mySites.guru

Ninja Forms File Uploads CVE-2026-0740: The AJAX Pattern Strikes Again

CVE-2026-0740 is a CVSS 9.8 unauthenticated RCE in the Ninja Forms File Uploads AJAX handler, affecting around 50,000 WordPress sites. Here is how the flaw works, why the first patch failed, and how to find vulnerable sites fast.

Phil E. Taylor | 7 April 2026

The vulnerable function in CVE-2026-0740, disclosed by Wordfence on April 6, is called `NF_FU_AJAX_Controllers_Uploads::handle_upload`. That class name tells you the entire story before you read a line of code. A WordPress plugin registered an AJAX handler for file uploads, missed a validation step, and shipped an unauthenticated remote code execution vulnerability to around **50,000 sites** running Ninja Forms File Uploads. CVSS 9.8.

This is the fifth AJAX-handler-with-missing-checks CVE we have written about this year. That is not a coincidence, it is a pattern. We dug into the pattern in depth last week: WordPress's `admin-ajax.php` and Joomla's `com_ajax` were built as lightweight utility endpoints where the framework does almost nothing and expects plugin developers to handle authentication, authorization, and input validation themselves. When they don't - or when they check the obvious things and miss the subtle ones - you get CVE-2026-0740, and CVE-2026-21628 (Astroid Framework), and CVE-2026-21627 (Novarain Framework), and CVE-2026-3098 (Smart Slider 3), and CVE-2025-8489 (King Addons for Elementor), and the next one that will land before the month is out.

Ninja Forms is interesting because the developers actually did check things. That is what makes this one different from the Astroid or Novarain pattern where the handler had no real checks at all. Ninja Forms checked a nonce, validated the source file type, and required form and field IDs. They even shipped a patch when Wordfence reported the issue. The patch did not fix it - 3.3.25 was a partial patch, and only 3.3.27 closes the hole.

If you manage WordPress sites running Ninja Forms with the File Uploads add-on, stop reading and check your versions. Then come back.

What are the details of CVE-2026-0740?

Detail	Value
CVE	CVE-2026-0740
CVSS	9.8 Critical
Type	Unauthenticated Arbitrary File Upload
Vulnerable function	`NF_FU_AJAX_Controllers_Uploads::handle_upload`
Root cause	No destination filename validation or sanitization
Affected versions	<= 3.3.26
Partial patch	3.3.25 (February 10, 2026) - still vulnerable
Full patch	**3.3.27** (March 19, 2026)
Auth required	No - unauthenticated
Publicly disclosed	April 6, 2026
Researcher	Sélim Lanouar (whattheslime), via Wordfence Bug Bounty Program
Active installs	Around 50,000

How does the Ninja Forms File Uploads AJAX vulnerability work?

The vulnerable function is `NF_FU_AJAX_Controllers_Uploads::handle_upload`. It's the AJAX handler Ninja Forms registers via `admin-ajax.php` so visitors can upload files through a contact form. Looking at the code Wordfence published in their technical analysis, it **does** do several things right:

- It checks a nonce via `check_ajax_referer` to prevent simple CSRF
- It validates the source filename's extension against a blacklist in a `_validate()` helper
- It requires a form ID and a field ID to be present

Where it fails is in the handling of the **destination** filename - what the plugin will call the file once it has been saved to disk.

After validating the uploaded file, the plugin reads a POST parameter (a key derived from the file field name) to determine the destination filename. In the vulnerable versions, that destination filename is not sanitized and its extension is not re-checked. An attacker uploads an innocent-looking file such as `document.jpg`, which passes the source validation, then supplies a POST parameter telling the plugin to move it to a file called something like `../../../../var/www/html/shell.php`.

The plugin calls `move_uploaded_file()` with the attacker-controlled destination and writes a PHP file to wherever the attacker asked for it. Request that PHP file directly, and you have remote code execution on a server running around 50,000 sites' worth of contact forms.

The fix in 3.3.27 adds the checks that were missing: `basename()` to strip path traversal, `sanitize_file_name()` to clean the filename, and `pathinfo()` plus a blacklist check on the destination extension. You can see the sanitization logic clearly in the patched code Wordfence published, including this comment from the vendor:

```
// Security fix: Sanitize user-provided filename to prevent path traversal
```

File upload AJAX handlers have far more surface area than they look. Checking the obvious things (nonce, source filename) is not the same as checking the complete data flow. The obvious checks are the ones developers remember. The subtle ones - destination path traversal, destination extension, basename sanitization, the interaction between raw POST input and WordPress's own filename functions - are the ones that get missed. That is the AJAX pattern in a sentence, and it's the reason this post exists.

