

WordPress Rebuild Protocol / Process

Use this protocol when a WordPress site is confirmed compromised and the decision is to rebuild it clean instead of disinfecting it in place.

“ **Production warning.** Every phase below changes a live site. Get approval before you suspend an account, change DNS, delete files, or rotate a credential.

Rule zero

A rebuild is not a cleanup.

A rebuild gives a clean *filesystem*. It does not close the hole the attacker came through, and it does not remove persistence that lives outside the document root. Rebuilding without finding the entry vector gets the new site infected the same way.

Answer three questions before you build anything:

- 1. **Entry vector** — how did the attacker get in?
- 2. **Persistence** — how do they get back in?
- 3. **Payload** — what did they do once inside?

If you cannot answer one of them, write "UNKNOWN". Do not invent an answer, and do not treat a timestamp as proof.

Decide: clean in place, or rebuild

Rebuild the account	Clean in place
Malware is in more than one place (core, plugins, uploads, database)	One known payload, one known entry vector, small blast radius

Rebuild the account	Clean in place
Persistence reaches outside the document root (cron, SSH keys, mail filters, <code>.user.ini</code>)	No sign of persistence outside the document root
The site ran a nulled or abandoned plugin or theme	Every plugin and theme comes from a paid or official source
Core checksums fail and you cannot explain why	Core verifies clean
The infection is older than your oldest good backup	You have a verified clean backup from before the first indicator
You cannot determine the full extent	You can state the extent with evidence

When you cannot decide, rebuild. A rebuild costs hours. A missed backdoor costs the client's domain reputation.

Rebuild into a NEW hosting account, not over the old one. A new account gives you a clean home directory, a clean crontab, clean SSH keys, clean mail filters, and a clean database user. Reusing the account keeps every one of those.

Phase 0 — Contain

Do this before anything else. The attacker may still be active.

1. Take the site offline or put it behind a maintenance page. Suspending the account stops code execution.
2. Record the current time. Every later timestamp is measured against it.
3. Do **not** delete anything yet.

Phase 1 — Preserve evidence

You cannot investigate what you have already overwritten. Capture first, look second.

Capture, outside the account home, to a location only an administrator can reach:

Artifact	Why
Full archive of the home directory	The filesystem as found, including files you have not seen yet
A dump of every database on the account	The database as found
Access and error logs	Logs rotate. They are the only record of the entry vector

Artifact	Why
The crontab and any system-level cron entries for the account	Persistence lives here
The SSH <code>authorized_keys</code> file	Persistence lives here
Email filters, forwarders, FTP accounts, API tokens	Persistence lives here
File creation and modification times across the document root	Establishes a timeline

Name the files so a later reader knows they hold malware, and include the date. Keep the evidence. Do not use it as a restore source without cleaning it first.

Phase 2 — Investigate

At minimum, check each of these:

Filesystem

- Verify core and plugin file integrity against known-good checksums.
- Search for executable files where none belong, especially inside the uploads directory (`.php`, `.phtml`, `.phar`, an unexpected `.htaccess`).
- Search every file in the uploads directory for a PHP opening tag, regardless of its extension.
- List any cache or object drop-in files (`object-cache.php`, `advanced-cache.php`, `db.php`) and any must-use plugin — confirm each one was installed on purpose.
- Sort files by creation time and review what appeared during the incident window. Confirm your tool reports creation time correctly; some report zero results instead of an error when it cannot.

Database

- Search option values for `eval()`, `base64_decode`, `gzinflate`, `assert()`, and `str_rot13`.
- List every user account, sorted by ID, and confirm you can name each one.
- Check user-capability metadata for an entry using the wrong table prefix — a common way to hide an administrator.
- Review the stored cron option for a hook you cannot explain.

Logs

Correlate the earliest malicious file's creation time with the access log. Look for a request to an admin-ajax, XML-RPC, upload, or plugin route from one IP address, seconds before the file appeared.

Outside the document root

Check cron, SSH keys, mail filters, any auto-prepend configuration, other domains or subdomains in the same account, and other accounts on the same server that share an owner or a plugin.

Record every finding before you continue.

Phase 3 — Inventory what you will rebuild

You need this list before you destroy anything.

- Core version.
- Every plugin and theme, with status and version.
- Every user account, with role.
- The site URL and home URL.
- Database size by table, to spot an oversized or unfamiliar table.

Sort every plugin and theme into three groups:

- **Free, from the official plugin directory** — reinstall from source, at the current version.
- **Premium, licensed** — you need the license and the vendor's own download. Get these before the cutover, not after.
- **Nulled, abandoned, or unknown origin** — do not reinstall. Replace the function or remove it. This group is where the entry vector usually lives.

Also record: theme customizations, custom post types, form definitions, redirects, and DNS records.

Phase 4 — Build the clean target

1. Create a **new hosting account**. Use a new account name, a new password, and a new database user.
2. Install WordPress core from the official source, at the current stable version.
3. Install the free plugins from source. Install the premium plugins from the vendor, using your own license. Install nothing from the old account.
4. Install the theme from source. For a custom or child theme, take it from version control if it exists. If it does not, diff every file against a clean copy of its parent and read every

remaining line before you use it.

