propose_admin.rs RUST

```rust
#[inline(always)]
pub fn process_propose_admin(accounts: &mut [AccountView], _data: &[u8]) -> ProgramResult {
    let ctx = ProposeAdminAccounts::context(accounts)?;

    let deployment = Deployment::verify(ctx.issuer_config)?;
    if ctx.authority.address() != &deployment.admin {
        return Err(GatewayError::Unauthorized.into());
    }

    let proposed = *ctx.new_admin.address();

    IssuerConfig::load_mut_checked(&mut ctx.issuer_config.try_borrow_mut()?)?.pending_admin =
        proposed;

    emit_no_mint(
        ctx.issuer_config,
        ctx.event_authority,
        ctx.program,
        &AdminProposed {
            admin: deployment.admin,
            proposed,
        },
    )?;

    Ok(())
}
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/admin/accept_admin.rs:L54-L71
- issuance-v2/3ac60b77/svm/crates/state/src/issuer_config.rs:L24-L29

DESCRIPTION

IssuerConfig.admin passes every gate in the issuer gateway without a role grant. It can mint through the proposal flow, seize, pause, grant and revoke roles, and eject mint authorities. The admin changes in two steps. **propose_admin** stores the nominee in **pending_admin**, and **accept_admin** replaces **admin** as soon as the nominee signs. **IssuerConfig** stores no timestamp, and **accept_admin** checks only the signer, so the nominee can accept in the same transaction as the proposal.

The EVM version lets each deployment set a wait between the start and the end of an admin transfer, so the new admin can take control only after that wait. The SVM gateway gives the issuer no such option. The SVM spec maps both EVM admin roots to one **IssuerConfig.admin** but does not mention

© 2026 Adevar Labs. All rights reserved.

that the delay is dropped. It describes the two steps only as typo protection ("the nominee must sign, so a typo cannot brick anything").

The repo docs recommend Squads multisigs for the admin (001-architecture.md), and a Squads **time_lock** could add a delay off-chain. The program does not check for this, and the spec does not mention it. So if an attacker compromises the admin key, or a threshold of its signers, they can move full operational control to their own key in one transaction. The issuer and its monitoring then get no time to react.

>_ 001-architecture.md MARKDOWN

```
122: setter, because a key nobody controls can never sign the accept, which is what
123: makes a typo survivable. Rotation runs even while the gateway is paused
124: (incident recovery). Operationally both the admin and the BPF upgrade authority
125: should be Squads multisigs (2-of-3 / 3-of-5, managed in the Squads UI, outside
126: this repo); the upgrade authority remains the ultimate recovery path.
127:
```

A delay would not stop the attacker from acting as admin with the stolen key during the wait. It would give the issuer and its monitoring time to see the nomination and respond, for example by preparing an upgrade, before the lockout takes effect.

POTENTIAL BENEFIT

A delay on admin rotation gives the issuer time to see a hostile nomination and cancel it before the new admin takes control.

RECOMMENDATION

Add an optional **admin_delay** to **IssuerConfig**. **propose_admin** would record when the nomination becomes valid, and **accept_admin** would refuse it before then using the **Clock** sysvar. A delay of **0** keeps the current instant recovery behavior for issuers that prefer it.

If M0 keeps the current design, the spec should state that the program enforces no delay and name a multisig time lock, such as the Squads **time_lock** setting, as the expected control for the admin and the upgrade authority.

In both cases, the spec should require the admin and the upgrade authority to sit in different multisigs, so the upgrade authority stays a working recovery path.

FIX REVIEW NOTES

Fixed: admin rotation now waits a fixed delay set in **initialize_issuer_config**, up to 30 days.

© 2026 Adevar Labs. All rights reserved.

E06: Enforcing distinct proposer and executor keys in execute_mint will improve issuance safety

LOCATION

issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/issuance/execute_mint.rs:L158-L170

RELEVANT SNIPPET

>_ execute_mint.rs RUST

```rust
}

// A revoke kills the queue and a regrant does not revive it.
if proposal.proposer != gate.admin {
    let since_slot = verify_grant(
        ctx.proposer_grant,
        &mint_key,
        &proposal.proposer,
        Role::MintProposer,
        GatewayError::MintProposerRevoked,
    )?;
    if since_slot >= proposal.proposed_slot {
        return Err(GatewayError::MintProposerRevoked.into());
    }
}

if ctx.destination.address() != &proposal.destination {
```

OTHER LOCATIONS

- issuance-v2/3ac60b77/svm/programs/issuer-gateway/src/actions/roles/grant_roles.rs:L149-L161

DESCRIPTION

The issuance flow has two steps. A **MintProposer** calls **propose_mint**, and after the delay a **MintExecutor** calls **execute_mint**. The module comment of **execute_mint** says "The executor role is separate from **MintProposer** so one key cannot both propose and perform an issuance". The spec repeats this in its role table ("never executes it", "pairs with MintProposer for two-key issuance") and in its role separation note, which says "a single stolen key can neither start nor finish an issuance alone".

>_ execute_mint.rs RUST