Why did the first Ninja Forms File Uploads patch not fix CVE-2026-0740?

Wordfence received the responsible disclosure on January 8, 2026 and shipped a firewall rule to Premium customers the same day. The vendor acknowledged on January 12, sent a patch for review on January 27, and released **3.3.25 on February 10** as the first patched version.

Except 3.3.25 was not actually patched. It fixed part of the destination filename handling but left the exploit path open. **3.3.26 was also still vulnerable.** The fully patched version, **3.3.27, did not ship until March 19** - over a month later.

If your WordPress auto-updater ran any time between February 10 and March 19, it pulled down a version that looked patched on every monitoring dashboard and every security scanner feed, but was still exploitable. The version number went up, the changelog mentioned a security fix, the dashboard went green. Nothing about that process told you the fix had failed. This is one of the reasons some agencies disable auto-updates entirely and drive all plugin updates manually from a central dashboard after verifying the patch is the *actual* fix.

Wordfence's advisory spells out the timeline in plain text: "The vulnerability was partially patched in version 3.3.25 and fully patched in version 3.3.27." If you only read the CVE summary and see "patched in 3.3.25", you will miss it.

The only safe version is **3.3.27 or later**. Anything below that is a live exposure, and "we updated it" and "we're safe" are not the same sentence. A vendor can genuinely try to fix an AJAX handler and still ship an exploit because the surface area is that large.

Which WordPress sites run Ninja Forms File Uploads?

Ninja Forms is one of the more popular form plugins for WordPress. The File Uploads extension is a paid add-on that plenty of sites use for:

- Recruitment sites collecting CVs from applicants
- Professional services sites accepting project briefs and portfolios
- Support portals where customers attach logs or screenshots

- Contact forms on agency sites that need file attachments
- Event registration forms accepting ID documents

Anywhere a WordPress site accepts files from the public through Ninja Forms, there is a reasonable chance this add-on is the plugin doing it.

Across the sites connected to mySites.guru, hundreds of installs turn up spread across many customer accounts, and a meaningful fraction are still on vulnerable versions. That is the reality across every large WordPress portfolio right now: some sites are patched, some are not, and without a dashboard that lets you group every install by version number, you are guessing.

How do I find which of my WordPress sites are running Ninja Forms File Uploads?

There are three ways to answer the question “which of my sites are running Ninja Forms File Uploads below 3.3.27”. Only one of them scales.

Manual check (single site)

If you only manage one WordPress site, the manual check takes a minute:

1. Log into wp-admin
2. Go to **Plugins** -> **Installed Plugins**
3. Find “Ninja Forms - File Uploads” in the list
4. Look at the version number

Anything below 3.3.27 - including 3.3.25 and 3.3.26, which show up as “updated” but are not actually patched - needs updating immediately.

WP-CLI (handful of sites)

If you have SSH access and WP-CLI on each site, this gets you the version without logging into wp-admin:

```
wp plugin get ninja-forms-uploads --field=version
```

And to update in one command:

```
wp plugin update ninja-forms-uploads
```

Workable for five or ten sites if you don't mind SSHing around. Unworkable for fifty.

Bulk check with mySites.guru (unlimited sites)

mySites.guru indexes every plugin on every connected site continuously. The Extension Search groups all variants of a plugin by its internal key hash, so a single URL lists every version of Ninja Forms File Uploads across your entire portfolio, grouped by version number.

mySites.guru subscribers: check your Ninja Forms File Uploads versions now

Open Ninja Forms File Uploads Extension Search

Lists every version of Ninja Forms File Uploads across all your connected sites, grouped by version number. Anything below 3.3.27 needs updating right now. Not a subscriber? Sign up free to get this visibility across your whole portfolio in minutes.

Or, if you prefer to drive it from the dashboard:

1. Open **Your WordPress Extensions** in mySites.guru
2. Type "ninja-forms-uploads" in the filter bar

3. Click **Which sites?** next to the entry to see every site running it, grouped by version number
4. Any site on 3.3.26 or earlier is vulnerable - you already have the list
5. Push the update in bulk from the same dashboard

Sites running vulnerable versions are flagged automatically with a red **Vulnerable Plugins!** badge. You do not need to remember version numbers or cross-reference CVE databases - mySites.guru cross-references every installed plugin against the Wordfence and Patchstack vulnerability feeds twice daily and flags outdated or vulnerable versions for you.

