SNAPSMACK — SECURITY FINDING

install.php Step 5 Missing POST Guard

Date: 2026-05-18

File: install.php

Severity: Medium

Status: Open

Summary

Step 5 of the installer (the final write step — generates core/db.php, core/constants.php, seeds settings, creates directories, and self-deletes) has no $_SERVER['REQUEST_METHOD'] === 'POST' guard. Every other step (2, 3, 4) includes this check. The omission allows the step to be triggered via a direct POST request without going through the preceding credential steps.

Affected Code

install.php line 996:

```
// VULNERABLE — no POST check
if ($step === 5 && empty($errors)) {
```

All other steps use the correct pattern:

```
if ($step === 2 && $_SERVER['REQUEST_METHOD'] === 'POST' && empty($errors)) {
if ($step === 3 && $_SERVER['REQUEST_METHOD'] === 'POST' && empty($errors)) {
if ($step === 4 && $_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['admin_user']) && empty($errors)) {
```

Attack Scenario

1. Attacker navigates to install.php on a fresh (not yet installed) server, establishing a session and receiving a valid CSRF token.

2. Attacker POSTs directly to install.php with step=5 and the valid CSRF token, bypassing steps 2–4.

3. Step 5 executes with empty $_SESSION values for db_host, db_name, db_user, db_pass.

4. core/db.php is written with null/empty credentials (via var_export(null) → literal NULL in the PHP file). The file is syntactically valid but produces a DB connection error on every page load.

5. Step 5 calls unlink(__FILE__) — the installer self-deletes.

Result: the site is broken, the installer is gone, and recovery requires FTP access to manually replace core/db.php.

The safety lock (lines 60–143) prevents this attack if SnapSmack is already installed (it checks for a populated snap_settings table). The window is a fresh/uninitialised server where core/db.php does not yet exist.

Fix

Add the missing POST check:

```
if ($step === 5 && $_SERVER['REQUEST_METHOD'] === 'POST' && empty($errors)) {
```

Additional Finding — db_prefix Field Ignored (UX, not security)

The database configuration form (step 2) displays an editable Table Prefix field but the step 2 handler hardcodes $db_prefix = 'snap_' and discards the user's input. Not a security issue, but confusing: users who change the prefix will have it silently overridden. Either remove the field from the form, mark it read-only, or honour the submitted value.

Notes

The CSRF token check (lines 169–174) is a partial mitigant — an attacker must first obtain a valid token by visiting the page in the same session. This is trivial to do before a site is configured.

The safety lock provides no protection on a fresh server.

Impact is limited to availability (broken site) rather than data exposure, since no database exists at the time of attack. However, combined with social engineering ("click this link"), it could be used to sabotage a competing install attempt.