```rust
//! is debited at execution, so queued proposals reserve nothing.
//!
//! Requires [`Role::MintExecutor`] or the issuer admin. The executor role is
//! separate from `MintProposer` so one key cannot both propose and perform an
//! issuance. The proposer must still hold `MintProposer`, granted before the
//! proposal (or be the admin).
//!
//! Accounts:
```

© 2026 Adevar Labs. All rights reserved.

>_ 000-spec.md MARKDOWN

```
965: | Role                  | May call                                  | The duty
                                                                        |
966: | --------------------- | ----------------------------------------- | -----------------
     ------------------------------------------------------------------ |
967: | MintProposer          | `propose_mint`, `cancel_mint`             | Starts and withdra
     ws issuance; never executes it                                     |
968: | MintExecutor          | `execute_mint`                            | Completes matured
     proposals; pairs with MintProposer for two-key issuance            |
969: | Burner                | `burn`                                    | Retires supply on
     redemption (the holder co-signs)                                   |
970: | FreezeManager         | `freeze_account`, `thaw_account`          | Per-holder complia
     nce actions                                                        |
```

>_ 000-spec.md MARKDOWN

```
985: admin-only knob is gated as the stricter one.
986:
987: The separations that matter operationally: proposing and executing a mint are
988: two roles, so with a nonzero mint delay a single stolen key can neither start
989: nor finish an issuance alone. The breaker is safe to hand to automated
990: monitoring precisely because tripping it never harms holders. Seizure is the
991: narrowest grant of all: give ForcedTransferManager to nobody and only the
```

The program does not enforce this:

- **grant_roles** inserts each requested role into the grant with no exclusion rule, so the admin can give one key both **MintProposer** and **MintExecutor**.
- **execute_mint** checks that the signer holds **MintExecutor** and that the proposer still holds **MintProposer**, but it never compares **proposal.proposer** with the signer.

A key that holds both roles can therefore propose a mint, wait for the delay, and execute it alone. If the issuer gives both roles to one hot key, a thief who steals that key can mint up to the rate limit.
The spec says this cannot happen. Whether issuance needs two keys therefore depends on how the issuer gives out roles, and the docs do not say so.

POTENTIAL BENEFIT

The program enforces the two-key issuance that the spec and the code comment describe, with one 32-byte compare and no extra account.

RECOMMENDATION

In **execute_mint**, refuse execution by the proposer's own key unless that key is the admin. Put the check after the proposer grant check:

© 2026 Adevar Labs. All rights reserved.

DIFF

```
if since_slot >= proposal.proposed_slot {
            return Err(GatewayError::MintProposerRevoked.into());
        }
    }
+
+    // Two-key issuance: the key that proposed may not execute.
+    if proposal.proposer != gate.admin && ctx.authority.address() == &proposal.proposer {
+        return Err(GatewayError::MintSelfExecution.into())
+    }
```

The check compares keys, not people, so one operator who controls two keys or two Squads vaults can still issue alone. If the team prefers not to change the code, the spec text and the code comment should say that the separation depends on the issuer giving the two roles to different keys.

>_ 000-spec.md MARKDOWN

```
985: admin-only knob is gated as the stricter one.
986:
987: The separations that matter operationally: proposing and executing a mint are
988: two roles, so with a nonzero mint delay a single stolen key can neither start
989: nor finish an issuance alone. The breaker is safe to hand to automated
990: monitoring precisely because tripping it never harms holders. Seizure is the
991: narrowest grant of all: give ForcedTransferManager to nobody and only the
```

>_ execute_mint.rs RUST

