Security Audit — 029 "Automatic File-Deletion Attack Surface"

Date: 2026-06-20 **Auditor:** Cowork session (Claude, Opus 4.8), for Sean **Severity:** High (design) — a fleet-wide remote deletion primitive. Caught and removed pre-release; never shipped in the remote form. **Status:** Remediated this session. All automatic file deletion removed; replaced with detect-and-report.

1. Summary

While fixing orphan-file accumulation (stale files left by upstream renames, which false-trip SMACKBACK strict detection — see secaudit 028), the existing `updater_remove_deprecated_files()` facility was extended **into the remote/hub update path** so a hub-pushed update would `@unlink()` listed files on a spoke.

That was the wrong call, and Sean correctly rejected it: **neither the operator (via the hub) nor an attacker should be able to alter a user's filesystem.** An automatic, list-driven file-removal path is a deletion primitive — useful to the operator, and a fleet-wide `rm` to anyone who compromises the hub or slips an entry into a package/list. The convenience of auto-cleanup does not justify the attack surface. Recording the mistake openly rather than quietly reverting it.

2. Finding — list-driven auto-deletion (High, design)

- `core/updater.php` defined `updater_remove_deprecated_files()`, which `@unlink()`'d any file in `UPDATER_DEPRECATED_FILES` still present after an update.
- The **local** updater (`smack-update.php`, two call sites) already invoked it.
- This session it was **also wired into the remote/hub path** (`core/multisite-api.php`), meaning a hub-triggered update silently deleted files on every spoke.

Threat model: - **Hub compromise -> fleet-wide deletion.** With the remote wiring, an attacker controlling the hub (or the update trigger) could delete arbitrary listed paths across every spoke. The remote path runs unattended — no local consent. - **Package/list tampering -> deletion.** Any code path that deletes based on a list is only as safe as the integrity of that list and the package carrying it. - **Consent.** It is not the operator's right to silently alter a user's install filesystem (cf. the "no silent install changes" and "hub has no authority over a spoke" principles, secaudit-adjacent to the SMACKBACK integrity model).

3. Remediation

The CMS now **never deletes a user's files.** Deletion is replaced by detection:

- `core/multisite-api.php` — remote auto-delete **removed**; an explicit comment forbids re-adding deletion to the remote path.
- `core/updater.php` — `updater_remove_deprecated_files()` -> `updater_detect_orphan_files()`: no `@unlink`, returns the listed orphans that are present on disk. The `UPDATER_DEPRECATED_FILES` list is retained as a **reference/annotation** only.
- `smack-update.php` — both call sites now **report** lingering orphans in the update log ("Known orphans present — remove manually: ...") instead of deleting.

Orphan handling end-state: SMACKBACK already detects orphans as UNEXPECTED; the deprecated-file list lets the breach review **annotate** which UNEXPECTED entries are known renames ("safe to remove"); a **local admin removes them by hand**. No automated deletion exists anywhere in the codebase.

4. Residuals / follow-ups

- Surface the "known orphan — safe to remove" annotation in the SMACKBACK breach review UI (ties into the integrity-model re-bless work, task #7).
- Directory cruft (`smack-central/` on a spoke, `.githooks/`, `outputs/`) is likewise **detected, never auto-removed**; `smack-central/` is legit on the hub so any future annotation must be role-aware. Removal stays manual.

- One-time: audit git history for other un-listed renames/deletes to make the detection/annotation reference complete.