Events Booking for Joomla: Anyone Could Upload Files to Your Server

mySites.guru found two unauthenticated flaws in Events Booking for Joomla: file upload enabled by default, and a leak of every user's name and email.

Phil E. Taylor | 28 August 2026

Events Booking is one of the most widely used event registration and ticketing components for Joomla. It runs conference sign-ups, class bookings and ticket sales, which means the accounts on the site belong to real attendees who gave over their details to book something. During routine security research on the extensions our customers rely on, **mySites.guru found two unauthenticated vulnerabilities in Events Booking and reported them privately to JoomDonation.**

FINDING ONE

Simply having Events Booking installed meant any anonymous visitor could upload files to your web server.

No account, no login, no CSRF token, enabled by default on every install.

Secondly, an access control flaw that returned **any user's name and email address** to anyone who asked for it by user ID.

If you run Events Booking on any Joomla site, update to **5.8.1** now, not 5.8.0. On Joomla 3 the fixed version is **4.9.5**, which is not in the public downloads area.

mySites.guru discovered these issues and reported them privately to JoomDonation, copying the Joomla Security Strike Team. Fixes shipped in Events Booking 5.8.0 and 5.8.1. We are withholding the exact requests and any proof-of-concept while site owners update. This is how we handle every vulnerability we find: fix first, publish second.

TL;DR

- **Anyone could upload files to your server.** A front-end endpoint accepted uploads from anonymous visitors with no login and no CSRF token, **enabled by default** on every install. The vendor has confirmed the default in writing
- **And the site serves it back out again.** For image extensions the matching download endpoint had no login check either, and filenames were not randomised, so an attacker uploaded under a name they chose and then fetched it from a URL they already knew. Anonymous in, anonymous out: a working file hosting service on your domain
- **That is your name on the hosting contract.** Free anonymous storage attracts pirated media, phishing kits, malware staged for other attacks, and illegal imagery, all served from your domain and billed to you. We have seen this arrive as a hosting abuse report before
- **Customers were emailed that it is "a security improvement rather than a vulnerability"** on 20 July, three days after the Joomla CNA assigned it a CVE
- **Anyone could read every user's name and email.** A second public endpoint returned a user's first name, last name and email address as JSON for any user ID passed to it, with no ownership check. Count up through the IDs and you have the whole user table
- **The vendor told us repeatedly that the upload was not a security issue.** The Joomla CNA assigned it CVE-2026-60024 regardless, describing it in those exact terms
- **Then came the fixes for the non-issue the developer stated did not exist. Funny, that.** Across 5.8.0 and 5.8.1 that endpoint gained a CSRF token, a per-field permission check, per-session and per-minute rate limits, folder size and file count caps, real image content validation, randomised filenames and an admin tool to clean up the files it had already accepted
- **We were then asked, more than once, to keep quiet about it**, each time after a release had shipped and, on the second occasion, after every customer had already been emailed
- **5.8.1 is the version to be on.** 5.8.0 closed the two reported flaws; 5.8.1 is where the upload hardening actually happened

- **Two different packages shipped as "5.8.0".** The vendor rebuilt it mid-afternoon on 15 July without changing the version number, so updating early got you the weaker build with no way to tell
- **Joomla 3 users need 4.9.5, which is not in the public downloads area.** Configure a Download ID and use Joomla's updater, or open a support ticket
- Three CVEs assigned via the Joomla CNA: CVE-2026-58149, CVE-2026-60024, CVE-2026-60025, all crediting Phil Taylor of mySites.guru
- Across the Events Booking installs we can see on our customers' sites, **not one was running the current release**, and roughly three quarters were a whole major version behind

Finding one: anonymous file upload, on by default

Events Booking supports File custom fields, so an event organiser can ask registrants to attach something during sign-up: a photo, a signed waiver, a proof of eligibility. The upload runs over an AJAX endpoint on the front end.

