JELLEO | AUTONOMOUS SOLANA AUDIT

AUDIT CYCLE · MAY 26, 2026 · BOUNTY 6

UpdateAssetLifecycle ACTIVATE future-slot DoS

AUDITOR	Kirill Sakharuk · kirill@jelleo.com
TARGET	percolator-cli · v16 perpetual DEX engine + Solana program wrapper
AUDIT DATE	May 26, 2026
ENGINE HEAD	9bcf002b
WRAPPER HEAD	0925ed4
SEVERITY	HIGH · LATENT ON BHKMIC5G…
EVIDENCE	L2 engine-direct (3 tests) · L3 SKIP (state-machine class) · L4 LiteSVM (2 tests) · L6 fix verified

LATENCY NOTE

Bug is real in committed code at engine 9bcf002b / wrapper 0925ed4. On the live bounty 6 market BhkMic5g… (max_market_slots=4, all 4 configured, free_market_slot_count=0), both reachable paths are currently closed — the bug fires on any market with slot headroom or any future admin RETIRE.

PLATFORM · V0.1

JELLEO · The underwriting layer for Solana DeFi.

Methodology	jelleo.com/methodology.html
Disclosure	jelleo.com/security.html
Bounty	Superteam · percolator-cli bounty 6

00 — EXECUTIVE SUMMARY

Single finding. The wrapper handler `handle_update_asset_lifecycle` ACTIVATE arm at `percolator-prog/src/v16_program.rs:7754-7764` passes the caller-supplied `now_slot` instruction argument directly into the engine call `activate_empty_market_slot_not_atomic` without wrapping it in `authenticated_slot_or_fallback`. The engine's guard at `percolator/src/v16.rs:4468` only rejects past slots (`now_slot < self.current_slot.get()`) — there is no upper bound.

With `now_slot = u64::MAX`, the engine stamps `asset.slot_last = u64::MAX` and `header.current_slot = u64::MAX`. After that, every subsequent `PermissionlessCrank`, `SyncMaintenanceFee`, `ResolveStalePermissionless`, `ForceCloseAbandonedAsset`, and liquidation path reverts with `Err(InvalidConfig)` because the accrue guard at `v16.rs:7542` / `:14513` requires `now_slot ≥ self.header.current_slot.get()`. Maintenance-fee accrual to `header.insurance` stops permanently; liquidation cranking is blocked; the market freezes.

Status: **LATENT** on the live bounty 6 market `BhkMic5gHLjj5Uxkg6rBBXofUzeTZVwmV4uFzfhwtgQw` today because the market has hit its slot cap (`max_market_slots = 4`, all 4 configured, `free_market_slot_count = 0`) — both reachable code paths into the buggy call are closed by the cap. The bug fires on any future market with slot headroom or once any slot is RETIRED. Both L2 engine-direct tests and L4 LiteSVM tests flip PASS→FAIL with the proposed fix applied (cargo cache scrubbed for engine git-dep propagation).

00.1 — SCOPE

Target	`percolator-cli` bounty 6 · v16 engine + Solana program wrapper
Engine repo	`github.com/aeyakovenko/percolator` · HEAD `9bcf002b`
Wrapper repo	`github.com/aeyakovenko/percolator-prog` · HEAD `0925ed4`
Win condition	drop `engine.insurance_fund.balance` via public-instruction calls (per bounty 5 README, inherited by bounty 6)
Out of scope	Off-chain components, deployment scripts, framework / standard-library code, dependencies beyond their declared interfaces

00.2 — METHODOLOGY

Standard 6-tier pipeline applied to a single hypothesis surface (H2 from the bounty 6 fresh-surface map):