The question "which of my sites are still on 3.3.26" goes from an afternoon of manual checks to a few seconds. A hundred-site portfolio resolves in the same time as a five-site portfolio.

Single-site tools cannot answer this question by design. Wordfence runs on each site individually and does not give you a cross-portfolio view. ManageWP and MainWP track WordPress but not Joomla. mySites.guru is built around this specific question: given a CVE, which of my sites are affected right now, and how do I push the fix without logging into each one.

What do I do if I find a WordPress site exposed to CVE-2026-0740?

Patching is step one, but this is an unauthenticated RCE. Anyone scanning the internet for vulnerable installs could already have dropped a webshell. Treat any site you find on 3.3.26 or earlier as potentially compromised and check before you close the ticket.

1. Update to 3.3.27 or later. Not 3.3.25. Not 3.3.26. From mySites.guru, push the update in bulk to every affected site at once.

2. Scan for webshells and backdoors at the file level. The vulnerability writes attacker-controlled PHP anywhere writable on disk, so grep-for-suspicious-filenames is not enough. mySites.guru's deep file scanner inspects every file in the webspace

against 12 years of threat signature data and flags webshells, backdoors, and known hacked files regardless of where they were placed. Run it on every site you just patched, not just the obviously dodgy ones. The ones that look clean are the ones you missed. There is more detail on how this works in our post on finding hacked files and backdoors in Joomla and WordPress.

3. Review the Ninja Forms uploads directory manually too. Uploads typically land under `wp-content/uploads/ninja-forms/`, but because of the path traversal, the malicious file could be anywhere. Look for PHP files in upload directories, recently modified PHP files outside `wp-content/plugins/`, and any file with a suspicious name in the webroot.

4. Grep your webserver access logs for POST requests to `admin-ajax.php` with an action parameter matching `nf_fu_upload` or similar. Anything from an unexpected IP, at an unusual time, or with a suspicious `referer` header is worth investigating. mySites.guru's real-time activity alerting also catches new PHP files being written and admin logins from unfamiliar IPs, which is exactly the post-exploitation signal you want to see on this kind of vulnerability.

5. Check the usual post-compromise indicators. Unknown admin users, modified core files, unexpected cron jobs, new scheduled tasks, outbound connections from the web server process. Our post on how to tell if your WordPress site is hacked covers the full checklist.

6. Rotate credentials if you find evidence of compromise. Treat anything in `wp-config.php` (database credentials, salts, API keys) as leaked and rotate it. Reset every admin password. Invalidate active sessions.

If you find a compromised site and do not want to handle the clean-up yourself, mySites.guru's hack recovery service takes over from here: we identify every affected file, remove the backdoors, patch the vulnerability, and harden the site against re-compromise.

How can I prevent the next WordPress plugin AJAX vulnerability from hitting my sites?

This is not going to be the last WordPress plugin AJAX handler to ship an unauthenticated RCE. The pattern of "check the obvious thing, miss the subtle one" is structural, not accidental, and every time a new one lands the game is the same: find your vulnerable sites before the scanners do.

A few things tilt the odds in your favour, and mySites.guru does all of them by default:

- Continuous vulnerability scanning across every connected site, cross-referenced against Wordfence and Patchstack feeds twice daily. A new CVE drops, the dashboard tells you who is exposed, you don't have to read the advisories yourself.
- Real-time file-change alerting so new PHP files appearing in upload directories or the webroot trigger an immediate notification. This is the single highest-signal alert for file-upload RCEs - the attacker's first move after exploitation is writing a file that did not exist before.
- Bulk plugin updates across your whole portfolio in one action, so the window between disclosure and patch is minutes, not days.
- **Admin login alerting** on every connected site, catching the common post-exploit move of creating a new administrator account.
- **Daily deep file scans** against 12 years of threat signature data, so webshells dropped via vulnerabilities you did not even know about still get caught before they do damage.

This is why auto-updates are a safety net and not a strategy. They will happily carry you onto a partial fix like 3.3.25 and then stop. Monitoring, bulk-update control, file-change alerting, and deep scanning are what catch the things auto-updates miss. If you want a concrete operational routine for running these checks across a portfolio, we wrote up exactly that in how to build a five-minute morning routine for checking all your sites.

What is the broader lesson for WordPress agencies?