In Events Booking 5.7.1 and earlier, that endpoint had no login check and no CSRF token. It took whatever file it was given, checked the extension against the configured attachment allow-list, and wrote it into `media/com_eventbooking/files`. Crucially, it did not check whether the site was even asking for uploads. The endpoint was live on every install from the moment the component was present, whether or not the administrator had ever created a File field.

That last point is the vendor's own, given to the Joomla Security Strike Team on 17 July:

> **JoomDonation, to the Joomla Security Strike Team, 17 July**
>
> It was enabled by default in previous versions. From 5.8.0, it will only be enabled if admin creates custom field to allow registrants to upload file when they register for the event.

The Joomla CNA description of CVE-2026-60024 puts the same fact in the official record: *"The Joomla extension Events Booking prior version 5.8.0 did by default allow unauthenticated users to upload media assets."*

What that actually buys an attacker

The allow-list is what keeps this from being a critical finding. The shipped default is `bmp|gif|jpg|png|swf|zip|doc|pdf|xls`, which **does not include a PHP extension**. The extension check reads the last segment of the filename, Joomla's own filename sanitising is applied, and the usual double-extension and traversal tricks do not get through.

So on a default configuration this stops short of remote code execution. On a site whose administrator has widened the allow-list to include an executable type, which is unusual but not unheard of where sites accept unusual attachments, it becomes a straightforward unauthenticated remote code execution.

Read that default list again with an attacker's eyes rather than an event organiser's, though. `zip`, `doc`, `xls` and `pdf` are no threat to the server itself. They are close to ideal for distributing malware from a domain nobody has any reason to block, and `doc` and `xls` have been the standard carriers for macro payloads for twenty years.

This is not the file upload RCE we found in Balbooa Forms, PageBuilder CK or RSFiles!, and we are not going to pretend it is. It is a stranger with write access to your file system, which is quite bad enough.

The site will also serve it back for you

The upload is only half of it, and the other half is the part that turns a storage problem into a distribution problem.

Events Booking exposes a matching front-end endpoint that reads files back out of that same folder. In 5.8.0 and earlier its permission logic ran like this: if the filename ends in `png`, `jpg`, `jpeg`, `gif` or `webp`, the file is served to whoever asked for it, with **no**

login check at all. Only the other extensions fall through to the ownership and permission checks underneath.

That test was on the extension, never the contents. Before 5.8.1 nothing verified the file was actually an image, and nothing randomised the stored filename, so the attacker chose the name they uploaded under and therefore already knew the URL to fetch it back from. The endpoint even takes an `inline` parameter from the caller, so the attacker picks whether the browser displays the file or downloads it.

BOTH HALVES TOGETHER

Before 5.8.1, Events Booking would accept a file from an anonymous stranger and then serve that same file back to the entire internet, at a filename the stranger chose, with nobody logged in at either end.

That is not a side effect of a file upload bug. It is a complete, working, anonymous file hosting service running on your domain and your bandwidth, which you did not know you were operating.

This does have a ceiling. The response declares a content type worked out from the extension, so a `.jpg` goes out as `image/jpeg`, and current browsers will not run a script out of that, which makes the direct cross-site scripting route weak. As free unattributable hosting on a reputable domain with a valid certificate, it works exactly as an abuser would want.

Both parts were tightened in 5.8.1: filenames are now randomised on upload, and the download path validates that a file really is an image before serving it inline.

A stranger with write access to your file system

Code execution is the loudest thing an unauthenticated upload buys an attacker, and treating it as the only one that counts is how this kind of finding gets waved through. The vendor's own changelog describes the risk as someone uploading files "to abuse server quota for some reasons". Server quota is the least of it.

Anonymous, unattributable file storage on somebody else's infrastructure is a commodity, and people go looking for it. An open upload endpoint on a legitimate business domain is close to ideal for that: it costs nothing, it is not registered to anyone, the bandwidth is billed to a stranger, and the content inherits the reputation of a real site with a real TLS certificate that no filter has any reason to block.

