

January 8th 2026

Echelon Coin (ELON)

Comprehensive Security Assessment

Table of Contents

Disclaimer

1. Introduction

1.1. Executive Summary

1.2. Project Timeline

1.3. Scope

1.4. Overview of Findings

2. Findings

2.1. [I-1] freeze_stores May Abort and Fail to Freeze

2.2. [I-2] register() May Be Misleading Under Modern Coin ↔ FA Behavior

2.3. [I-3] Manager Can Burn From Arbitrary Accounts

Disclaimer

This security assessment represents a time-boxed security review using tooling and manual review methodologies. Our findings reflect our comprehensive evaluation of the materials provided in-scope and are specific to the commit hash referenced in this report.

The scope of this security assessment is strictly limited to the code explicitly specified in the report. External dependencies, integrated third-party services, libraries, and any other code components not explicitly listed in the scope have not been reviewed and are excluded from this assessment.

Any modifications to the reviewed codebase, including but not limited to smart contract upgrades, protocol changes, or external dependency updates will require a new security assessment, as they may introduce considerations not covered in the current review.

In no event shall Plainshift's aggregate liability for all claims, whether in contract or any other theory of liability, exceed the Services Fee paid for this assessment. The client agrees to hold Plainshift harmless against any and all claims for loss, liability, damages, judgments and/or civil charges arising out of exploitation of security vulnerabilities in the contracts reviewed.

By accepting this report, you acknowledge that deployment and implementation decisions rest solely with the client. Any reliance upon the information in this report is at your own discretion and risk. This disclaimer is governed by and construed in accordance with the laws specified in the engagement agreement between Plainshift and the client.

1. Introduction

Plainshift is a full-stack security firm built on the “shift left” security philosophy. We often work with teams early in the product development process to bring security to a greater organizational range than just smart contracts. From the web app, to fuzzing/formal verification, to a team’s operational security, full-stack security can only be achieved by first understanding there is no “scope” to fully protect the users that trust you.

We’re here to meaningfully revolutionize how teams approach security and guide them towards a holistic approach rather than the single sided approach so prevalent today.

Learn more about us at <https://plainshift.io>.

1.1. Executive Summary

Plainshift completed a security review of the ELON coin module from January 7th 2026 to January 8th 2026. The audit identified three informational findings, with no severe issues identified. All issues reported are acknowledged by the project team. The findings primarily arise from edge-case behavior in Aptos’ evolving coin ↔ fungible-asset model and the explicit trust assumptions around administrative controls. These items were classified as informational because they do not present exploitable vulnerabilities in the current design; rather, they impact operational reliability, integration clarity, and documentation of intended privileges.

1.2. Project Timeline

Date	Phase
January 7th, 2026	Kickoff
January 8th, 2026	Audit End
January 8th, 2026	Delivery

1.3. Scope

Repositories	https://github.com/EchelonMarket/echelon-modules
Version	aaf06666a0f64a9e72471dfc17970ee62d0f93b3
Contracts	echelon_coin/sources/echelon_coin.move
Type	Move
Platform	Aptos

1.4. Overview of Findings

Our comprehensive review yielded 0 high, 0 medium, 0 low and 3 informational issues.

Severity/Impact Level	Count
High	0
Medium	0
Low	0
Informational	3



2. Findings

2.1. [I-1] freeze_stores May Abort and Fail to Freeze

Target	echelon_coin/sources/echelon_coin.move
Severity	● Informational
Category	Edge Case

2.1.1. Description

The `freeze_stores()` entrypoint batches freezing for both legacy coin stores and primary fungible stores. It first calls `freeze_coin_stores()` and only then calls `freeze_fungible_stores()` :

```

1  public entry fun freeze_stores(manager: &signer, account_addresses: vector<address>)
    acquires Capabilities {
2      freeze_coin_stores(manager, account_addresses);
3      freeze_fungible_stores(manager, account_addresses)
4  }
```

Inside `freeze_coin_stores()` , each address is processed with `coin::freeze_coin_store()` without checking for a `CoinStore<ELON>` resource. In Aptos' current coin ↔ FA model, many holders may only have a Primary Fungible Store and never publish a `CoinStore<ELON>` . If any address lacks a `CoinStore<ELON>` , the underlying `borrow_global_mut<CoinStore<...>>()` aborts and the entire transaction rolls back, preventing any freeze from being applied to remaining accounts.

2.1.2. Impact

The batch freeze can fail for legitimate holder lists. A single address without a `CoinStore<ELON>` prevents all accounts in the list from being frozen in that transaction, which can block operational responses that depend on reliable freezing.

2.1.3. Recommendation

Make the CoinStore portion best-effort or split the steps so fungible-store freezing can succeed even if a legacy CoinStore is missing. For example, guard the legacy freeze with an existence check:

```
1  vector::for_each(account_addresses, |account_address| {  
2      if (exists<CoinStore<ELON>>(account_address)) {  
3          coin::freeze_coin_store(account_address, &cap.freeze_capability);  
4      }  
5  });
```

2.1.4. Remediation

Acknowledged by the project team.

2.2. [I-2] register() May Be Misleading Under Modern Coin ↔ FA Behavior

Target	echelon_coin/sources/echelon_coin.move
Severity	● Informational
Category	Code Quality

2.2.1. Description

The module exposes `register()` which calls `coin::register<ELON>()` :

```
1 public entry fun register(account: &signer) {  
2     coin::register<ELON>(account);  
3 }
```

Under the modern Aptos coin ↔ fungible-asset model, registration is not a reliable indicator that a `CoinStore<ELON>` exists or that it is required before receiving funds. Core flows such as `coin::deposit()` may route to the Primary Fungible Store. This can mislead integrators into believing registration is a prerequisite or that it implies a published `CoinStore<ELON>` .

2.2.2. Impact

Integrators or frontends may introduce unnecessary transactions or incorrect UX flows, potentially causing operational errors when onboarding users or reconciling balances.

2.2.3. Recommendation

Clarify behavior in documentation/UI, or replace the entrypoint with an explicit “ensure primary fungible store exists” helper that matches expected behavior. For example, rename and update the function to make its intent explicit:

```
1 public entry fun ensure_primary_store(account: &signer) {  
2     primary_fungible_store::register(account, metadata());  
3 }
```

2.2.4. Remediation

Acknowledged by the project team.

2.3. [I-3] Manager Can Burn From Arbitrary Accounts

Target	echelon_coin/sources/echelon_coin.move
--------	--

Severity	● Informational
----------	-----------------

Category	Privileged Action
----------	-------------------

2.3.1. Description

The `burn_from()` entrypoint allows the manager to burn from any account:

```
1 public entry fun burn_from(manager: &signer, account_address: address, amount: u64)
  acquires Capabilities {
2     assert!(manager::is_manager(manager), ERR_ECHELON_COIN_UNAUTHORIZED);
3     let cap = borrow_global<Capabilities>(package::package_address());
4     coin::burn_from(account_address, amount, &cap.burn_capability);
5 }
```

The underlying `coin::burn_from()` is designed to bypass frozen semantics. As a result, freezing does not protect users against administrative burning. This is acceptable if the design intentionally grants this power to the manager, but it is important for integrators and users to understand the trust model.

2.3.2. Impact

Administrators can unilaterally reduce any account's balance, even if the account is frozen. This affects user expectations and risk assumptions for custody, exchange listings, and compliance workflows.

2.3.3. Recommendation

Explicitly document the administrative burn authority in public documentation and listing materials. If the intent is to restrict this power, remove `burn_from()` or gate it behind stricter governance controls.

2.3.4. Remediation

Acknowledged by the project team.