Securitize Solana Redemption Audit Report

Prepared by Cyfrin
Version 2.0

Lead Auditors
Farouk
Naman

August 29, 2025

Contents

1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

2 Disclaimer

The Cyfrin team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Securitize Off-Ramp is a Solana program that converts an asset token into a liquidity token using a NAV rate, with identity checks, country restrictions, fees, and pause control. It is composed of:

- **State**
  - `OffRampCounter` - global counter for unique off-ramp ids.
  - `OffRampState` - per instance config: admin, asset mint, asset policy (Recipient or Burn), liquidity mint, NAV provider program id, fee manager, pause flag, restricted countries bitmap, and a dedicated withdraw authority. A PDA authority controls the program vault.
- **External dependencies**
  - **Identity stack** - `IdentityRegistryAccount`, `IdentityAccount`, `WalletIdentity` used to bind wallets to identities and read a country code.
  - **NAV provider program** - CPI interface that returns a rate for the asset mint.
  - **SPL Token and Token-2022** - asset and liquidity mints may be legacy or 2022, with optional extensions and transfer hooks.
- **Core instructions**
  - **Initialize** - creates `OffRampState`, increments the counter, and creates the liquidity vault ATA owned by the off-ramp authority PDA. Validates fee manager and asset policy.
  - **Redeem** - checks pause, identity and country, fetches NAV, computes output and fee, enforces min-out, moves liquidity to user and fee collector, and burns or deposits the asset per policy.
  - **Withdraw liquidity** - withdraw authority transfers liquidity tokens from the vault to its ATA.
  - **Admin updates** - pause or unpause, change admin, change withdraw authority, update NAV provider, fee manager, asset policy, and country restrictions.
  - **Quote** - returns the computed liquidity amount for a given asset amount, optionally before fees.

5 Audit Scope

Cyfrin conducted an audit of the Securitize Solana Redemption program based on the code present in the repository commit hash 36051eb. The following files construct the scope of the audit:

```
programs
 securitize-off-ramp
     src
         constants.rs
         errors.rs
         events.rs
         instructions
            calculate_liquidity_token_amount.rs
            change_admin.rs
            change_liquidity_withdraw_authority.rs
            initialize.rs
            mod.rs
            pause.rs
            redeem.rs
            unpause.rs
            update_asset_policy.rs
            update_countries_restriction.rs
            update_fee_manager.rs
            update_nav_provider.rs
            withdraw_liquidity.rs
         lib.rs
         states
            mod.rs
            off_ramp_counter.rs
            off_ramp_state.rs
         utils
             fee_manager
                fee_manager_core.rs
                mod.rs
                mpbs_fee_manager.rs
             get_authority_signer.rs
             get_rate.rs
             get_transfer_hook_program_id.rs
             mod.rs
             token_calculator.rs
             transfer_maybe_hook.rs
```

6 Executive Summary

Over the course of 5 days, the Cyfrin team conducted an audit on the Securitize Solana Redemption smart contracts provided by Securitize. In this period, a total of 7 issues were found.

Securitize Off-Ramp binds compliant identity to redemption and converts assets to a separate liquidity token at a NAV rate. The trust model assumes an admin that can configure providers, fees, policy, and restrictions, plus a withdraw authority that can move liquidity from the vault. The program relies on three runtime checks: correct identity to wallet binding for country enforcement, strict routing of redeemed assets to the configured recipient when not burning, and correct pricing based on a provider supplied rate. Because pricing and settlement depend on external CPIs and Token-2022 behavior, tight payload vs state checks and careful arithmetic are the backbone of safety.

Our review focused on authorization boundaries, PDA derivations and vault control, NAV CPI shaping, identity and country enforcement, fee handling, Token-2022 edge cases, arithmetic safety, and DoS surfaces around initialization and account lists.

High risk issues included a missing equality check that lets callers redirect asset deposits when the policy is Recipient, and incomplete wallet to identity binding that allows bypassing country restrictions by mixing a valid WalletIdentity with an unrelated IdentityAccount.

Medium risk issues included an initialization DoS by pre-creating the PDA owned vault ATA, while lower severity items covered silent truncation on u128 to u64 cast in the calculator, fee collector field ambiguity wallet vs token account, missing liquidity vault balance validation, and missing two step ownership and authority transfer validation.