What ends up in those folders is not theoretical. Pirated media and warez, because someone else is paying for the disk and the transfer. Phishing kits and fake login pages, which work far better when the URL is a real company domain than when it is a disposable host. Malware and droppers staged for a different campaign entirely, so the attack on somebody else runs through your server. And illegal imagery, including sexual content and material involving children, parked somewhere the people trading it will not be traced.

I should be straight about the limits of that list. I am not a hacker. I do not think the way they think, and every time I have looked closely at a real compromise the actual use has been more inventive than anything I would have predicted. People who do this for a living are extremely good at finding free storage where they will not get caught, and they will find uses for your upload folder that have not occurred to me.

The part that matters for you is that all of it arrives with your name on it. You signed the hosting contract. The IP address, the domain and the billing details are yours. When the abuse report is written it is addressed to you, and hosts generally suspend first and investigate afterwards, which means the first thing you know about it may be a dead site and a support ticket. Beyond that sit domain and IP blocklisting, a payment processor taking an interest, and for the worst categories of content a set of legal obligations that are considerably more serious than a hosting suspension.

At the far end of that road is the police at your door, because a Joomla extension on your site let a stranger store illegal material on it. That does happen. It happens to ordinary small business websites run by people who did nothing reckless, because an open upload form on a real company domain is exactly the sort of place that content gets parked. The person who has to answer for it is whoever's name is on the domain, and "the form needed no login" is a poor opening line for that conversation.

We have watched this happen. The Balbooa Forms flaw we disclosed this month did not come to us from a code audit. It came from a Hetzner abuse report sent to one of our customers, who brought us the raw access log. That is what an anonymous upload endpoint looks like from the owner's side: a message from your host telling you your server has been doing something you knew nothing about.

The quiet part is that you will probably not notice on your own. Nothing errors. No page breaks. Files accumulate in a directory that nobody has any reason to open, attached to no registration, and the site keeps serving events exactly as it did before.

DO THIS AFTER YOU UPDATE

5.8.1 ships a tool that finds and deletes every file in the upload folder with no matching registration record.

Run it even if you are certain you were never a target. It is the only way to see what that endpoint accepted while it was open, and there is a scheduled task version so it keeps checking.

Finding two: every user's name and email

The second endpoint is a textbook insecure direct object reference. Hand it a user ID and it returns that user's stored registration details. Where the person never registered for anything, the code falls back to loading the Joomla user account directly and returns their **first name, last name and email address** as JSON anyway.

Nothing in that path checked who was asking. No login check, no CSRF token, and critically no ownership check: the code never asked whether the requester was the user whose details were being returned. The component's own JavaScript only ever calls this endpoint for the currently logged-in user, so in normal use it looks harmless. The server never enforced what the browser assumed.

An attacker requests user ID 1, then 2, then 3, and keeps counting. Each request returns one person's name and email. There was no rate limit and no authentication, so a short script walks the entire user table in minutes.

The user ID is cast to an integer before it goes near the database, so there is no SQL injection here and no way to bend the query. This is purely an authorisation failure. The code asked the database a perfectly sensible question and gave the answer to the wrong person.

Why enumeration matters more than it sounds

"Names and email addresses" sounds like the mild end of a breach, and compared to a database of password hashes it is. That framing undersells it for three reasons.

Targeted, not random

These are the verified contact details of people who signed up on a specific site for a specific thing. An attacker who knows you booked onto a named event at a named organisation can write a phishing email that reads exactly like the follow-up you are expecting. That is worth far more than a bulk address list, and it prices accordingly.

A shopping list for the next attack

Enumeration is reconnaissance. Once an attacker has every username and email on a Joomla site, credential stuffing has real targets, password reset flows have real addresses to abuse, and the administrator accounts stand out immediately.

