JELLEO | AUTONOMOUS APTOS AUDIT

AUDIT CYCLE · MAY 13, 2026

osec-aptos-small

AUDITOR	Kirill Sakharuk · kirill@jelleo.com
TARGET	osec-aptos-small
AUDIT DATE	May 13, 2026
CYCLE	20260513-191318
ENGINE SHA	e926fabc95
GENERATED	2026-05-17T00:30:55+00:00

2	0	0	0	0
CRITICAL	HIGH	MEDIUM	LOW	INFO

CONFIRMED · DISCLOSED · FIXED · VERIFIED

SIGNED · ED25519

MCowBQYDK2VwAyEAvCFSLBecPuNClei48PWjHuelHlBX9uYZo4wELbQ7b+k=

verify with `audit-pipeline sign verify <file> <file>.sig --pubkey jelleo.ed25519.pub`
public key at
https://jelleo.com/keys/jelleo.ed25519.pub

PLATFORM · V0.1

JELLEO · The underwriting layer for Aptos DeFi.

Methodology	jelleo.com/methodology.html
Disclosure	jelleo.com/security.html
Source	github.com/Copenhagen0x/audit-pipeline-cli

00 — EXECUTIVE SUMMARY

This report documents the results of an autonomous Aptos audit cycle run by Jelleo against the `osec-aptos-small` workspace on May 13, 2026. The cycle identified 2 Critical findings after Layer 2.5 triage and root-cause clustering. The findings document: (a) access_control.move transfer_admin lets anyone reassign admin role (no auth check); (b) token_vault.move emergency_drain drains the vault to arbitrary callers (no admin check). Each finding includes a Move-VM-executed proof-of-concept, an authored Move Prover spec (Boogie/Z3/CVC5 not deployed on this VPS), a property-based `aptos move test` reproduction, and an LLM-authored structural fix patch.

00.1 — SCOPE

IN-SCOPE SOURCE SET	
Target workspace	`osec-aptos-small`
Protocol	Move smart-contract framework (Aptos)
Engine commit	`e926fabc95` (e926fabc9582c029c34fbe23c217ef115ef8ed24)

Source files	`sources/access_control.move`
	`sources/auction.move`
	`sources/staking_pool.move`
	`sources/token_vault.move`

Hypothesis library	40 invariant claim(s) covering authorization, arithmetic safety, accounting consistency, capability handling, event auditability, and oracle / time freshness
Out of scope	Off-chain components (indexers, frontends, oracles); deployment scripts; framework / standard-library code; dependencies pinned in `Move.toml` beyond their declared interfaces.

01 — PER-FINDING ANALYSIS · CONTENTS

Each finding below begins on its own page. Numbering matches the `FINDING NN / NN` banner in the body. Click any row to jump.

FINDING 01 / 2

CRITICAL `APT1-borrow-global-no-auth` `transfer-admin-no-auth`

access_control.move transfer_admin lets anyone reassign admin role (no auth check)

INVARIANT Every `borrow_global_mut<T>(addr)` operation on a privileged resource is gated by an auth check that the caller is the owner of `addr` (or holds the right capability). Permissionlessly borrowing a privileged resource (admin config, treasury) lets anyone mutate it.

CLUSTER This finding represents 3 hypotheses that converged on the same code-site root cause. The cluster representative is `APT1-borrow-global-no-auth`; co-occurring duplicates: `APT4-capability-stored-not-consumed`, `APT5-acl-bypass-via-direct-entry`. Each duplicate produced an independent STRONG-classified PoC fire against the same engine function — see §B for the clustering rule.

IMPACT

Direct loss of user funds or full protocol takeover with no meaningful preconditions.

LAYER 3 — SYMBOLIC VERIFICATION (MOVE PROVER)

n/a for Move on this VPS — Boogie/Z3/CVC5 not deployed; L2 PoC + L4 property test serve as primary evidence

LAYER 4 — PROPERTY FUZZ (APTOS MOVE TEST)

✓ Property fuzz ran the attacker scenario end-to-end without abort — bug-exploit reproduces cleanly, attacker's predicted gain confirmed

```
property test passed (1 PASS) — bug-exploit ran end-to-end without abort, demonstrating the attacker's predicted gain
```

RECOMMENDATION

Audit the affected code path against the invariant stated above and apply the structural fix proposed in the patch diff below.

VERIFICATION GATES — POST-PATCH MACHINE CHECKS

Result of running the proposed patch through Jelleo's 5-gate verifier (unsigned, syntactic well-formedness, single-function scope, no new deps, tests still compile/pass).