A version number that *looks* patched is not the same as a version that *is* patched. The only authoritative check is to compare the installed version against the fully patched version published by the vendor, not the first version the vendor labelled as a fix.

AJAX file upload handlers have far more surface area than they look, and "we patched it" and "we are safe" are not the same sentence. A vendor can genuinely try to fix an AJAX handler and still ship an exploit because the surface area is that large. If you manage more than a handful of sites, you need a way to answer "which of my sites are running plugin X below version Y" in under a minute, and a way to see if anything unexpected has been written to disk since the last scan. Without both, you cannot respond to vulnerabilities like this before the scanners finish their sweep.

Further reading

- Wordfence: 50,000 WordPress Sites affected by Arbitrary File Upload Vulnerability in Ninja Forms File Upload - the primary public disclosure with full technical analysis, vulnerable code, and disclosure timeline
- Wordfence advisory for CVE-2026-0740 - the Wordfence Intelligence entry with CVSS vector and version detail
- AJAX Endpoints Are A Big CMS Security Blind Spot - our deep dive on why WordPress and Joomla AJAX handlers keep producing critical CVEs

- Four Major WordPress Plugins Patched Security Flaws in March 2026 - the previous month's plugin CVEs, several of which shared the AJAX-handler root cause
- Astroid Framework Vulnerability - sibling Joomla AJAX handler CVE (CVE-2026-21628) with the same root cause
- Novarain Framework Joomla Vulnerability - another Joomla `com_ajax` handler shipping unauthenticated file operations (CVE-2026-21627)
- Smart Slider 3 Arbitrary File Read Vulnerability - WordPress `admin-ajax.php` handler skipping authorisation (CVE-2026-3098)
- Is My WordPress Site Hacked? - checklist of post-compromise indicators if you suspect a site has been hit
- Patchstack entry for Ninja Forms File Uploads - full vulnerability history for the plugin

Frequently Asked Questions

What is CVE-2026-0740 in Ninja Forms File Uploads?

CVE-2026-0740 is an unauthenticated arbitrary file upload vulnerability in the Ninja Forms File Uploads WordPress plugin, rated CVSS 9.8 Critical. It affects all versions up to and including 3.3.26 and allows unauthenticated attackers to upload arbitrary PHP files and achieve remote code execution. The vulnerability was responsibly disclosed to Wordfence on January 8, 2026 and publicly disclosed on April 6, 2026.

Which version of Ninja Forms File Uploads fixes CVE-2026-0740?

Only version 3.3.27 contains the full fix. Version 3.3.25 was a partial patch that did not fully close the vulnerability and is still exploitable. Version 3.3.26 is also still vulnerable. If your sites are on anything below 3.3.27 you need to update immediately.

How does the Ninja Forms File Uploads vulnerability actually work?

The vulnerable code is the NF_FU_AJAX_Controllers_Uploads::handle_upload function. It checks a nonce and validates the source file type, but does not validate the destination filename or sanitize it against path traversal. An attacker uploads a harmless-looking file, then supplies a POST parameter telling the plugin to move it to a PHP filename anywhere writable on disk. The plugin honours the destination path and the attacker now has a PHP file they can request directly.

Why did the first patch for CVE-2026-0740 fail?

The vendor shipped a partial patch in version 3.3.25 on February 10, 2026, but it did not fully address the destination filename handling. The complete fix landed in version 3.3.27 on March 19, 2026. Sites that auto-updated to 3.3.25 or 3.3.26 are still exploitable and many site owners do not realise this.

How can I check Ninja Forms File Uploads versions across all my WordPress sites at once?

Log into each site's wp-admin, go to Plugins, and look at the version column for Ninja Forms File Uploads. For agencies managing more than a handful of sites this is impractical. mySites.guru indexes every plugin on every connected site and groups all versions of a given plugin into one dashboard view. You can see every Ninja Forms File Uploads install across your portfolio in one screen, with vulnerable versions flagged.

Is Ninja Forms File Uploads being actively exploited?

Wordfence Premium, Care, and Response customers have had firewall protection against CVE-2026-0740 since January 8, 2026, and free Wordfence users since February 7, 2026. The vulnerability was only publicly disclosed on April 6, 2026, so widespread post-disclosure exploitation is still developing. Given the CVSS 9.8 rating and unauthenticated nature, mass scanning should be expected in the days following disclosure.

Why do AJAX endpoints keep appearing in WordPress plugin CVEs?

