SNAPSMACK — SECURITY AUDIT

Audit #020 · 2026-06-04 · v0.7.200 “Barcalounger+”

Audit number	020
Date	2026-06-04
Release	0.7.200 / 0.7.201 (pending)
Auditors	Claude (Anthropic), Sean Moir
Scope	Hub/spoke multisite attack surface — full read of core/multisite-api.php, smack-push-it.php, smack-settings.php, smack-back.php, core/sidebar.php, core/mesh-helpers.php
Audit type	Deep manual code review — complete read of all multisite API endpoints and admin push-control surfaces
Prior audit	019 — 2026-05-26 — SMACKBACK Skin JS Scanner, archive landing, update track
Background	Sean raised a design question about whether hub/spoke stats should be shared across sites, which led to a f

Findings Overview

#	Title	Severity	Status
1	SSRF in posts/create — hub-supplied img_url fetched without IP guard	High	Closed
2	SSRF in skins/reinstall — hub-supplied download_url fetched without IP guard	High	Closed
3	Hub can silently disable SMACKBACK on all spokes via settings push	High	Closed
4	Hub can downgrade SMACKBACK response mode (lockout → alert) silently	Medium	Open
5	hub_controls_* flags never delivered to spokes — spoke locking broken end-to-end	Medium	Open
6	AI key push silently fails — keys not on settings/push API allowlist	Low	Open
7	No warning on spoke connection page re: hub authority	Low	Open
8	SSO endpoint grants hub full spoke admin access	Informational	By design

---

Finding 1 — SSRF in posts/create — Hub-Supplied img_url Fetched Without IP Guard [High]

`Location: core/multisite-api.php — POST multisite/posts/create, ~line 706`

The cross-post endpoint accepts an img_url parameter from the hub and fetches it via curl to save a copy of the image on the spoke. The original code validated the URL with filter_var(FILTER_VALIDATE_URL) but performed no check on whether the resolved host was a private, loopback, or reserved address. A compromised hub could supply an img_url such as http://192.168.1.1/admin or http://10.0.0.1/internal to probe the spoke’s internal network or local services. The spoke would silently make the request, and while the response is expected to be an image (validated via getimagesizefromstring), an attacker who controls a host that returns a valid image at an RFC-1918 address could use this to confirm internal hosts exist. More practically, on a shared hosting environment this could reach other customers’ sites via localhost routing.

Fix applied (0.7.201):

ms_is_safe_remote_url() added to core/mesh-helpers.php. The function rejects any URL whose host resolves to a loopback, link-local, private (RFC-1918/RFC-4193), or reserved address, including the literals localhost, ::1, and 0.0.0.0. The check runs before curl is invoked. Any private-host URL returns HTTP 403 to the hub.

Verdict: Closed. Fixed in 0.7.201. ms_is_safe_remote_url() guard applied before curl fetch.

Finding 2 — SSRF in skins/reinstall — Hub-Supplied download_url Fetched Without IP Guard [High]

`Location: core/multisite-api.php — POST multisite/skins/reinstall, ~line 1174`

The skin reinstall endpoint accepts a download_url from the hub and passes it to skin_registry_install(), which fetches the ZIP via curl. The same absence of private-IP filtering applies as in Finding 1. A compromised hub could supply an internal address as download_url, causing the spoke to make an outbound request to an internal host. The signature verification step (which checks the downloaded payload against a public key stored on the spoke) would fail for a fabricated payload, limiting the attack to SSRF probe only — code execution via this path is not achievable as long as the spoke's update_public_key is not on the settings push allowlist (it is not). However, the SSRF probe itself is still a finding.

Fix applied (0.7.201):
ms_is_safe_remote_url() called on download_url before skin_registry_install() is invoked. Private/loopback hosts return HTTP 403.

Verdict: Closed. Fixed in 0.7.201. ms_is_safe_remote_url() guard applied before ZIP fetch.

Finding 3 — Hub Can Silently Disable SMACKBACK on All Spokes via Settings Push [High]

`Location: core/multisite-api.php — POST multisite/settings/push, ~line 1195`

smackback_enabled is on the settings push allowlist. A hub pushing smackback_enabled=0 would instantly disable file integrity monitoring on every connected spoke with no spoke-side confirmation. This is the highest-risk item in the audit: a compromised hub's optimal attack sequence is (1) disable SMACKBACK fleet-wide, (2) push tampered files or skin packages. Without SMACKBACK running, no tamper detection fires and no breach alert is sent. The attack is fully silent from the spoke's perspective.

Fix applied (0.7.201):
The settings push handler now intercepts smackback_enabled=0 before the allowlist write. Instead of applying it, the handler writes smackback_hub_pending_disable=1 to snap_settings and adds smackback_enabled to a pending_confirmation array in the API response. The smack-back.php admin page detects this pending flag and renders a high-contrast orange warning block requiring the spoke admin to explicitly approve or reject the disable request. Approval requires a browser confirm() dialog as a second gate. The hub can still push smackback_enabled=1 (enabling) and smackback_mode changes immediately — only the disable path is gated. A rejected request is cleared with no notification to the hub.

Design rationale: Hub/spoke is a single-owner topology. If the hub is requesting SMACKBACK be disabled, the owner can confirm it locally. If the hub is compromised and the owner did not initiate the request, the local confirmation gate prevents silent disabling.

Verdict: Closed. Fixed in 0.7.201. Disable requests staged as pending; spoke admin confirmation required.

