SNAPSMACK — SECURITY FINDING

SMACKBACK — Undetected Unknown Files & Scheduled-Verify Gap

Audit:
027

Date:
2026-06-19

Auditor:
Cowork session (Claude, Opus 4.8), for Sean

Files:
core/smackback.php, smack-admin.php, smack-back.php, cron-version-check.php, core/sidebar.php

Severity:
High (Finding A — integrity false-negative) / High (Finding B — scheduled verify never ran) / Low (Finding C — information disclosure)

Status:
Remediated this session; ships next release. Strict-mode rollout caveat (§5) + residual items (§6).

Summary

SMACKBACK is presented to operators as integrity monitoring that "catches FTP credential compromise before it becomes a bigger problem." Live testing — uploading an empty `stats.php` over SFTP to a production spoke — showed it did **not** raise a breach. Investigation found two independent High-severity gaps on the integrity-monitor surface: a detection blind spot (unknown/added files were invisible) and a long-standing fatal parse error that meant scheduled verification never ran at all. A Low-severity information-disclosure issue was also identified. The two detection gaps are remediated this session; the disclosure issue and several hardening items remain (§6). This document is written for full transparency: these were real gaps that shipped in prior releases.

Finding A — Integrity false-negative: unknown / non-baselined files not detected (High)

Severity: High • Status: Remediated this session

Description

`smackback_verify_all()` iterated only the baseline manifest (`snap_file_manifest`), checking each *known* file for a content change or deletion. It never compared the set of files actually on disk against the manifest. Any monitored-type file (PHP/CSS/JS) present on disk but absent from the baseline was therefore invisible — including the material threat this product claims to defend against: a newly dropped PHP webshell placed via an FTP/SFTP compromise. Content changes to *baselined* files were correctly caught via hash mismatch; **additions and replacements of non-baselined files were the blind spot.** Proven live: replacing a non-baselined `stats.php` with an empty file produced no breach.

Resolution

Strict unknown-file detection was added. After the manifest loop, `verify_all()` diffs `smackback_get_monitored_files()` — which already honours every `should_monitor()` exclusion (uploads, skins, vendor, etc.) — against the manifest path set; any monitored file on disk that is not in the baseline is flagged `unexpected` and counts as a breach. The check is guarded to skip when the manifest is empty (unarmed install) so it cannot false-flag the entire tree. The new bucket is threaded through `handle_breach()` (persisted in `breach_files`), the alert email, the network report, the breach page (shown as `UNEXPECTED` with a "remove or re-baseline" action — RESTORE is meaningless for a file that has no baseline), and the login / cron / manual callers. Treatment is **strict** (immediate breach) by owner decision: aggressive posture, duty of care to hosted users.

Finding B — Scheduled verification never ran: fatal parse error in cron-version-check.php (High)

Severity: High • Status: Remediated this session

Description

`cron-version-check.php` is the script a host's cron invokes; among other tasks it runs the scheduled SMACKBACK full verify. Its header docblock contained the cron example `0 */6 * * *`. The `*/` inside `*/6` prematurely closed the `/** ... */` comment, leaving the remainder of the header parsed as live code — a **fatal PHP parse error**. The file therefore never executed under cron. Net effect: SMACKBACK full verification ran **only** on admin login. On a site without frequent admin logins, integrity was effectively unmonitored between sessions — directly contradicting the scheduled-monitoring assurance. Surfaced by `php -l` during this session.

Resolution

The header was rewritten as a single valid docblock; the cron example is expressed as `0 0,6,12,18 * * *` to avoid any literal comment-closing sequence. Independently, scheduled verification was made host-independent: a new **mandatory verify interval** (`smackback_verify_interval_hours`, admin-selectable 1–24h, with no "off") now drives a due-based full verify from admin page loads as well as cron (`smackback_verify_due()` / `smackback_verify_interval_hours()` in `core/smackback.php`; `smack-admin.php` verifies when due and stamps the time on every run; cron is gated by the same check). The schedule now holds even where system cron is absent or misconfigured.

Finding C — Information disclosure: smackback-manifest.json in webroot (Low)

Severity: Low • Status: Open recommendation (operator + packaging)

Description

A populated `smackback-manifest.json` (the complete monitored-file inventory plus SHA-256 baselines and EOF signatures) is present in the live webroot. It is not a monitored extension, so the integrity scan ignores it; but if it is web-reachable it hands an attacker the full file map and hashes — useful reconnaissance for locating targets or crafting a tamper. The runtime baseline is the database table, so the JSON copy is a leftover artifact.

Recommendation

Deny web access to the file (an `.htaccess` / vhost rule) or do not retain it in the webroot after install. Not remediated in code this session — flagged for the operator and a packaging follow-up.

Posture improvements (not vulnerabilities)

• SMACKBACK was promoted to a first-class sidebar entry (“SMACKBACK Security”, `core/sidebar.php`), shown in both UI modes; previously it was reachable only via a button on the Configuration page. • The mandatory 1–24h verify interval (Finding B resolution) is itself a posture improvement: integrity verification can no longer be silently left unscheduled, and cannot be switched off — only its cadence is configurable.

5. Rollout caveat (read before deploying)

Strict unknown-file detection means any existing install carrying a *legitimate* unbaselined file (for example a legacy `stats.php`) will breach on its first post-deploy verify — and in LOCKOUT mode that locks the admin out. Recommended rollout: set the response mode to **ALERT** during the push, let each spoke report its `UNEXPECTED` list, review and remove genuine cruft (or re-initialise the baseline from disk if the disk is trusted), then return to **LOCKOUT**. Re-baseline absorbs whatever is currently on disk, so only do it after confirming the site is clean — otherwise it would bless an intruder file.

6. Residual / follow-up

1. **smackback-manifest.json webroot exposure** (Finding C) — deny access or stop shipping it to the webroot.
2. **Hub-controlled branch** of the SMACKBACK page does not yet expose the interval selector (defaults to 6h, still enforced).
3. **verify_quick** (public-pageload, mtime-only) does not detect new files by design — a per-request disk walk is too costly; unknown-file detection runs in full verify (login / cron / manual). Tighten via cron cadence; a throttled periodic scan is a possible future option.
4. **Optional one-click quarantine** for UNEXPECTED files on the breach page — deliberately not added without step-up (reauth) gating, since a web-delete action is itself attack surface.

This audit covers the unknown-file detection and scheduled-verify remediation only. Lint status at time of writing: core/smackback.php, smack-admin.php, smack-back.php, cron-version-check.php pass php -l; core/sidebar.php pending operator lint. Treatment and rollout decisions (strict mode; mandatory interval) were made by the owner this session.