Solana DSToken Whitelist Audit Report

Prepared by Cyfrin

Version 2.0

Lead Auditors

DadeKuma

Naman

November 7, 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

The DSToken Whitelist manages the investor identity verification and wallet authorization process by orchestrating cross-program invocations to the RWA RBAC system. Investor wallets are onboarded through a coordinated whitelisting flow that establishes identity verification and access control relationships.

Upon successful whitelisting, this program validates wallet ownership and authority before executing up to three sequential operations: optionally registering the investor in the identity metadata registry with compliance tracking, optionally adding compliance levels to meet policy engine requirements, and finally attaching the wallet to the verified identity account. The program accepts flexible CPI data parameters to accommodate both simple wallet associations and comprehensive investor onboarding with compliance tier assignments.

The program uses common administrative framework featuring role-based access control with a single admin authority, atomic admin transfer capabilities with duplicate-prevention checks, and synchronized pause/unpause mechanisms. The architecture prioritizes operational flexibility and emergency response capabilities while maintaining strict separation of concerns between identity verification and token custody layers.

5 Audit Scope

The audit scope was limited to:

```
programs/dstoken-whitelist/src/instructions/admin/change_admin.rs
programs/dstoken-whitelist/src/instructions/admin/initialize.rs
programs/dstoken-whitelist/src/instructions/admin/mod.rs
programs/dstoken-whitelist/src/instructions/admin/pause.rs
programs/dstoken-whitelist/src/instructions/admin/unpause.rs
programs/dstoken-whitelist/src/instructions/mod.rs
programs/dstoken-whitelist/src/instructions/whitelist.rs
programs/dstoken-whitelist/src/states/dstoken_whitelist_state.rs
programs/dstoken-whitelist/src/states/mod.rs
programs/dstoken-whitelist/src/constants.rs
programs/dstoken-whitelist/src/errors.rs
```

```
programs/dstoken-whitelist/src/events.rs
programs/dstoken-whitelist/src/lib.rs
```

6 Executive Summary

Over the course of 2 days, the Cyfrin team conducted an audit on the Solana DSToken Whitelist smart contracts provided by Securitize. In this period, a total of 2 issues were found.

Summary

Project Name	Solana DSToken Whitelist
Repository	bc-solana-whitelist-sc
Commit	3aeb8df7a98b...
Fix Commit	2a83f4e3f518...
Audit Timeline	Nov 6th - Nov 7th, 2025
Methods	Manual Review

Issues Found

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

Summary of Findings

[I-1] Unnecessary use of `emit_cpi!` increases CU cost	Resolved
[I-2] Transaction size limit could be exceeded with non-empty hashes	Acknowledged

7 Findings

7.1 Informational

7.1.1 Unnecessary use of `emit_cpi!` increases CU cost

Description: The `whitelist` instruction uses `emit_cpi!` and the `#[event_cpi]` attribute to emit the `Whitelisted` event.

The `emit_cpi!` macro is designed for programs that are called via CPI and need their events to propagate to calling programs.

However, the `dstoken-whitelist` program is intended to be called directly by end users, not invoked via CPI by other programs.

Impact: Using `emit_cpi!` when unnecessary adds complexity without providing any benefit:

- Increases transaction size and CU cost
- It's currently not possible to directly subscribe to these events
- Requires additional event authority accounts to be passed in the instruction

Recommended Mitigation: Consider replacing `emit_cpi!` with `emit!`, and removing the `#[event_cpi]` attribute.

Securitize: Here is the fix: https://github.com/securitize-io/bc-solana-whitelist-sc/commit/2a83f4e3f518c9d0801b24d900be56240bd7da31

Cyfrin: Verified.

7.1.2 Transaction size limit could be exceeded with non-empty hashes

Description: The `whitelist` instruction accepts variable-length strings (`investor_id`, `collision_hash`, `proof_hash`) that are encoded in CPI data buffers.

Solana enforces a 1,232-byte transaction size limit, which might limit the whitelisting feature in some edge cases.

Impact: The production configuration uses empty strings for `collision_hash` and `proof_hash`, which keeps transaction sizes below the 1,232-byte limit.

However, if non-empty hashes are used in the future, the transaction size could exceed the limit when adding 5 or more levels in a single transaction, causing the transaction to fail.

Proof of Concept: With the configuration using empty hashes, the `whitelist` instruction maintains a safety margin below the 1,232-byte transaction limit:

Levels	Transaction Size	Margin	Status
2	994 bytes	238 bytes	Safe
3	1,016 bytes	216 bytes	Safe
5	1,060 bytes	172 bytes	Safe

For comparison, if non-empty 32-character hashes were used, the transaction would revert with 5 or more levels:

Levels	Transaction Size	Margin	Status
2	1,098 bytes	134 bytes	Safe
3	1,143 bytes	89 bytes	Safe
5	1,233 bytes	-1 byte	Exceeds limit

The development team confirmed that production will use empty strings for `collision_hash` and all `proof_hash` values. The validation `rbac_utils::is_valid_hash` accepts empty strings as valid:

```
pub fn is_valid_hash(hash: &str) -> bool {
     hash.is_ascii() && hash.len() <= 32
}
```

Recommended Mitigation: No mitigation required for the current configuration, but consider documenting this behavior.

If future requirements change to use non-empty hashes with 5 or more levels, the function would need to be split into separate transactions to avoid reverting.

Securitize: Acknowledged.