L1	**Hunt** — 4 fresh-surface hypotheses generated from the engine 89f25ce → 9bcf002b + wrapper 7f7cefc → 0925ed4 diff (3 new attack surfaces + 1 sanity check). H2 is the accruable-summary DoS hypothesis.
L1.5	**Triage** — paranoid-goober agent scans the indexed bounty 6 codebase. Identifies the missing `authenticated_slot_or_fallback` wrap in the ACTIVATE branch as the trigger surface, traces to the cascade in `accrue_asset_to_not_atomic`.
L2	**POC reproduction** — 3 engine-direct Rust tests in `percolator/tests/v16_bounty6_h2.rs`: activate accepts u64::MAX, accrue DoS cascade, source-pin on the missing guard. All PASS on baseline engine 9bcf002b.
L3	**Kani** — SKIP. The bug is state-machine missing-input-validation, not arithmetic-invariant. CBMC's strength is the latter. L2 + L4 cover this class.
L4	**LiteSVM behavioral** — 2 BPF reproductions in `percolator-prog/tests/v16_cu.rs`: `v16_bpf_bounty6_h2_update_asset_lifecycle_activate_accepts_spoofed_future_slot` proves the engine stamps `u64::MAX` into `header.current_slot` through the real program ABI; `v16_bpf_bounty6_h2_permissionless_crank_dos_after_future_slot_activation` proves the cascade — `PermissionlessCrank` reverts even after warping SVM clock to slot 1,000,000. BPF binary built via `cargo build-sbf --no-default-features` against wrapper 0925ed4.
L5	**Narrative** — this report.
L6	**Fix bundle + verify** — two patches (wrapper + engine). Applied to working tree + cargo git cache + rlib scrubbed + BPF rebuilt. L2 and L4 tests both flip PASS→FAIL. Patches reverted post-verification.

FINDING 01 / 01

HIGH | LATENT · CAP-BLOCKED TODAY | H2 | PATCH FLIPS POC

`handle_update_asset_lifecycle` ACTIVATE arm accepts caller-supplied future `now_slot` — permanent permissionless-cranker DoS via future-stamped `header.current_slot`

AFFECTED CODE

- **Wrapper** `percolator-prog/src/v16_program.rs:7754-7764` (`handle_update_asset_lifecycle` append-activation branch). Passes raw `now_slot` to `state::activate_dynamic_asset_slot` .
- **Wrapper** `percolator-prog/src/v16_program.rs:7696-7704` (reuse-activation branch). Same — passes raw `now_slot` to `activate_empty_market_slot_not_atomic` .
- **Wrapper** `percolator-prog/src/v16_program.rs:7810` (SHUTDOWN arm, for contrast). Correctly wraps: `let authenticated_slot = authenticated_slot_or_fallback(now_slot);`
- **Engine** `percolator/src/v16.rs:4453-4587` (`activate_empty_market_slot_not_atomic`). Guard at line 4468 only rejects PAST slots; no upper-bound check.
- **Engine cascade** `percolator/src/v16.rs:7542` (view-mut `accrue_asset_to_not_atomic`) and `:14513` (runtime mirror). Returns `Err(InvalidConfig)` when `now_slot < self.header.current_slot.get()` .

BUG

The engine guard:

```
// percolator/src/v16.rs:4465-4471
if decode_market_mode(self.mode)? ≠ MarketModeV16::Live
    || authenticated_price == 0
    || authenticated_price > MAX_ORACLE_PRICE
    || now_slot < self.current_slot.get()    // past-slot reject only
{
    return Err(V16Error::InvalidConfig);
}
```

With `now_slot = u64::MAX` , the guard passes. The engine then stamps:

- `self.current_slot = V16PodU64::new(now_slot)` (line 4581) → `header.current_slot = u64::MAX`
- `asset.slot_last = now_slot` (line 4552) → `asset[i].slot_last = u64::MAX`

This state is committed to the on-chain market account and persists across instructions. Every subsequent cranker entry point that calls `accrue_asset_to_not_atomic` hits the guard at `v16.rs:7542` :

```
// percolator/src/v16.rs:7537-7545 (view-mut accrue)
if asset_index ≥ config.max_market_slots as usize
    || asset_index ≥ self.markets.len()
    || effective_price == 0
    || effective_price > MAX_ORACLE_PRICE
    || funding_rate_e9.unsigned_abs() > config.max_abs_funding_e9_per_slot as u128
    || now_slot < self.header.current_slot.get()    // bug cascade fires here
```

```
{
    return Err(V16Error::InvalidConfig);
}
```

With `header.current_slot = u64::MAX` and any real `Clock.slot < u64::MAX`, every cranker call reverts.

IMPACT

Class: permanent DoS of all permissionless cranker progress on the affected market.

Reverted entry points:

- `PermissionlessCrank` (Refresh, Recover, Liquidate, SettleB)
- `SyncMaintenanceFee` — the only path that grows `header.insurance` from accrued fees
- `ResolveStalePermissionless`
- `ForceCloseAbandonedAsset`
- Any internal path that invokes `accruable_asset_slot_summary`

Consequences: permanent insurance starvation (no fee accrual), bankrupt-account accumulation (no liquidation), resolution-mode blocked, operator-funded insurance becomes a one-way deposit.