Summary

Project Name	Securitize Solana Redemption
Repository	bc-solana-redemption-sc
Commit	36051ebd2739...
Audit Timeline	Aug 19st - Aug 25th
Methods	Manual Review

Issues Found

Critical Risk	0
High Risk	1
Medium Risk	2
Low Risk	1
Informational	3
Gas Optimizations	0
Total Issues	7

Summary of Findings

[H-1] Broken identity-to-wallet binding in redeem allows country restriction bypass	Resolved
[M-1] DoS in initialize via pre-created ATA for `off_ramp_authority`	Resolved
[M-2] Permissionless OffRampState initialization under official program ID enables spoofed “official” instances	Resolved
[L-1] Silent truncation on u128 to u64 cast in liquidity token amount calculator	Resolved
[I-1] Fee collector field ambiguity wallet vs token account	Resolved
[I-2] Missing Two Step Ownership And Authority Transfer Validation	Acknowledged
[I-3] Missing Liquidity Vault Balance Validation	Resolved

7 Findings

7.1 High Risk

7.1.1 Broken identity-to-wallet binding in redeem allows country restriction bypass

Description: The `redeem` instruction accepts three identity objects that should all refer to the same person and wallet: `IdentityRegistryAccount`, `IdentityAccount`, and `WalletIdentity`. The account constraints only ensure:

- `IdentityRegistryAccount` matches the `asset_mint`.
- `IdentityAccount` belongs to that registry.
- `WalletIdentity` is the PDA for `(redeemer, asset_mint)`.

There is no on-chain assertion that the provided `IdentityAccount` is the one linked to the `WalletIdentity` and the `redeemer`. As a result, the caller can mix a valid `WalletIdentity` for their wallet with someone else's `IdentityAccount` that has a permitted country, then pass country checks.

```
/// Identity registry for asset mint compliance
#[account(
    has_one = asset_mint,
    seeds = [asset_mint.key().as_ref()],
    seeds::program = ::identity_registry::ID,
    bump = identity_registry.bump,
)]
pub identity_registry: Box<Account<'info, IdentityRegistryAccount>>,

/// User's identity account with country information
#[account(
    has_one = identity_registry,
    seeds = [identity_registry.key().as_ref(), identity_account.owner.as_ref()],
    seeds::program = ::identity_registry::ID,
    bump
)]
pub identity_account: Box<Account<'info, IdentityAccount>>,

/// Links wallet to identity account
#[account(
    seeds = [redeemer.key().as_ref(), asset_mint.key().as_ref()],
    seeds::program = ::identity_registry::ID,
    bump,
)]
pub wallet_identity: Box<Account<'info, WalletIdentity>>,
```

Impact: Country restriction bypass. A wallet from a restricted country can redeem by supplying a different user's `IdentityAccount` that reports an allowed country.

Recommended Mitigation: Consider requiring the association between the provided `IdentityAccount` and the `WalletIdentity` .

Securitize: Fixed in 78ad18d.

Cyfrin: Verified.

7.2 Medium Risk

7.2.1 DoS in initialize via pre-created ATA for `off_ramp_authority`

Description: The `initialize` instruction creates the liquidity vault as an **associated token account** for the program PDA `off_ramp_authority`:

```
#[account(
    init,
    payer = admin,
    associated_token::mint = liquidity_token_mint,
    associated_token::authority = off_ramp_authority,
    associated_token::token_program = liquidity_token_program,
)]
pub liquidity_token_vault: Box<InterfaceAccount<'info, TokenAccount>>;
```

Associated token accounts are globally derivable and can be created by **anyone** for any owner without the owner's signature. Because both `off_ramp_state` and `off_ramp_authority` PDAs are deterministically derived from public seeds (counter and state key), an attacker can precompute the vault ATA and create it first. When `initialize` later runs with `init`, Anchor will fail with "already in use," reverting the whole transaction.

Impact: Hard denial of service on program initialization for a given `(off_ramp_state, liquidity_token_mint)`. An attacker can repeatedly grief by precreating the ATA for each anticipated off_ramp ID, blocking deployment unless the admin changes parameters. This is cheap for the attacker and can be repeated.

Recommended Mitigation: Switch to `init_if_needed` to make initialization idempotent and immune to precreation:

```
#[account(
    init_if_needed,
    payer = admin,
    associated_token::mint = liquidity_token_mint,
    associated_token::authority = off_ramp_authority,
    associated_token::token_program = liquidity_token_program,
)]
pub liquidity_token_vault: Box<InterfaceAccount<'info, TokenAccount>>;
```

This accepts a pre-existing correct ATA and proceeds.

Securitize: Fixed in 1a8a098.

Cyfrin: Verified.

7.2.2 Permissionless OffRampState initialization under official program ID enables spoofed "official" instances

Description: The `initialize` instruction lets any signer create a new `OffRampState` and become its `admin`. The global `OffRampCounter` is `init_if_needed` and unguarded, and the new state PDA is derived from `[OFF_RAMP_STATE_SEED, off_ramp_counter.counter.to_le_bytes()]`. There is no allowlist or registry check tying the initializer to an official Securitize operator. This means anyone can spin up an OffRamp instance under the same Program ID and emit an `Initialized` event, which can be marketed as if it were an official, Securitize backed off ramp.

```
/// Global counter for generating unique off-ramp IDs
#[account(
    init_if_needed,
    payer = admin,
    space = 8 + OffRampCounter::INIT_SPACE,
    seeds = [OFF_RAMP_COUNTER_SEED],
    bump,
)]
pub off_ramp_counter: Box<Account<'info, OffRampCounter>>,
/// Off-ramp state containing configuration and settings
#[account(
    init,
    payer = admin,
    space = 8 + OffRampState::INIT_SPACE,
    seeds = [OFF_RAMP_STATE_SEED, off_ramp_counter.counter.to_le_bytes().as_ref()],
    bump,
)]
pub off_ramp_state: Box<Account<'info, OffRampState>>,
```

Impact: A third party can deploy a look alike instance with arbitrary fees, NAV provider, and recipient policy, then present it as "the Securitize off ramp" because it is hosted under the same Program ID.