Finding 4 — Hub Can Downgrade SMACKBACK Response Mode Silently [Medium]

`Location: core/multisite-api.php — POST multisite/settings/push, allowlist`

smackback_mode is on the settings push allowlist and is applied immediately without confirmation. A compromised hub can push smackback_mode=alert, downgrading a spoke from lockout or paranoid mode to alert-only. In alert mode, a breach does not lock the spoke admin out — the attacker retains admin access even after a tamper is detected. Combined with Finding 3 (now fixed), this was the second step of the optimal attack chain. With Finding 3 fixed, the attacker cannot disable SMACKBACK entirely, but can still soften its response before a tampering attempt.

Status: Open.
Recommended fix: Apply the same pending-confirmation gate to smackback_mode downgrades (lockout→alert, paranoid→alert, paranoid→lockout). Upgrades (alert→lockout, any→paranoid) can be applied immediately. Severity is reduced by Finding 3's fix: an attacker can downgrade mode but cannot disable monitoring entirely.

Verdict: Open. Partial mitigation via Finding 3. Mode downgrade via hub push remains possible.

Finding 5 — hub_controls_* Flags Never Delivered to Spokes — Spoke Locking Broken [Medium]

`Location: core/multisite-api.php — settings/push allowlist; smack-push-it.php push manifest`

The hub push manifest (smack-push-it.php, line 44–48) includes hub_controls_timezone, hub_controls_akismet, hub_controls_ai, hub_controls_smackback, hub_controls_comments, and hub_controls_email. These flags are intended to lock the corresponding settings on spokes so spoke admins cannot override hub-controlled values. However, none of the hub_controls_* keys are on the settings/push API allowlist in core/multisite-api.php. They are silently placed in the skipped[] array on every push. As a result, spokes never receive the lock flags and spoke admins can always change hub-controlled settings locally, defeating the entire hub-control mechanism.

Status: Open.
Recommended fix: Add hub_controls_timezone, hub_controls_akismet, hub_controls_ai, hub_controls_smackback, hub_controls_comments, and hub_controls_email to the allowlist. These are boolean flags (0/1) with no injection surface. Note: hub_controls_smackback on the spoke should additionally gate the smackback_mode change from Finding 4.

Verdict: Open. Spoke-side locking non-functional. hub_controls_* keys missing from API allowlist.

Finding 6 — AI Key Push Silently Fails — Keys Not on API Allowlist [Low]

`Location: core/multisite-api.php — settings/push allowlist; smack-push-it.php AI push group`

The AI push group in smack-push-it.php includes ai_provider, ai_key_claude, ai_key_gemini, and ai_key_openai. Only ai_training_policy is on the settings/push allowlist. The three key fields and the provider field are silently dropped on every push. The hub UI shows a PUSH TO ALL SPOKES button for AI settings that has never worked as intended. This is a functional bug rather than a security finding, but is documented here as it was discovered during the security review. Severity is Low because the failure mode is silent data loss rather than a vulnerability.

Status: Open.
Recommended fix: Add ai_provider, ai_key_claude, ai_key_gemini, ai_key_openai to the allowlist. Note that pushing API keys from hub to spoke means all spokes share the hub's AI credentials — this is the intended design for single-owner fleets but should be clearly documented.

Verdict: Open. Functional bug. AI key push has never succeeded; allowlist fix required.

Finding 7 — No Warning on Spoke Connection Page Re: Hub Authority [Low]

`Location: smack-multisite.php — spoke connection UI, ~line 713`

The page where a site operator connects their installation to a hub contains no warning about the authority the hub gains. Once connected, the hub can: push and lock settings, trigger spoke updates, install/reinstall skins, toggle maintenance mode, request SSO admin tokens, moderate comments, cross-post content, sync banlists, and (with Finding 3 now fixed) request SMACKBACK be disabled. An operator connecting to a hub they do not own would be granting an unknown party substantial administrative authority over their site. The current UI treats this as a technical configuration step with no trust framing.

Status: Open.
Recommended fix: Add a prominent warning block to the spoke connection flow stating that hub/spoke is designed for single-owner fleets only. Exact copy agreed with Sean: “Never connect to a hub you don’t control yourself. Hub/spoke is designed for the same owner who runs a fleet — it is not intended for individuals to merge sites. A hub has significant authority over this site.”

Verdict: Open. Warning copy agreed; not yet implemented in smack-multisite.php.

Finding 8 — SSO Endpoint Grants Hub Full Spoke Admin Access [Informational]

`Location: core/multisite-api.php — POST multisite/auth/sso-token`

The SSO endpoint issues a short-lived one-time token that allows the hub to open an authenticated admin session on any spoke without knowing the spoke’s admin password. This is correct and intended behaviour for a single-owner fleet. A compromised hub therefore gains full admin access to every connected spoke’s admin panel within five minutes of requesting SSO tokens. Tokens are 64 hex characters (32 random bytes), single-use, and expire after 300 seconds. There is no spoke-side notification when an SSO session is opened. This is noted as an audit confirmation: the implementation is sound and the risk is inherent to the single-owner hub/spoke design.

Verdict: Closed. By design. Risk inherent to single-owner topology; implementation is correct.

Session Summary

Three High-severity findings were identified and fixed in-session. The most critical (Finding 3) closes the attack chain by which a compromised hub could silence SMACKBACK before tampering with spoke files. Two Medium-severity items (Findings 4 and 5) remain open and are related: once hub_controls_* delivery is fixed (Finding 5), the smackback_mode gate from Finding 4 can be implemented on the spoke side. Finding 6 is a pre-existing functional bug with no security impact. Finding 7 is a UI gap with agreed copy. Finding 8 is by design.

#	Item	Priority
4	Gate SMACKBACK mode downgrades via hub push (pending confirmation)	High
5	Add hub_controls_* keys to settings/push API allowlist	High
6	Add ai_provider + ai_key_* to settings/push API allowlist	Medium
7	Add single-owner warning to spoke connection UI in smack-multisite.php	Medium