Your users are your customers

On an events site, unlike a brochure site where a leaked user table might be three staff logins, the users are the customers.

THIS IS A REPORTABLE DATA BREACH

On a busy Events Booking install it is every attendee who ever registered, which under the UK GDPR and the EU GDPR is personal data you are the controller of.

An unauthenticated stranger reading it is a personal data breach, and one you are expected to assess and potentially report to your regulator within 72 hours.

Note

Enumeration leaves almost no trace. The requests are ordinary GET requests to index.php that return a normal 200 response, so in an access log they look like a visitor browsing the site quickly. There is usually no error, no alert and nothing in the Joomla logs. If you are wondering whether anyone did this to your site, the answer for most sites is that you cannot tell from the logs alone.

"To me, this is not really a security issue"

Part of deciding whether to trust an extension is watching what the vendor does when someone hands them a security problem. On the enumeration flaw JoomDonation moved fast and added the missing permission check without argument, and they deserve credit for that.

The upload went differently.

The vendor's position, held across several days and multiple emails, was that the finding was invalid. The argument was that because the endpoint is meant to be public, a CSRF token adds nothing:

> **JoomDonation's reply**
>
> The point here is this is public end point to allow public users (registrants) to upload files while they register for event. It is not something restricted for admin, so it is not necessary for attackers to use to use an external site to perform upload.

And again, thirteen minutes later:

JoomDonation, thirteen minutes later

I wonder why someone has to use an external site malicious website website to make a request like that for a public accessible form? What prevent them from make request directly to that public form?

We quote these exactly as written, duplicated words included. English is not the developer's first language and we are not making a point about the phrasing. The point is the position, which was maintained clearly and consistently: this is public, therefore a token is pointless, therefore there is no bug.

The Joomla Security Strike Team, copied on the thread throughout, disagreed on the record:

The Joomla Security Strike Team

I disagree with your assessment on the CSRF issue: a file upload is a mutating operation and should be protected with a CSRF token check. Otherwise, I could trick a site admin on an external site to submit a form and having that form targeted to the website that the admin is managing. If the admin has an active session, the upload will be performed.

When the vendor pushed back a second time, the JSST reply opened: *"You are wrong :)"*, followed by a step-by-step walkthrough of what cross-site request forgery actually is.

The objection misses what the token is for. It stops a third-party site from making your visitors upload files without their knowledge, which is a different attack from an attacker calling the endpoint directly. Both are real. Fixing one does not answer the other. And neither addresses the bigger problem, which was never the token: it was that the endpoint accepted files from strangers at all, on every install, by default.

The 5.8.0 announcement, as originally published, put the vendor's view in writing: *"To me, this is not really a security issue, but since it is reported, I have it fixed/improved anyway."* The same announcement, a few lines earlier, described the release as fixing

“two medium security issues”. Both statements cannot be true at once. That sentence, along with a claim that the first issue had been fixed before we reported it, was removed from the announcement three minutes after we put the two quotes to the vendor and said they were wrong. No correction note was added. We screenshotted the original beforehand, which is the only reason this paragraph can be written.

For the record, the Joomla CNA settled the question independently. It assigned CVE-2026-60024 for the default anonymous upload and CVE-2026-60025 for the missing CSRF token. Two CVEs, for the thing that was not really a security issue.

The position held in public, too. On the changelog thread, after the fixes had shipped, the vendor wrote:

> **JoomDonation, on the changelog thread**
>
> BTW, the current implementation is quite safe. But since the upload file is open for public users to upload files during their registration, if someone uses it to upload multiple files, there is no way for the system to prevent that. That is the same for any public forms which allow public users to upload files

The fixes for the non-vulnerability

“There is no way for the system to prevent that” appears in the same changelog post that documents four brand new settings whose entire purpose is to prevent exactly that.

We obtained the 5.8.0 and 5.8.1 packages and diffed them. If the anonymous upload were not a security issue, 5.8.1 would be a quiet maintenance release. Instead, here is what the vendor shipped for that one endpoint across the two versions.