```
  4: //! is debited at execution, so queued proposals reserve nothing.
  5: //!
  6: //! Requires [`Role::MintExecutor`] or the issuer admin. The executor role is
  7: //! separate from `MintProposer` so one key cannot both propose and perform an
  8: //! issuance. The proposer must still hold `MintProposer`, granted before the
  9: //! proposal (or be the admin).
 10: //!
 11: //! Accounts:
```

FIX REVIEW NOTES

Fixed according to the recommendation: **execute_mint** now refuses a non-admin signer that proposed the same mint.

© 2026 Adevar Labs. All rights reserved.

About Us

About Adevar Labs

Adevar Labs is a boutique blockchain security firm specializing in Web3 audits.

The firm was built by a mix of experienced professionals from traditional enterprise security and crypto natives who have contributed to some of the most critical projects in blockchain infrastructure.

Our team's background spans companies like Bitdefender, Asymmetric Research, Quantstamp, Chainproof, and Juicebox, and includes experience securing smart contracts, bridges, and L1 and L2 protocols across ecosystems like Solana, Ethereum, Polkadot, Cosmos, and MultiversX.

With over 100 audits completed and a portfolio that includes custom fuzzers, exploit modeling, and runtime testing frameworks, Adevar Labs brings both depth and precision to every engagement.

Our auditors have discovered critical vulnerabilities, built high-impact tooling, and placed in top positions in premier audit competitions including Code4rena and Sherlock.

Team members hold distinctions such as PhDs in software protection, key roles at Fortune 500 companies, and leadership of flagship conferences like ETH Bucharest.

With team members whose publications have earned over 1,100 academic citations, and with the trust of projects securing over $500M in on-chain value, Adevar Labs blends elite technical rigor with real-world security impact.

We also collaborate with some of the best independent security researchers in the Web3 space.

Projects may optionally request specific contributors to be part of their audit.

Audit Methodology

Our audit methodology is designed to provide thorough security assessments of Solana programs. We focus explicitly on rigorous manual analysis, detailed threat modeling, and careful validation of implemented fixes to ensure your programs operate securely and as intended.

1. Program Context and Architecture Analysis

Our auditors begin by deeply examining your Solana program's documentation, intended functionality, and account design. We meticulously map out program interactions, instruction processing flows, and state management logic. Special attention is given to understanding how your program interfaces with critical Solana system programs, such as the SPL Token Program, Stake Program, and System Program. Finally, we examine external integrations with other Solana projects and verify that inputs and return values are handled properly.

© 2026 Adevar Labs. All rights reserved.

2. Threat Modeling

We conduct targeted threat modeling tailored specifically for Solana's execution environment. We carefully define attacker capabilities and identify potential vulnerabilities that may arise from Solana-specific issues, including but not limited to:

- Unauthorized account data manipulation
- Improper ownership or signer verification
- Misuse of Program-Derived Addresses (PDAs)
- Incorrect use of Cross-Program Invocations (CPI)
- Failure to adequately handle account privileges or account states
- Risks stemming from rent-exemption and account initialization logic

3. In-depth Manual Security Review

Our experienced auditors perform an extensive manual security review of your Rust-based Solana programs. This involves a comprehensive line-by-line inspection of source code, focusing on common Solana vulnerabilities including, but not limited to:

- Missing or insufficient ownership checks
- Inadequate signer checks
- Incorrect handling of CPI calls and invocation privileges
- Arithmetic and integer overflow or underflow errors
- Unsafe deserialization and serialization of account data structures
- Improper token transfers and SPL-token logic issues
- Logic flaws in financial operations or state transitions
- Edge-case handling in instruction input validation
- Potential denial-of-service vectors related to transaction execution and account handling

During this stage, we clearly document any discovered vulnerabilities, including detailed descriptions, precise severity ratings, and recommendations for secure implementation. In situations where it is not clear how the vulnerability might be exploited, we may also include a detailed proof-of-concept exploit with code snippets and instructions on how the attack could be performed.

4. Detailed Fix Review and Validation

After the initial audit and your team's subsequent remediation efforts, we perform a comprehensive fix review to ensure the vulnerabilities identified have been effectively resolved.

© 2026 Adevar Labs. All rights reserved.

We verify each fix individually, confirming that:

- Corrections effectively eliminate the security risks
- Changes do not inadvertently introduce new vulnerabilities or regressions
- The fixes align closely with Solana best practices and secure coding guidelines

Our detailed validation ensures that security improvements are robust, complete, and aligned with best practices specific to the Solana development ecosystem.

Confidentiality Notice

This report, including its content, data, and underlying methodologies, is subject to the confidentiality and feedback provisions in your agreement with Adevar Labs. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Adevar Labs.

Legal Disclaimer

The review and this report are provided by Adevar Labs on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Adevar Labs disclaims all warranties, express or implied, in connection with this report, its content, and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

You agree that access to and/or use of the report and other results of the review, including any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time of the engagement.

You acknowledge that blockchain technology remains under development and is subject to unknown risks and flaws. Adevar Labs does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open-source or third-party software, code, libraries, materials, or information accessible through the report. As with the purchase or use of a product or service in any environment, you should use your best judgment and exercise caution where appropriate.

© 2026 Adevar Labs. All rights reserved.