Confirm the base is clean before you add content: re-run the core and plugin integrity checks from Phase 2 against the new install.

Phase 5 — Move the content

This is the phase where a rebuild fails. **Do not copy the plugins, themes, or content directory wholesale.** The malware lives there.

Move	Never move
The database, after Phase 6 cleaning	Core files
Uploads, with executables stripped	Plugin files — reinstall from source
A version-controlled custom theme	Theme files — reinstall from source
Exported form and redirect definitions	Must-use plugins — rebuild by hand
	The configuration file, <code>.htaccess</code> , and any auto-prepend file — write new ones
	Cache or object drop-in files

Two ways to move content. Pick by size and by trust:

- **A content export and import** — safest. It carries posts, pages, terms, and users, and it drops everything else. Custom post types fail to import until the plugin that registers them is active, so install plugins first.
- **A cleaned database import** — needed when the site has plugin tables you cannot export. Clean the dump in Phase 6 before you import it.

For uploads, transfer with executables and unexpected `.htaccess` files excluded, then verify the destination holds none.

Phase 6 — Clean the database

Scanning is not cleaning. Work on the dump or on the new database, and record every change.

- Delete every administrator you cannot name. Check user-capability metadata for a wrong-prefix key.
- Remove options whose value contains `eval()`, `base64_decode`, `gzinflate`, `assert()`, or `str_rot13`. Check auto-loaded options first: they run on every request.

- Empty the stored cron option and let the platform rebuild it. Before you delete a hook, confirm no active plugin still registers it — a tool that checks only for a registered action can report a false orphan.
 - Remove injected script tags, iframes, and hidden link blocks from post, meta, option, and link tables.
 - Remove spam posts and spam terms: casino, pharma, loan, crypto, replica.
 - Reset the site URL and home URL to the correct domain.
 - Check the active-plugins option for a plugin directory that does not exist.
-

Phase 7 — Rotate every credential

Password rotation for site users is one item on this list, not the list.

Rotate
Site administrator passwords
Site secret keys and salts — this invalidates every session
Database user password
Hosting account password
FTP accounts — delete every one you cannot name
SSH keys — start from an empty authorized-keys file on the new account
API keys stored in the site — mail, payment gateway, storage, analytics
Any password the client reused elsewhere — ask. A stolen password is a
second attack vector
Control-panel or reseller credentials — only if evidence shows
higher-level access

Turn on two-factor authentication for every administrator that supports it.

Phase 8 — Harden

- Set file ownership to the account user only. Confirm no file remains owned by another user, especially `root`.
- Restrict the configuration file to the account user.
- Disable in-dashboard file editing.
- Block code execution inside the uploads directory.

- Directories at a non-writable-by-others permission, files the same. Nothing world-writable.
 - Turn on automatic background updates for core and for plugins the client does not customize.
 - Confirm the SSL certificate covers the domain, and that renewal notifications are on.
-

Phase 9 — Verify before you call it clean

Every check must pass. One failure sends you back a phase.

- ☐ Core integrity check passes.
 - ☐ Plugin integrity check passes for every plugin with a public checksum.
 - ☐ No executable file in the uploads directory.
 - ☐ No `eval`, `base64_decode`, or `gzinflate` in the database.
 - ☐ No administrator you cannot name.
 - ☐ Cron holds only hooks that a live plugin registers.
 - ☐ The web-server rewrite rules hold only rules you wrote.
 - ☐ The must-use plugin directory holds only what you put there.
 - ☐ The site renders correctly, forms send, and payments work.
 - ☐ A search-engine console shows no security issue and no spam page in the index. Request removal of any spam URL found.
 - ☐ An external scanner reports the site clean.
 - ☐ The entry vector is closed, and you can say how.
-

Phase 10 — Cut over

1. Lower the DNS time-to-live 24 hours before the cutover.
2. Test the new site by IP address or a temporary hostname first.
3. Change the DNS records. Wait for propagation.
4. Issue the SSL certificate after DNS resolves to the new host.
5. Re-point any uptime monitor at the new host.
6. **Suspend** the old account. Do not terminate it right away. Keep it for at least seven days, then terminate. Keep the evidence archive longer.

Phase 11 — Watch, then close out

- Read the access and error logs daily for the first 72 hours. Look for the same IP address, the same endpoint, and the same request pattern as before.
- Re-scan after seven days. A backdoor that survives shows itself when the attacker returns.
- Write an incident record: indicator, path, IP address, timestamp, log evidence, entry vector, persistence, payload, remediation, confidence.
- Write a rebuild report for the account.
- Tell the client what was taken, what was rotated, and what they must do.

Revision #1

Created 2025-04-07 13:14:58 UTC by avelica.ws@gmail.com

Updated 2026-09-21 15:21:10 UTC by avelica.ws@gmail.com