Hardening added	Version	What it does
CSRF token check	5.8.0	The token the vendor argued was useless is now required on the upload

Hardening added	Version	What it does
Upload gated behind an existing File field	5.8.0	The endpoint is no longer live on every install by default
Per-field permission validation	5.8.1	The specific field being uploaded to must exist, must be a File type, and must be within the requester's authorised view levels
Max files per session (default 100)	5.8.1	New admin setting, capping how many files one visitor may upload
Max files per minute (default 20)	5.8.1	New admin setting, throttling upload rate
Upload folder size limit	5.8.1	New admin setting in MB, ships disabled
Max files in upload folder	5.8.1	New admin setting, ships disabled
Real image content validation	5.8.1	A 260-line image helper that runs `getimagesize` and `finfo` against anything with an image extension, so a file claiming to be a JPEG has to actually be one
Randomised filenames	5.8.1	Uploads are now prefixed with 16 random hex characters, so an attacker cannot predict or link to the path they wrote
Image validation on serving	5.8.1	The download endpoint re-validates that a file is a real image before serving it inline
`X-Content-Type-Options: nosniff`	5.8.1	Stops browsers MIME-sniffing a served file into something executable
Orphan file cleanup tool	5.8.1	A new admin function, described by the vendor as a "hidden tool", that deletes files in the upload folder with no matching registration record. There is also a scheduled task to run it automatically

Look at that last row again. The vendor shipped a tool to clean up the files that the not-a-security-issue endpoint had already accepted from strangers, and a scheduled task

so it keeps cleaning them up.

The changelog also credits an `.htaccess` in the upload folder, added "just in case admin allows uploading php files by mistake". We can confirm the file is present in both packages and contains `deny from all`. Do not lean on it: that is an Apache directive. On nginx it is an inert text file, and plenty of Joomla hosting is nginx or a proxy in front of Apache.

Alongside the upload work, 5.8.1 also tightened output escaping from `ENT_COMPAT` to `ENT_QUOTES` (the former does not escape single quotes, a common source of cross-site scripting), and extended Joomla's `filterText` to sixteen editor fields on the event form that were previously being read raw.

All of this is good work, and 5.8.1 is a better release than 5.8.0. It is also twelve layers of defence built around an endpoint the vendor was still describing, in public, as fine.

There are two different builds of 5.8.0

One more thing to check if you already updated. Buried in the changelog thread is this:

> **JoomDonation, on the changelog thread, 15 July**
>
> Just want to update that some additional checks were added to make field upload more secure (just in case someone uses upload end point to attack your site by mass upload), so if you updated Events Booking before 16:10pm, 15 July 2026, you might want to download latest package and update it to your site again

The version number did not change. Two different packages shipped as 5.8.0, and which one you have depends on what time of day you clicked update. Anyone who updated promptly, which is to say anyone who took the security release seriously, got the weaker build and has no way to tell from the version number.

This is the second time in this story that a package changed without the version changing: the enumeration fix was pushed into the download package on 14 July with no changelog entry at all. Version numbers are how site owners and tools like ours

reason about who is patched. When the contents move underneath a fixed version string, everybody downstream is working from bad data.

Going to 5.8.1 resolves it, which is the simplest reason to skip 5.8.0 entirely.

What 5.8.1 still does not do

Being straight about the limits of the fix, because “updated to 5.8.1” is not the same as “no longer reachable”:

Anonymous uploads still work, by design

A public registration form has to accept uploads from people who are not logged in. Where a File field’s access level is Public, which it must be for public registration, a guest still passes the new permission check. That is a legitimate design decision and the JSST accepted it as such. It does mean the endpoint remains an anonymous write path, now with limits on it.

The rate limits are per session

The 100-per-session and 20-per-minute counters are stored against the visitor’s session. An attacker who discards their session cookie between batches gets a fresh allowance every time. This slows down a naive script, not a determined one.

