SnapSmack Security Audit

018 — Hub-Initiated Maintenance Mode (Multisite)

Version 0.7.171 · 2026-05-23 · Sean McCormick & Claude

Feature	Hub-initiated maintenance mode toggle for spoke sites
Files	core/multisite-api.php, smack-multisite.php
Migration	migrate-spoke-maintenance-mode.sql
Severity	Informational — no new attack surface identified
Status	CLOSED — ships in 0.7.171

Overview

0.7.171 adds hub-initiated maintenance mode to the multisite system. A hub administrator can now put any connected spoke into maintenance mode — or take it out — directly from the Multisite Management dashboard, without logging into each spoke. Two new UI controls ship: a per-row MAINT ON / MAINT OFF toggle in the spoke table, and MAINTENANCE ALL ON / MAINTENANCE ALL OFF bulk buttons for fleet-wide operations. A new API endpoint (POST multisite/maintenance/set) handles the spoke-side command. Maintenance state is cached in snap_multisite_nodes.maintenance_mode and refreshed on every heartbeat sweep.

Finding 1 — Authentication: No New Surface

The new multisite/maintenance/set endpoint sits behind the same Bearer API key authentication as every other hub-write endpoint (e.g. updates/trigger, disconnect). The key is validated against snap_multisite_nodes.api_key_local before reaching the endpoint body. Hub role is additionally asserted:

```
if ($node['role'] !== 'hub') ms_err('Only a hub may set maintenance mode', 403);
```

A compromised api_key_local would allow an attacker to toggle maintenance mode, but that same key already grants access to updates/trigger — which extracts and executes arbitrary release packages. Maintenance mode toggle is strictly lower-severity than existing endpoints. No escalation path introduced.

> Verdict: Informational. Existing key security posture unchanged.

Finding 2 — Input Validation on mode Parameter

The endpoint accepts a JSON body with a single mode key. It is cast to boolean then to the string '0' or '1' before being written to snap_settings via a prepared statement:

```
$mode = $body['mode'] ? '1' : '0'; $pdo->prepare("INSERT INTO snap_settings ... ON
DUPLICATE KEY UPDATE ...") ->execute([$mode, $mode]);
```

PHP's truthy cast means any non-empty, non-zero value enables maintenance. The prepared statement prevents SQL injection. No unvalidated user content reaches the database. The maintenance gate reads the value with a strict string comparison (=== '1'), so only the literal string '1' activates it.

> Verdict: Closed. Input handling is correct.

Finding 3 — SSRF Considerations (Hub-Side Outbound)

The hub issues a cURL POST to each spoke's API URL. This is the same outbound pattern already used by the update trigger and heartbeat sweep — both of which existed before this build. The SSRF guard (snap_is_private_url()) is checked at registration time; spoke URLs in snap_multisite_nodes are pre-validated. The maintenance command uses the stored URL directly, consistent with all other hub→spoke calls. No new SSRF surface is introduced.

> Verdict: Informational. No change from existing posture.

Finding 4 — Offline Spoke Handling

The bulk MAINTENANCE ALL ON/OFF handler iterates all active spokes and fires cURL with a 10-second timeout per spoke. If a spoke is unreachable, the handler records ok: false for that node and continues — no exception terminates the loop. The hub DB is only updated (maintenance_mode = ?) on confirmed success ($result['ok'] true). This means a spoke that was offline will not have its cached state incorrectly flipped. On next heartbeat, the true state is refreshed from the spoke directly.

Note: the 10-second per-spoke timeout means a fleet with many offline spokes could cause the page to load slowly during a bulk operation. This is consistent with the existing update trigger behaviour and is acceptable for an admin-only operation.

> Verdict: Closed. Failure isolation is correct.

Finding 5 — Maintenance Gate Does Not Block API

core/maintenance-gate.php is included only on public-facing pages after $settings is populated. It is not included by api.php. This means that while a spoke is in maintenance mode, the hub can still reach its API to issue the MAINT OFF command (or any other command). This is intentional and correct — a hub-initiated maintenance mode must be reversible from the hub without manual intervention.

> Verdict: Closed. By design.

No security findings require remediation. Feature ships as implemented in 0.7.171.