SNAPSMACK — SECURITY FINDING

SMACKBACK — File Integrity Monitoring and EOF Sentinel Implementation

Date:
2026-05-22

Files:
core/smackback.php, smack-smackback.php, migrations/migrate-smackback.sql, tools/_build/build-release.php, tools/_build/build-skin-package.php, tools/_build/package-skin.php, smack-update.php, core/skin-registry.php, install.php, smack-admin.php, core/auth-smack.php, cron-version-check.php, database/schema/snapsmack_canonical.sql, and all 8 public page files

Severity:
Informational (Finding 1) / Low (Finding 2) / Informational (Finding 3)

Status:
Addressed in 0.7.170 “Wingback”

Summary

This audit documents the design and implementation of SMACKBACK (file integrity monitoring) and the integrated EOF Sentinel, both shipping in SnapSmack 0.7.170. Three areas are reviewed: (1) the gap between the marketing claim that SMACKBACK was live and the reality that it was not implemented; (2) the security properties and intentional design decisions in the 0.7.170 implementation; and (3) functionality deferred to Phase 2 where the marketing copy still claims it is present. No vulnerabilities were introduced. One pre-existing misrepresentation is resolved by this build.

Finding 1 — Vaporware: SMACKBACK Claimed as Live Feature Before Implementation

Severity: Informational • Status: Resolved in 0.7.170

Description

The smack-help.php admin documentation and the snapsmack.ca marketing page both described SMACKBACK as active functionality: “Layer 6 — SMACKBACK file integrity monitoring.” No such feature existed in any release prior to 0.7.170. The database tables (snap_file_manifest, snap_smackback_log), the core engine (core/smackback.php), and all integration hooks were absent.

This is a documentation and marketing accuracy issue, not a security vulnerability. A site owner who believed SMACKBACK was active might not take other integrity-monitoring precautions. It also created a false assurance in the threat model for the snapsmack.ca hosted install.

Resolution

0.7.170 delivers the full SMACKBACK implementation as described. The smack-help.php documentation now accurately reflects operational behaviour. The snapsmack.ca marketing page requires a separate FTP update (wotcha.php / index.php — deferred, see swapfile).

Finding 2 — Security Design: BREACH Page, Baseline Trust, and Re-initialise Risk

Severity: Low • Status: Mitigated by design; one residual risk documented

2a. BREACH Page — All CSS Inline (Intentional)

The BREACH lockout page rendered by smackback_render_breach() emits a fully self-contained HTML page with all CSS inline in the PHP string. No external stylesheet is loaded. This is intentional: if an attacker tampers with a CSS file and SMACKBACK detects the breach, the BREACH page must still render correctly even if every external asset on disk has been replaced or deleted. A BREACH page that depends on admin.css to display correctly could be neutralised by deleting admin.css.

2b. Baseline from Signed Package (Intentional)

Hash baselines are set from smackback-manifest.json embedded in the release ZIP. The ZIP is covered by the Ed25519 signature verified before extraction. This means the baseline hashes carry the same trust level as the package signature — an attacker who can forge the signature can also forge the manifest. No separate trust channel is needed.

Fresh installs use smackback_init_from_disk() because no ZIP is retained post-extraction. This is a weaker baseline (files just came from a verified package, but no hash comparison is performed at this moment). An attacker who intercepts the download and replaces files before installation would have those files blessed into the baseline. This is the same trust assumption as the install itself — if the package was compromised in transit, every file is already wrong.

2c. Re-initialise Baseline Risk (Residual)

The Re-initialise Baseline from Disk button in the SMACKBACK admin panel re-hashes all monitored files and writes the results as the new baseline. If clicked during an active breach, it would bless any tampered files as the new expected state, silently clearing the breach.

Mitigation: The admin page displays a persistent warning: “Do not use this if a breach is active — it would bless the tampered files.” The confirmation dialog repeats this warning. No hard programmatic block is in place. The assumption is that an admin who has just been told their files are tampered and clicks through two warnings to re-baseline is making a deliberate choice. The incident is logged to snap_smackback_log regardless.

> *A future hardening option is to require the admin to explicitly mark the breach as a false positive before re-initialise is permitted, rather than relying on the warning. Deferred to a future release.*

Finding 3 — Phase 2 Deferral: Cross-Spoke Correlation Not Implemented

Severity: Informational • Status: Partially unresolved (marketing copy)

Description

The snapsmack.ca marketing page (and smack-help.php prior to 0.7.170) claims cross-correlated tamper reports across hub/spoke networks as a SMACKBACK capability. This feature — where spokes report breach findings to the hub, the hub aggregates across the fleet, and fleet-wide anomaly detection identifies coordinated attacks — is not implemented in 0.7.170.

The paranoid response mode includes a no-op stub in smackback_handle_breach() with a comment: "// Phase 2: smackback_hub_report($tampered, $missing); Stub — intentionally a no-op until Phase 2 build." The hook is wired, the implementation is absent.

Resolution

smack-help.php updated in 0.7.170: the Paranoid mode description reads "Lockout + hub breach reporting (Phase 2 — coming in a future release)". The snapsmack.ca marketing page (wotcha.php / index.php) needs to be updated to move the cross-correlation claim to a 'planned' section. This FTP update is pending (see swapfile.md).

EOF Sentinel — Design Rationale

The EOF Sentinel is documented here because the signal matrix represents a deliberate security design choice that affects how breach alerts are interpreted.

Background: File integrity monitoring based on SHA-256 alone cannot distinguish between three meaningfully different failure modes: active tampering (an attacker modified the file), truncation (a write failure left the file shorter than intended), and corruption (a filesystem fault zeroed part of the file). All three produce a hash mismatch. Without the EOF signal, every mismatch is reported identically and the admin must investigate to determine the cause.

Signal Matrix

Hash	EOF last line	Null bytes	Classification	Typical cause
Match	—	—	OK	No change
Mismatch	Matches baseline	No	TAMPERED	Active modification
Mismatch	Differs	No	TRUNCATED	Write failure, partial transfer
Mismatch	—	Yes	CORRUPTED	Filesystem fault, atomic write fail
N/A	N/A	N/A	MISSING	File deleted

Implementation: At baseline time, smackback_get_eof_signature() reads the last 1024 bytes of each monitored file, checks for null bytes (returns 'NULL_BYTES' sentinel if found), then returns the last

non-empty line trimmed to 512 characters. This value is stored in snap_file_manifest.eof_signature and embedded in smackback-manifest.json in every release and skin ZIP. At verify time, smackback_classify_mismatch() is called only on hash-mismatched files — hash-matching files skip the EOF check entirely.

> *This audit covers the 0.7.170 implementation only. Phase 2 (hub/spoke correlation) will require a separate audit when implemented. The Re-initialise Baseline hardening item is tracked as an open recommendation.*