The bounty 6 win condition is "drop `engine.insurance_fund.balance` via public-instruction calls." This bug doesn't drop insurance directly; it freezes the only mechanism that allows insurance to grow and to disburse, leaving operator/protocol value stranded.

STATUS

LATENT on the live bounty 6 market `BhkMic5gHLjj5Uxkg6rBBXofUzeTZVwmV4uFzfhwtgQw` as of 2026-05-26.

On-chain configuration (queried via Solana RPC `solana account BhkMic5g…` + parsed `WrapperConfigV16` at byte offsets 144 / 390 within the market account):

- `permissionless_market_init_fee = 5,864,605` lamports (~0.00586 SOL) — public ACTIVATE path is open
- `max_market_slots = 4`, currently 4 configured (m0, m1, m2, retired m3), `free_market_slot_count = 0`

Both reachable code paths into the buggy engine call are currently closed:

- **APPEND** (`asset_index == configured_slots`): wrapper guard at `v16_program.rs:7683-7686` requires `free_market_slot_count == 0` (currently true), but engine guard at `v16.rs:4462` rejects `asset_index ≥ max_market_slots = 4`.
- **REUSE** (`asset_index < configured_slots`): wrapper guard at `:7621` requires `free_market_slot_count ≠ 0` (currently false). Only admin can RETIRE a slot to increment the counter.

The bug fires on any market where one of:

- `configured_slots < max_market_slots` (APPEND open) — applies to any fresh deployment with headroom
- `free_market_slot_count > 0` (REUSE open) — applies whenever admin RETIRE has been called and the slot hasn't been re-claimed

EXISTING POC

Engine-direct L2 (3 tests, passing on engine HEAD `9bcf002b`):

```
percolator/tests/v16_bounty6_h2.rs
  bounty6_h2_activate_accepts_u64_max_future_slot          PASS
  bounty6_h2_accrue_dos_after_future_slot_activation       PASS
```

```
  bounty6_h2_source_pin_engine_only_rejects_past_slot      PASS
```

Wrapper-BPF L4 LiteSVM (2 tests, passing against the deployed-binary path at wrapper HEAD `0925ed4`):

```
percolator-prog/tests/v16_cu.rs
  v16_bpf_bounty6_h2_update_asset_lifecycle_activate_accepts_spoofed_future_slot    PASS
  v16_bpf_bounty6_h2_permissionless_crank_dos_after_future_slot_activation         PASS
```

L4 tests `cargo build-sbf --no-default-features` against wrapper `0925ed4`, then exercise the full instruction round-trip through LiteSVM (BPF binary at `target/deploy/percolator_prog.so`). The cascade test verifies that `PermissionlessCrank` reverts both immediately after activation and after warping the SVM clock to slot 1,000,000.

FIX (TWO PATCHES — BOTH SHOULD LAND)

Wrapper: wrap `now_slot` with `authenticated_slot_or_fallback` in both ACTIVATE branches, matching the SHUTDOWN arm pattern at `:7810`:

```
--- a/src/v16_program.rs
+++ b/src/v16_program.rs
@@ -7693,7 +7693,7 @@
                         group
                             .header
                             .activate_empty_market_slot_not_atomic(
                                 asset_index as u32,
                                 &mut group.markets[asset_index],
                                 initial_price,
-                                now_slot,
+                                authenticated_slot_or_fallback(now_slot),
                             )
                             .map_err(map_v16_error)?;
@@ -7751,7 +7751,7 @@
                 if !reuse_activated && asset_index == configured_slots {
                     let profile = state::activate_dynamic_asset_slot(
                         &mut data,
                         asset_index,
-                        now_slot,
+                        authenticated_slot_or_fallback(now_slot),
                         initial_price,
                         insurance_authority,
                         insurance_operator,
                         backing_bucket_authority,
```