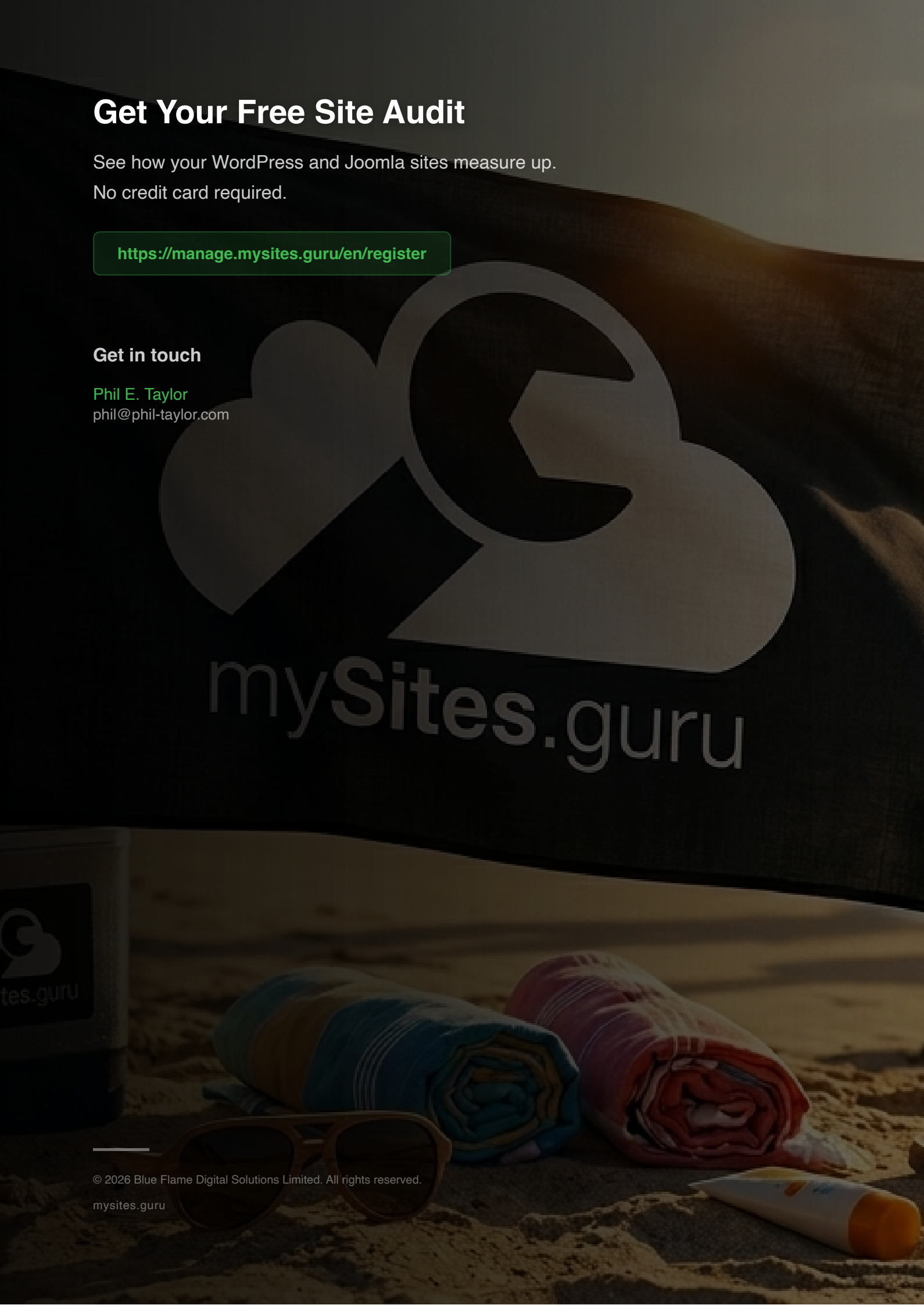
WordPress's admin-ajax.php is a utility endpoint designed for plugins to register their own request handlers. The framework performs minimal checks and leaves authentication, authorization, and input validation to the plugin developer. Ninja Forms File Uploads is a case where the plugin did check a nonce and did do some validation, but missed the specific case of destination filename handling. AJAX handlers have a large attack surface and it is easy to check the obvious things while missing the subtle ones.

Get Your Free Site Audit

See how your WordPress and Joomla sites measure up.

No credit card required.

https://manage.mysites.guru/en/register



Get in touch

Phil E. Taylor
phil@phil-taylor.com