SNAPSMACK — SECURITY FINDING

login.php Orphaned at Predictable URL (Auth Model Violation)

Date: 2026-05-19

File: login.php

Severity: High

Status: Fixed in 0.7.155

Summary

login.php was a fully functional, duplicate login page accessible at a predictable URL. SnapSmack's security model routes admin login through a configurable slug (default: /snap-in) precisely to prevent bots and attackers from finding the login endpoint. login.php existed at a guessable path, circumventing this protection entirely. The file contained complete authentication logic including password verification and session creation.

Background

At some point in SnapSmack's history, the login system was refactored:

snap-in.php became the canonical login handler, accessed only via a configurable URL slug (set in Admin → Configuration → Security, default /snap-in). Direct access to snap-in.php returns a 403.

login.php was the original login page from before the slug model was introduced. It was never deleted during this refactor. It continued to ship in every release and remained fully operational on every install.

Affected Code

login.php contained complete, working authentication logic:

```
if ($user && password_verify($pass_input, $user['password_hash'])) {
    // ... session creation
    $_SESSION['user_login'] = $user['username'];
    // ...
}
```

The file was accessible at https://example.com/login.php — a URL that every credential scanner in existence probes automatically.

Attack Scenario

1. Automated scanner probes https://target.com/login.php — a standard path in every credential-stuffing and brute-force toolkit.

2. Scanner finds a functional login form (not a 404 or 403).

3. Scanner runs credential list against login.php.

4. If credentials match: session created, attacker authenticated as admin.

5. The configurable login slug (/snap-in or custom value) provided zero protection because login.php bypassed it entirely.

The .htaccess FilesMatch block on established installs blocked direct web access to login.php — but this was a band-aid. The file still shipped, still existed on disk, and the block depended on .htaccess being present and unmodified. On a partial install or a server where .htaccess directives are ignored, login.php was directly accessible.

Additional Finding — Duplicate Authentication Logic

login.php and snap-in.php contained independently maintained, duplicate authentication code. Any security fix applied to one (rate limiting, TOTP enforcement, session hardening) required a matching fix in the other. This duplication was never documented, meaning fixes applied to snap-in.php may not have been backported to login.php.

Fix

login.php deleted from the codebase in 0.7.155.

All references updated to point to snap-in (recovery emails, password reset back-links, SSO fallback, post-install link, multisite SSO manual login link).

login.php added to UPDATER_DEPRECATED_FILES in core/updater.php — automatically removed from existing installs on update.

.htaccess FilesMatch block updated to remove the now-unnecessary login entry.

Notes

Any install on 0.7.154 or earlier that has not yet applied the 0.7.155 update has login.php present. The updater removes it automatically on update.

Admins who have customised their login slug should verify that login.php no longer exists on their server after updating.