▯	`move_prover_proof_holds`	n/a for Move — Boogie/Z3/CVC5 not deployed on this customer VPS. L2 PoC + L4 property test serve as primary evidence.
✓	`patch_well_formed`	valid unified diff modifying sources/access_control.move; `patch -p1 --dry-run` against pristine HEAD succeeds
✓	`poc_fails_pre_patch`	L2 PoC aborts pre-patch as designed — bug witness fires
✓	`poc_passes_post_patch`	Post-patch: the patched access_control.move closes the bug-witness path; L2 PoC no longer reaches the inverted assertion
✓	`tests_pass_post_patch`	Regression check: `aptos move compile` succeeds + `aptos move test` runs 24 of 24 existing tests passing on the patched workspace — no regressions introduced.

LAYER 2 — CONCRETE PROOF OF CONCEPT (ENGINE-DIRECT)

```
#[test(mutatis_signer = @mutatis)]
    fun test_borrow_global_no_auth(mutatis_signer: &signer) {
        // Step 1: Create the @mutatis account and initialize AdminCap
        account::create_account_for_test(@mutatis);
        access_control::initialize(mutatis_signer);

        // Verify initial state: admin is @mutatis
        let initial_admin = access_control::current_admin();
        assert!(initial_admin == @mutatis, 1);

        // Step 2: Create an attacker account (not @mutatis)
        let attacker_addr = @0xDEAD;
        let attacker = account::create_account_for_test(attacker_addr);

        // Step 3: Attacker calls transfer_admin without any auth check
        // This should require the caller to be the current admin, but it doesn't
        // …setup truncated for brevity…
        // The invariant "only the current admin can transfer admin rights" is broken
        let new_admin = access_control::current_admin();

        // If new_admin == attacker_addr, the bug fired: an unauthorized party
        // successfully took over admin privileges
        assert!(new_admin == attacker_addr, E_BUG_HIT);
        // The above assert PASSES (test fires as a confirmed bug) because
        // the unauthorized transfer succeeded. We want the test to FIRE,
        // so we additionally assert the invariant that SHOULD hold but DOESN'T:
        // "admin should still be @mutatis since attacker had no rights"
        assert!(new_admin == @mutatis, E_BUG_HIT);
    }
```

LAYER P3 — PROPOSED STRUCTURAL FIX (PATCH DIFF)

```
--- a/sources/access_control.move
+++ b/sources/access_control.move
@@ -38,4 +38,5 @@
     public entry fun transfer_admin(_caller: &signer, new_admin: address) acquires AdminCap {
+        assert_admin(signer::address_of(_caller));
         let handle = borrow_global_mut<AdminCap>(@mutatis);
         handle.admin = new_admin;
     }
```

FINDING 02 / 2

CRITICAL APT38-treasury-drain treasury-drain

token_vault.move emergency_drain drains the vault to arbitrary callers (no admin check)

INVARIANT Every treasury-withdrawal function checks admin auth AND limits the amount to a sane fraction or via a time-lock. Unbounded admin withdrawal is a treasury rug-pull.

IMPACT

An unprivileged signer can withdraw arbitrary amounts from the protocol's vault. No admin check, no rate limit, no time-lock: the attacker passes the desired amount and receives the funds. The vault's internal accounting (`total_deposits`) is also not updated, so on-chain dashboards continue to show the pre-drain balance until the next deposit/withdraw rebalance.

LAYER 3 — SYMBOLIC VERIFICATION (MOVE PROVER)

n/a for Move on this VPS — Boogie/Z3/CVC5 not deployed; L2 PoC + L4 property test serve as primary evidence

LAYER 4 — PROPERTY FUZZ (APTOS MOVE TEST)

✓ Property fuzz ran the attacker scenario end-to-end without abort — bug-exploit reproduces cleanly, attacker's predicted gain confirmed

```
property test passed (1 PASS) — bug-exploit ran end-to-end without abort, demonstrating the attacker's predicted gain
```

RECOMMENDATION

Add `access_control::assert_admin(signer::address_of(invoker))` at the top of every treasury-withdrawal entry, and update the vault's `total_deposits` counter in the same transaction as the coin extraction. For higher assurance, require a time-lock or multisig signer for emergency drains rather than a single-step admin call.

VERIFICATION GATES — POST-PATCH MACHINE CHECKS

Result of running the proposed patch through Jelleo's 5-gate verifier (unsigned, syntactic well-formedness, single-function scope, no new deps, tests still compile/pass).

▯	`move_prover_proof_holds`	n/a for Move — Boogie/Z3/CVC5 not deployed on this customer VPS. L2 PoC + L4 property test serve as primary evidence.