The absolute caps ship disabled

`upload_folder_size_limit` and `upload_max_files_in_folder` both default to `0`, meaning off. Out of the box, 5.8.1 has no ceiling on total files or total disk consumed. If you want one, you have to go and set it.

A cap you set buys a different problem

Both caps are global to the upload folder. Once the limit is hit, further uploads are rejected for everyone, including real registrants. An attacker who fills the folder to your configured cap has denied service to your event sign-ups. Set these values, by all means, but set them with headroom and monitor the folder.

What customers were actually told

On 20 July, JoomDonation emailed its Events Booking customers about the 5.8.1 release. That email is how most site owners will learn any of this happened, so it matters more than the changelog does. It describes the enumeration flaw as "Information disclosure: Under certain conditions, an unauthenticated user could retrieve the name and email address associated with a registered user." Then, on the upload, in bold:

> **JoomDonation's customer email, 20 July**
>
> This should be considered a security improvement rather than a vulnerability. It is not a remote code execution vulnerability, and attackers could not upload executable files or gain control of your website.

Taking that apart fairly, because parts of it are true.

On a default configuration, "it is not a remote code execution vulnerability" is correct, and we have said the same thing throughout this post. The default allow-list has no PHP in it.

"Attackers could not upload executable files" is correct for the default allow-list, and wrong as an absolute. That list is a configuration setting an administrator can edit, and the sentence makes a promise about every install rather than about the shipped defaults.

"Or gain control of your website" is a claim the vendor is not in a position to make. The same release that email is announcing added image content validation to the file-serving path and an `X-Content-Type-Options: nosniff` header to it. Those are defences against a served upload being interpreted as something other than an image. You do not add them to a path where nothing could ever go wrong.

Someone else already settled "this should be considered a security improvement rather than a vulnerability". The Joomla CNA assigned CVE-2026-60024 three days before

that email went out, describing the extension as having “by default allow[ed] unauthenticated users to upload media assets”. A CVE is the industry’s formal record of a vulnerability. It is not a security improvement.

“Under certain conditions” is doing an enormous amount of work here. The condition, for the enumeration flaw, is that somebody sends a request.

Two things the email leaves out are worth noting. It presents “this update automatically disables the upload endpoint when your site does not use any File Upload custom fields” as a helpful new behaviour, when it is the fix for the CVE above: the endpoint should never have been live on installs that were not using it. And across the whole message there is no CVE number, no advisory link and no indication that three CVEs exist for what it describes.

The practical effect is what concerns us. A busy agency owner reading “these issues are not considered critical” and “a security improvement rather than a vulnerability” will file it under housekeeping and get to it next month. Given that the version data above shows not one install we can see was even on the current release, that is the outcome least likely to get anybody patched.

And then we were asked, repeatedly, to say nothing

Running alongside all of the above was a request that kept coming back: please hold off publishing.

It came on 15 July, after the 5.8.0 announcement had already gone out describing both bugs and naming the fixed version. We were asked for two days. It came again on 20 July, after the newsletter had gone to the entire customer base, this time phrased as *“I hope you can wait and give customers few days to update because I only start sending newsletter today”*, with *“as mentioned before”* in front of it confirming this was not the first time. The same message asked whether publishing was *“even legal by low”*.

There is a legitimate version of this request, and it is worth stating clearly before disagreeing with it. Coordinated disclosure normally does include a grace period after a

patch ships. Researchers hold the technical detail back for a window so that site owners can update before attackers get a map. We honour that on every disclosure we make, which is why we sat on both of these findings until a fix existed.

The logic depends entirely on the vendor not having published the details themselves. That is not what happened here.