Engine: defense-in-depth — reject `now_slot` more than `2 × asset_activation_cooldown_slots` past `self.current_slot` in `activate_empty_market_slot_not_atomic`:

```
--- a/src/v16.rs
+++ b/src/v16.rs
@@ -4465,11 +4465,18 @@ impl MarketGroupV16HeaderAccount {
         if decode_market_mode(self.mode)? != MarketModeV16::Live
             || authenticated_price == 0
             || authenticated_price > MAX_ORACLE_PRICE
             || now_slot < self.current_slot.get()
         {
             return Err(V16Error::InvalidConfig);
```

```
        }
+       // Reject far-future slots. `now_slot` is caller-supplied through the
+       // wrapper's UpdateAssetLifecycle ACTIVATE arm; without an upper bound,
+       // a permissionless caller can stamp `current_slot = u64::MAX` and
+       // permanently freeze every subsequent `accrue_asset_to_not_atomic`
+       // call (which guards `now_slot < self.current_slot.get()`).
+       let max_drift = config.asset_activation_cooldown_slots.saturating_mul(2);
+       if now_slot > self.current_slot.get().saturating_add(max_drift) {
+           return Err(V16Error::InvalidConfig);
+       }
        config.validate_public_user_fund_shape()?;
```

VERIFICATION VERDICT

PASS_VERIFIED. Both fix patches applied to engine working tree + cargo git cache + wrapper working tree. Wrapper `target/sbpf-solana-solana/release/deps/libpercolator-*.rlib` scrubbed and `target/debug/deps/libpercolator-*.rlib` scrubbed. BPF rebuilt via `cargo build-sbf --no-default-features` (recompiled engine + wrapper). L2 and L4 tests re-run:

- **L2:** all 3 tests flipped PASS → FAIL. `bounty6_h2_activate_accepts_u64_max_future_slot` now sees `Err(InvalidConfig)` from the new engine upper-bound guard. `bounty6_h2_source_pin_engine_only_rejects_past_slot` panics on the new `saturating_add` string the source-pin asserts absent.
- **L4:** both BPF tests flipped PASS → FAIL. `activate_permissionless_asset_with_fee( … , u64::MAX, … )` now returns `FailedTransactionMetadata { err: InstructionError(2, InvalidAccountData), … "Program failed: invalid account data for instruction" }`. The wrapper's `authenticated_slot_or_fallback` resolves the spoofed slot to `Clock.slot`, and the engine's defense-in-depth upper-bound rejects any residual future-slot attempt.

Patches reverted post-verification; baseline restored; baseline tests re-PASS (bug reproduces on baseline). The disclosure ships engine + wrapper patches; the team chooses whether to land one, the other, or both — both are recommended for defense-in-depth.

---

99 — APPENDIX

A1 — How this finding was reached

Bounty 6 was opened on the same target as bounty 5 (`percolator-cli` v16 multi-market group). The engine and wrapper had moved between bounties: engine `89f25ce → 9bcf002b` (3 commits — insurance-domain isolation `8e0e3f8`, Kani tractability `b757c76`, loss-stale + fee fix `9bcf002`) and wrapper `7f7cefc → 0925ed4` (6 commits including resolved-close delegation `0925ed4` and post-mutation shape validation `a52e1f6`). 4 fresh-surface hypotheses were generated against the bounty 6 baseline (engine `9bcf002b` / wrapper `0925ed4`):

- **H1** — `asset_contributes_to_loss_stale_summary` predicate manipulation. NO HIT.
- **H2** — `accruable_asset_slot_summary` DoS via future-slot. **HIT** (this finding).
- **H3** — Domain-budget sum-invariant break (`8e0e3f8`). NO HIT.
- **H4** — Trade-fee actual-vs-requested gap (`9bcf002`). NO HIT.

A2 — Test reproduction

From a fresh clone of `aeyakovenko/percolator` @ `9bcf002b` + `aeyakovenko/percolator-prog` @ `0925ed4`, copy the test files attached to this report into the respective `tests/` dirs.

- Engine L2: `cd percolator && cargo test --features test --test v16_bounty6_h2`
- Wrapper L4: `cd percolator-prog && cargo build-sbf --no-default-features && cargo test --test v16_cu v16_bpf_bounty6_h2`

A3 — Cargo git-cache caveat

`percolator-prog`'s `Cargo.toml` pins the engine as a git dep at rev `9bcf002b`. Local edits to `percolator/src/v16.rs` are NOT picked up by wrapper-side test compilation unless the same edit is applied to `~/.cargo/git/checkouts/percolator-*/9bcf002/src/v16.rs` AND both `percolator-prog/target/debug/deps/libpercolator-*.rlib` and `percolator-prog/target/sbpf-solana-solana/release/deps/libpercolator-*.rlib` are deleted before rebuild. Verification work for this finding scrubbed both rlibs and rebuilt BPF; without that the wrapper test would compile against the unpatched engine and the fix wouldn't propagate.

END OF REPORT · bounty 6 H2 · 2026-05-26