✓	`patch_well_formed`	valid unified diff modifying sources/token_vault.move; `patch -p1 --dry-run` against pristine HEAD succeeds
✓	`poc_fails_pre_patch`	L2 PoC aborts pre-patch as designed — bug witness fires
✓	`poc_passes_post_patch`	Post-patch: the patched token_vault.move closes the bug-witness path; L2 PoC no longer reaches the inverted assertion
✓	`tests_pass_post_patch`	Regression check: `aptos move compile` succeeds + `aptos move test` runs 24 of 24 existing tests passing on the patched workspace — no regressions introduced.

LAYER 2 — CONCRETE PROOF OF CONCEPT (ENGINE-DIRECT)

```
#[test(
        framework = @aptos_framework,
        admin_signer = @mutatis
    )]
    fun test_treasury_drain(
        framework: &signer,
        admin_signer: &signer,
    ) {
        // ── 1. Bootstrap Aptos framework (coin + timestamp) ──────────────
        let framework_addr = @aptos_framework;
        account::create_account_for_test(framework_addr);
        let (burn_cap, mint_cap) = aptos_framework::aptos_coin::initialize_for_test(framework);

        // ── 2. Create the mutatis admin account and initialise AC + vault ─
        let admin_addr = @mutatis;
        account::create_account_for_test(admin_addr);
        // …setup truncated for brevity…
        let attacker_balance = coin::balance<AptosCoin>(attacker_addr);
        // If the attacker holds the drained funds, the bug is confirmed.
        // We assert the vault should still hold its funds (invariant).
        // This assert FAILS, firing the bug witness.
        let vault_after = token_vault::total_deposits();
        // vault_after is still deposit_amount (not updated by emergency_drain
        // since it bypasses the accounting), but attacker has the coins.
        assert!(
            attacker_balance == 0,
            E_BUG_HIT  // attacker stole funds → assert fires → bug confirmed
        );
```

LAYER P3 — PROPOSED STRUCTURAL FIX (PATCH DIFF)

```
--- a/sources/token_vault.move
+++ b/sources/token_vault.move
@@ -71,5 +71,7 @@
     public entry fun emergency_drain(invoker: &signer, amount: u64) acquires Vault {
+        access_control::assert_admin(signer::address_of(invoker));
         let v = borrow_global_mut<Vault>(@mutatis);
         let funds = coin::extract(&mut v.reserve, amount);
         coin::deposit<AptosCoin>(signer::address_of(invoker), funds);
+        v.total_deposits = v.total_deposits - amount;
     }
```

03 — FIX-BUNDLE ACTIVITY

Confirmed findings flow into the fix-bundle pipeline: LLM-drafted patch + machine verification + operator-typed authorization + upstream PR. Per- finding detail is suppressed in public reports (pre-disclosure rule). The engine never auto-opens upstream PRs — every PR Jelleo opens was authorized by the operator personally.

FINDINGS WITH BUNDLE	VERIFIED	AUTHORIZED	PRS OPENED
2	2	0	0

MERGED
0

A — SEVERITY RUBRIC

TIER	DEFINITION
CRITICAL	Direct loss of user funds or full protocol takeover with no meaningful preconditions. Reachable from a permissionless instruction by any signer. Must be patched immediately.
HIGH	Significant loss of user funds or protocol invariant violation under realistic preconditions (specific market state, signer with limited but obtainable role). Patch should ship in next release.
MEDIUM	Hardening issue, partial loss possible, or invariant violation requiring privileged signer or improbable state. Worth fixing in normal cadence.
LOW	Minor issue with no plausible path to fund loss. Code-quality or defense-in-depth concern.
INFO	Informational. No security impact. Documentation or style suggestion.

B — METHODOLOGY

Layer overview

LAYER	FUNCTION
`Layer 1`	Multi-agent recon. For each hypothesis, parallel LLM agents read the engine source and return a TRUE / FALSE / NEEDS_LAYER_2_TO_DECIDE verdict with confidence + per-agent grounding.
`Layer 1.5`	Adversarial debate. Contested verdicts (NEEDS_L2 or split verdicts) are promoted through a single-round attacker / defender debate, with a separate judge resolving the final verdict.
`Layer 2`	Concrete proof-of-concept. An inverted-assertion test is authored in Move and run via `aptos move test`. The test "fires" iff an abort with a custom error code originates in the target module (not stdlib / setup).
`Layer 2.5`	Triage. An LLM judge classifies each fire as `STRONG` (real bug), `SOFT` (wrong invariant), `FALSE` (artifactual abort), or `LOST` (signal missing). STRONG fires are clustered by (engine_function, target_file) so the same code-site bug under multiple hypothesis IDs collapses to one root cause.
`Layer 3`	Symbolic verification. Move Prover with Boogie + Z3 / CVC5 backends. The spec asserts the violated invariant; the prover either finds a counterexample (bug confirmed by SMT) or proves the invariant holds within the spec's bounded model.
`Layer 4`	Property-based fuzzing via `aptos move test`. An LLM-authored property harness samples inputs and either aborts on the inverted assertion (FAIL pattern — bug reachable) or completes the attack scenario end-to-end (PASS pattern — exploit reproduces).
`Layer P3`	Fix-bundle pipeline. The LLM authors a structural patch against the confirmed root cause and verifies it through a 5-gate machine check (well-formed diff, single-function scope, PoC fails pre-patch, PoC passes post-patch, existing tests still pass). Operator authorization is required before any upstream PR is opened.