By the time the first request arrived, JoomDonation had already pushed a silent package on 14 July, published a changelog on 15 July that described both bugs by category and named the fixed version, and rebuilt the package again that afternoon. By the time the second arrived, an email had gone to every customer. A security release is itself a disclosure, and the loudest one available: it hands anyone who cares two packages to compare. We ran that comparison ourselves for this article. It took an afternoon and produced the table above. Nobody needs our permission or our write-up to do the same thing, and the people most likely to do it quickly are the ones with a reason to hurry.

So the delay does not slow an attacker down. It reaches the other group. The people who read security write-ups are the people trying to work out what to prioritise across a portfolio of client sites this week, and they are the only ones a silence reliably affects.

That matters more than usual here because of what those customers were told in the same breath. An email saying the upload is “a security improvement rather than a vulnerability” and that “these issues are not considered critical”, combined with no independent analysis available to weigh it against, is not a neutral state of affairs. It is a one-sided account of a set of flaws with three CVEs attached to them. Waiting a few days would have meant customers making patching decisions with only that account in front of them.

The argument does not survive the data either. “Give customers a few days to update” assumes a world in which a few days changes the outcome. Not one Events Booking install we can see was running the current release, and roughly three quarters were an entire major version behind. The sites actually at risk here are years behind, not days.

We agreed to the first request. We published nothing on 15 July, nothing on the 16th, and nothing in the days after that. This article is going out on 20 July, five days later.

What happened during those five days is the reason there is no sixth. A second build shipped under the same version number. A follow-up release went out. An email reached every customer describing a CVE as a security improvement. And the request came round again, this time for “a few days” more, with a question about whether publishing was even legal attached to it. A hold that renews itself every time the vendor makes another public statement is not a coordination window, it is an open-ended veto.

Our commitment is to withhold the exploit until a fix exists. We have done that and are still doing it: there is no proof-of-concept, no endpoint name and no request format anywhere in this article, and there will not be. That commitment does not extend to staying silent indefinitely while the vendor publishes its own version of events to thousands of people.

Which brings us to what happens next. Today we reported a further security issue to the same vendor, an IDOR in 5.8.1. That report enters a **90 day disclosure period**, the industry-standard clock, which runs to **18 October 2026**. JoomDonation has until that date to ship a fix, and we will publish after it whether one exists or not.

Fixing the date in advance is the answer to everything in this section. Nobody has to ask anybody for silence, nobody has to argue about whether a release counts as disclosure, and nobody has to wonder when the window closes. It is written down, both sides can see it, and the clock does not restart every time somebody publishes a changelog.

What coordinated disclosure actually requires

It would be easy to read the last few sections as a personal complaint. They are not. Coordinated disclosure is a documented, standardised process with obligations on both sides, and the reason those obligations exist is that site owners cannot make good decisions without them. This section is the part we would ask any extension developer

to read, because most of the friction in this case came from a process gap rather than from bad intent.

The relevant standards are not obscure. ISO/IEC 29147 covers vulnerability disclosure: how a vendor should receive reports and, importantly, how it should publish advisories. ISO/IEC 30111 covers the internal handling process that sits behind it. The CERT Guide to Coordinated Vulnerability Disclosure from Carnegie Mellon is the practitioner reference most researchers work to. FIRST publishes guidance for coordinating between multiple parties, which is exactly the situation once a national or project CNA is involved. OWASP publishes a short, readable cheat sheet for anyone who wants the summary rather than the standard.

They differ in detail and agree on the shape. The researcher withholds technical detail while a fix is produced. The vendor investigates in good faith, fixes, and then communicates accurately. Both parts matter, and the vendor's part is the one that gets skipped.

In practice, the vendor's obligations come down to this:

- **Assess the report on its merits, not on whether it feels like an attack you recognise.** "This endpoint is meant to be public" is a design statement, not a security assessment.
- **Publish an advisory that describes what was wrong**, not a changelog line that could equally describe a bug fix. Users triage from your advisory.
- **Change the version number when the contents change.** A package rebuilt under the same version breaks the one signal everybody downstream relies on, including automated tooling.
- **Request a CVE, and put the number in your communications.** Joomla runs its own CNA, so for Joomla extensions this costs an email.
- **Characterise severity honestly in customer-facing messages.** Understating it does not calm anybody down, it just means the update does not get installed.
- **Correct the record visibly.** Editing a published advisory without a note is how you lose the benefit of having published it.