Recommended Mitigation: Add a `GlobalConfig` PDA that stores an `authorized_initializer` or allowlist. In `initialize`, require the `admin` signer to be on that list.

Securitize: Fixed in 30362cf.

Cyfrin: Verified.

7.3 Low Risk

7.3.1 Silent truncation on u128 to u64 cast in liquidity token amount calculator

Description: The `utils::token_calculator::calculate_liquidity_token_amount` function computes the output in `u128` and returns it with a plain `as u64` cast:

```
let result = /* u128 math */;
Ok(result as u64)
```

If `result` fits in `u128` but exceeds `u64::MAX`, the cast truncates the high bits without error. The function then reports a much smaller number than intended. Both the quote path and `redeem` use this helper, so the truncation can silently underpay.

Impact: Silent underpayment and wrong accounting.

Recommended Mitigation: Replace the lossy cast with a checked conversion:

```
use core::convert::TryFrom;

let result_u64 = u64::try_from(result)
    .map_err(|_| SecuritizeOffRampError::Overflow)?;
Ok(result_u64)
```

Securitize: Fixed in 7172884.

Cyfrin: Verified.

7.4 Informational

7.4.1 Fee collector field ambiguity wallet vs token account

Description: `FeeManager::collector` is a `Pubkey` named like a wallet, but every place that uses it expects a **token account address** for the liquidity mint. In `initialize` and `update_fee_manager` you require `fee_collector_ta.address == fee_manager.fee_collector()` and enforce `token::mint = liquidity_token_mint`.

If an integrator sets `collector` to a wallet pubkey instead of the token account pubkey, the instruction fails or fees route incorrectly across environments.

```rust
/// Fee manager enum for fee strategies
pub enum FeeManager {
    /// Mbps-based fee manager
    MbpsFeeManager(MbpsFeeManager),
}
```

```rust
/// Mbps-based fee manager (basis points)
pub struct MbpsFeeManager {
    /// Fee numerator (bps)
    pub numerator: u32,
    /// Fee collector address
    pub collector: Pubkey,
}
```

Recommended Mitigation: Rename the field to `collector_token_account` to reflect intent.

Securitize: Fixed in ab7f4d2.

Cyfrin: Verified.

7.4.2 Missing Two Step Ownership And Authority Transfer Validation

Description: Both `change_admin_handler` and `change_liquidity_withdraw_authority_handler` directly reassign critical control fields (`off_ramp_state.admin` and `off_ramp_state.liquidity_withdraw_authority`) in a single transaction without requiring confirmation from the proposed new key. This one-step transfer model increases the risk of accidental misconfiguration or malicious key injection. Additionally, neither function validates against assignment of the default zero address (`Pubkey::default()`), which could permanently lock the system by assigning an unusable authority.

Impact: If a privileged signer mistakenly or maliciously sets the new authority to the default address, administrative or liquidity withdrawal rights could be irreversibly lost. Furthermore, the absence of a two-step acceptance process enables unilateral transfers without the consent of the intended new owner, reducing operational safety and creating potential governance disputes.

Recommended Mitigation:

- Introduce a two-step transfer process where the current owner proposes a new authority, and the proposed authority must explicitly accept before finalization.
- Also, enforce non-default key validation to prevent assignment to the zero address.

Securitize: Acknowledged.

7.4.3 Missing Liquidity Vault Balance Validation

Description: In `withdraw_liquidity_handler`, liquidity tokens are transferred from `liquidity_token_vault` to the withdrawer's associated token account. While the instruction checks that the withdrawal amount is greater than zero and that the system is not paused, it does not validate whether the vault actually holds at least `amount` tokens before attempting the transfer. This omission may allow withdrawal attempts that exceed the available vault balance, leading to failed transactions or unintended program behavior depending on the token program implementation.

Impact: If the vault contains fewer tokens than requested, the transfer may fail at runtime, causing unnecessary transaction failures.

Recommended Mitigation: Add a balance check to ensure that `liquidity_token_vault.amount >= amount` before executing the transfer.

```
+ let liquidity_vault = &ctx.accounts.liquidity_token_vault;
+    require_gte!(
+      liquidity_vault.amount,
+      amount,
+    SecuritizeOffRampError::InsufficientLiquidity
+);
```

Securitize: Fixed in 3163cd9.

Cyfrin: Verified.