SECAUDIT 044 - Hub read-side: shared profile store (GYSS), shared Gemini prompts, launcher exe-discovery

SECURITY AUDIT / PUBLIC REMEDIATION RECORD

Field	Value
Date	2026-08-13
Scope	The Hub read-side wiring: GYSS shared-profile store (`tools/gyss/src/scripts/profiles.js`, `paths.js siteKey()`); shared Gemini prompts (`tools/_shared/snap_prompts.py`, `snap_home.prompts_dir()`, `tools/sybu/config.py` + `tools/coldsnap/config.py`); THE HUB launcher exe-discovery (`tools/hub/main.py _find_exe`/ROSTER).
Baseline	`dev` @ `577f6440`; fixes below applied on top (THE HUB 0.1.1->0.1.2, SYBU 0.1.43->0.1.44, COLD SNAP 0.1.4->0.1.5).
Status	**Two MEDIUM findings FIXED, including the architectural jail-root residual. Four LOW: two fixed, two documented as negligible/pre-existing. Source-level release gate satisfied.**
Method	Author self-audit, then an INDEPENDENT adversarial reviewer tasked to *refute* the self-audit's "no HIGH/MEDIUM" claim. It did - Finding 1 below was under-rated as LOW in the first pass and is corrected here. That correction is why the independent pass was run.
Positive controls	Atomic writes (`os.replace`/tmp); unreadable prompts store preserved as `.corrupt` (and no longer overwritten on a second corruption); profile/prompt filenames pass `site_key()`/`siteKey()` -> `[a-z0-9.-]`, strip leading/trailing `.-`, throw on empty (fail-closed, verified non-traversable both languages); GYSS FS additionally jailed by the Rust `resolve_in_root` (verified sound against `../`UNC/`\\?\`/sibling-prefix); launcher uses list-form `subprocess.Popen([path])` (no shell/arg injection); migrations marker-guarded + now per-file-stamped, idempotent, never delete legacy data; no SQL, no `eval`, no new network endpoints.
Disclosure	No exploitation known. Surface is offline desktop tooling running as the owner. The sharp finding (1) needs either a GYSS webview compromise (a threat the project already treats as reachable, `src-tauri/src/lib.rs:82-84`) OR a weak-ACL local user able to write under `C:\snapsmack`.

1. Executive result

The data-plane of this change (shared profile store, shared prompt store, migrations) is sound: no traversal, atomic + recoverable writes, graceful handling of hostile/corrupt local files, and no new network/SQL/shell. The independent pass confirmed the traversal jail and credential handling hold and that GYSS's profile migration is well-hardened.

The real issue was in **THE HUB launcher**, not the stores. The first (author) pass rated it LOW; the independent pass correctly escalated it to **MEDIUM**: the launcher was globbing `*.exe` from directories INSIDE `C:\snapsmack`, which is the GYSS write-jail root. The jail's entire safety argument (SECAUDIT 039) is that a compromised GYSS webview may write anywhere under the root but cannot reach an executable that something will run. Placing auto-launched exe locations inside that root breaks that conclusion. **Fixed** (§3). The narrower, partly pre-existing exact-path residual was then closed by restricting every GYSS file command to its two actual data roots (§4).

2. Trust-boundary map

```
owner at the keyboard (owner's Windows machine)
-> THE HUB picks an exe from ROSTER candidates, launches it (list-form, no shell)
-> SYBU / COLD SNAP read/modify/write shared_library/prompts/gemini_prompts.json
-> GYSS / SYBU read/modify/write shared_library/profiles/<site_key>.json (GYSS FS jailed)

compromised GYSS webview (reachable per lib.rs:82-84; app fs commands are ungated)
-> download_to / write_file can write ANY bytes to ANY path under C:\snapsmack
=> before fix: plant C:\snapsmack\ohsnap\evil.exe -> Hub shows OH SNAP "installed" -> click -> RCE
=> after fix: wildcard candidates inside the root are refused (Finding 1)
=> after closure: exact tool targets are outside the GYSS data-root allowlist (Finding 2)

low-priv local user (only where C:\snapsmack\* dirs inherit user-writable ACLs)
-> same plant/overwrite without any webview compromise (local priv-esc)

network -> NONE introduced by this diff.
```

3. Finding 1 - launcher auto-ran a wildcard-matched exe from inside the GYSS write-jail (MEDIUM, FIXED)

`tools/hub/main.py` added wildcard ROSTER candidates rooted under `C:\snapsmack` (`C:\snapsmack\ohsnap\*.exe`, `C:\snapsmack\suyb\smackupyourbackup*.exe`) and made `_find_exe` return the newest match. `C:\snapsmack` is the GYSS jail root (`shared_root()`), and GYSS's own fs commands (`download_to`, `write_file`) are ungated and permit writing arbitrary bytes anywhere under it.

Exploit chain: a compromised GYSS webview calls `download_to` with `path="C:\\snapsmack\\ohsnap\\payload.exe"` (jail allows it, even `create_dir_all`s the folder). OH SNAP is not really installed, so that planted file becomes its ONLY match - the Hub then displays OH SNAP as installed/enabled and one click runs the payload as the user. For SUYB, newest-mtime-wins means a planted `smackupyourbackup-evil.exe` even beats a legitimately-installed versioned exe. A webview compromise is not required in the weak-ACL local-user variant.