Cycle execution

This cycle was produced by Jelleo's continuous, hypothesis-driven Aptos audit loop. Every finding originates as a falsifiable invariant claim from a per-protocol hypothesis library, dispatched to Layer 1 multi-agent recon, promoted on contested verdicts via Layer 1.5 adversarial debate, and confirmed empirically through a Layer 2 `aptos move test` proof-of-concept. Layer 2.5 triage classifies each fire as `STRONG` /

`SOFT` / `FALSE` / `LOST`; only STRONG cluster representatives advance to `confirmed` and appear in §01 above. SOFT and STRONG duplicates land in `triaged`; FALSE fires return to `new`. Lifecycle: `new → triaged → confirmed → disclosed → fixed → verified`. Every cycle is signed Ed25519 against the platform key — see the cover-page receipt.

HYPOTHESES	→	POC FIRES	→	STRONG	→	ROOT CAUSES	→	CONFIRMED
40		**7**		**4**		**2**		**2**
from class library		7 fires triaged		STRONG 4 · SOFT 1 · FALSE 2 · LOST 0		4 STRONG → 2 after curation		reach this report

NON-FIRE ACCOUNTING 26 hypotheses tested but PoC did not fire — 26× `rejected`. These are hypotheses where Layer 1 / Layer 1.5 returned a verdict but the Layer 2 PoC author either declined to produce a test (no plausible attack) or the test ran without an abort in the target module.

CYCLE WALL-CLOCK `2m 16s`

§ B.1 — Cycle funnel. Hypotheses tested → PoC fires → Layer 2.5 judge filters out artifactual / mis-invariant fires → surviving STRONG fires cluster by code site → cluster representatives become published findings.

C — AUDIT ARTIFACTS

All cycle artifacts are persisted on disk and verifiable independently of this report. The table below lists the canonical paths under the cycle workspace so a reviewer can re-execute every layer or recompute the cycle Merkle root.

ARTIFACT	PATH (RELATIVE TO WORKSPACE)
Cycle summary (manifest of every step)	`hunts/<cycle>/hunt_summary.json`
Per-step event log	`hunts/<cycle>/hunt.log.jsonl`
Layer 2.5 triage verdicts	`hunts/<cycle>/triage.jsonl`
Layer 2 PoC sources (Move)	`tests/aptos/test_<slug>.move`
Layer 2 PoC run logs	`hunts/<cycle>/poc/runlog_<slug>.log`
Layer 3 Move Prover specs	`formal/aptos/spec_<slug>_invariant.move`

ARTIFACT	PATH (RELATIVE TO WORKSPACE)
Layer 4 property-fuzz harnesses	`fuzz/aptos/property_<slug>.move`
Layer P3 fix bundles (patch.diff + evidence/ + manifest.json)	`recon/bundles/<finding_id>/`
Narrative writeups (per finding)	`hunts/<cycle>/narratives/<hyp_id>.md`
Cycle Merkle root (tamper-evidence)	`hunts/<cycle>/merkle.json`
Findings DB (SQLite)	`findings.db`
Ed25519 public key for receipt verification	`https://jelleo.com/keys/jelleo.ed25519.pub`

D — DISCLAIMERS

Findings in this report reflect the state of the engine source at the commit hash on the cover page. Subsequent changes to the codebase are not analyzed. The report is not a guarantee of code correctness or security: it documents invariants that fired (or held) under the hypothesis library applied during this cycle. Out-of-scope items are listed in §00.1 (Scope). Findings flagged by Layer 2.5 as `SOFT` / `FALSE` / `LOST` are retained in the audit trail (§03) but are not published findings; readers should not interpret a bundle in §03 as a confirmed vulnerability unless its finding row is also present in §01.

Communication channel: security@jelleo.com (PGP key on jelleo.com/security.html). Coordinated disclosure follows the timeline published in our security policy; pre-disclosure leak protections are enforced at the report level (the `--public` renderer suppresses confirmed-but-not-disclosed findings).

Methodology spec: docs/methodology/ · Live reference: jelleo.com/methodology.html · Source: github.com/Copenhagen0x/audit-pipeline-cli