Fix (applied, THE HUB 0.1.2):

- Removed every wildcard candidate that resolves inside `C:\snapsmack`

(`ohsnap\*.exe`, `suyb\smackupyourbackup*.exe`) and every speculative inside-root path (`snapsmack\gyss\...`, `snapsmack\suyb\suyb.exe`). Inside the root the roster now lists ONLY the two real shared-layout installs as EXACT paths (SYBU, COLD SNAP). Wildcards remain only for out-of-jail legacy dirs (`C:\SUYB`, `C:\SmackUpYourBackup`).

- Added a defense-in-depth guard in `_find_exe`: a wildcard candidate whose directory

resolves inside `_shared_root()` is refused outright, so a future roster edit cannot reintroduce the class.

- Verified: real tools still resolve (SYBU/GYSS/COLD SNAP/SUYB); a planted

`C:\snapsmack\ohsnap\payload.exe` yields `None`; out-of-jail `C:\SUYB\*.exe` still resolves.

4. Finding 2 - exact-path tool exes shared the broad GYSS jail root (MEDIUM, FIXED)

Even after Finding 1, the two exact launch targets `C:\snapsmack\sybu\sybu.exe` and `C:\snapsmack\coldsnap\coldsnap.exe` sit inside the GYSS-writable root. A compromised webview could OVERWRITE one of them (when not locked) with a payload; the Hub would then launch it. This is narrower than Finding 1 (the attacker must target the exact exe of a tool the owner will actually launch, and a running tool's exe is locked), and it is PARTLY PRE-EXISTING - the Hub shipped launching `C:\snapsmack\sybu\sybu.exe` in v0.1.0; this diff only added COLD SNAP's equivalent by installing it under the shared layout.

Root cause is architectural: the `C:\snapsmack\<tool>` shared install layout collides with the GYSS jail root, so "tool exes" and "GYSS-writable data" share a tree.

Fix (applied): `resolve_in_root` now authorizes only paths beneath `C:\snapsmack\shared_library` and `C:\snapsmack\config_files\gyss`, while `shared_home` continues to return the common root for correct UI path construction. The restriction covers read, write, list, download, delete, and catalog commands. Component-aware path checks and the existing `..` rejection prevent sibling-prefix or traversal bypass. A Rust regression test proves both legitimate data roots work and rejects SYBU/COLD SNAP executables, a lookalike `shared_library-evil` directory, and a traversal attempt. `cargo test --lib` passes.

5. LOW findings

- **L1 - prompts migration could resurrect a deleted preset (FIXED).** `config.py`

`_migrate_prompts_once` guarded only with a single best-effort dir marker; if that write failed, a later launch re-folded the legacy `gemini_prompts.json`, re-adding presets deleted from the shared store. Fixed by adding a DURABLE per-file `<legacy>.migrated` stamp (mirroring GYSS's profile-migration hardening), checked and written in both SYBU and COLD SNAP.

- **L2 -** `.corrupt` **backup was single-slot (FIXED).** `snap_prompts.save` overwrote an

existing `.corrupt` on a second corruption. Now it preserves the first backup (`os.replace` only when `.corrupt` is absent).

- **L3 -** `siteKey()` **vs** `site_key()` **diverge on contrived inputs (DOCUMENTED, negligible).**

E.g. a non-numeric port `example.com:abc`. Confirmed by the independent pass to be an interop mismatch only - NOT a traversal - and unreachable for real site URLs (numeric port / none), which were verified identical incl. IDN. Left as-is to avoid an IPv6 handling regression for a non-exploitable contrived case.

- **L4 - GYSS Rust** `write_file` **is non-atomic (DOCUMENTED, pre-existing).**

`lib.rs:121-128` does a direct `std::fs::write` (no tmp+rename) unlike the Python peer, so a crash mid-write can leave a torn profile (readers skip it). This remediation changes only the authorization boundary in `lib.rs`; atomic-write work remains a separately scoped reliability improvement.

6. Verification performed

- Executable harnesses (all green, re-run after fixes): GYSS↔Python profile interop

(UTF-8 + IDN, siteKey parity); GYSS migration incl. deleted-profile resurrection cases; prompt concurrency (stale-sibling add not clobbered) + corrupt-store recovery; built-in vs user-override preset semantics.

- Hardened launcher tested against the real machine (all real tools resolve) AND against a

planted `C:\snapsmack\ohsnap\payload.exe` (refused -> None).

- `node --check` on the JS; `py_compile` on the Python; EOF markers on all changed files.
- GYSS Rust unit test for the narrowed data-root jail: one test passed, including

positive data paths and negative executable, sibling-prefix and traversal paths.

- Two independent agent reviews: a functional pass (5 findings, all fixed earlier) and this

security pass (Finding 1 escalation + the LOW set).

- Not exercised: a real multi-user/weak-ACL host, or an actual GYSS webview compromise;

Findings 1/2 are reasoned from the code + the SECAUDIT 039 jail model.

7. Release gate

Desktop tools ship by building from the checkout (`C:\snapsmack\<tool>`, `C:\GYSS`), not the core updater, so no fleet rollout is tied to this audit. Source-level gate: **satisfied** - Findings 1 and 2 (MEDIUM) are fixed; LOWs are fixed or documented. The launcher cannot discover wildcard executables inside the shared root, and GYSS file commands cannot access the sibling tool-install directories even by exact path. The source-level release gate is satisfied.