- **Coordinate the timing, once, and in advance.** Agree a publication date with the reporter before you ship, rather than asking for open-ended silence afterwards.
- **Credit the reporter.** It costs nothing and it is the entire economy that produces reports like this one.

Measured against that list, this case was mixed rather than uniformly poor, and it is worth saying so. JoomDonation fixed the enumeration flaw quickly and without argument. They credited us publicly. They engaged with the JSST rather than ignoring it. They patched the Joomla 3 line when other vendors this month simply walked away from Joomla 3 altogether. The engineering response, once it got going, was thorough, as the hardening table above shows.

The communications were the failure. A silent package on 14 July, a changelog on 15 July that argued with the finding while fixing it, a second build shipped under the same version number that afternoon, published statements edited away without a correction note, a customer email describing a CVE as a security improvement, and repeated requests for our silence after each of those public acts. Every one of those is a communication decision rather than a coding one, and every one of them made it harder for a site owner to work out whether they needed to act.

That distinction matters because the bugs themselves are ordinary. Any codebase with a public front end grows them. What separates vendors is not whether they ship a vulnerability, it is what happens in the week afterwards, and that week is almost entirely made of communication.

Joomla 3 users: the fix exists, but it is not where you would look for it

JoomDonation has released **Events Booking 4.9.5** for Joomla 3, containing the same fixes. Credit where it is due: many vendors would not have bothered patching a Joomla 3 line at all, and this month we wrote about JoomShaper abandoning Joomla 3 entirely while known critical holes remained open. JoomDonation did the work.

The problem is the distribution. When a customer on the changelog thread asked where to download it, having found that the download button only offered 5.8.0, the answer was:

> **JoomDonation, on the changelog thread**
>
> We do not provide download package for Joomla 3 directly on Downloads section. If you need it, please submit a support ticket and we will send it to you

There is a second route, and it is the better one if it works for you: configure your **Download ID** in the component and use Joomla's own updater from the site's administrator area, which will offer 4.9.5 in place. That is the path the vendor recommends. Worth knowing, though, that the customer who asked reported back on the same thread that the updater route gave them a **403 error** when they clicked through, which they suspected was their Admin Tools firewall blocking the update URL.

So on Joomla 3 the fix is real, but reaching it means either configuring a Download ID and hoping your firewall lets the updater through, or opening a support ticket and waiting for a human to email you a zip. Every extra step between a fix and the sites that need it costs you installs, and a support ticket is an expensive step: it needs an active subscription, a login, someone to write the ticket, and someone at the other end to answer it. Meanwhile the vulnerable version stays on the site. Attackers do not need to open a ticket to read the changelog and work out what changed.

If you run Events Booking on Joomla 3, start that today. And treat it as one more argument for the migration you have been putting off, because a security fix you have to request by hand is not a maintenance model that scales.

The number that worries us is not the severity

Here is the part of this story we find most alarming, and it has nothing to do with how clever either bug is.

We can see Events Booking installed on **just over 470 Joomla sites** across our customers' accounts. When we checked which versions those sites were running, we did not find a long tail of stragglers behind a well-updated majority. We found this:

Version line	Share of installs
5.7.x (the current line at the time)	0%
5.x, but behind the current line	25%
4.x	65%
3.x	7%
2.x or older	2%

Read that top row again. **Not one site we can see was running the current release.** Roughly three quarters were an entire major version behind, and the single most common version across every install was a 4.9 release.

This is the uncomfortable truth behind every responsible disclosure. We found the bugs, they got argued about, they got fixed, and a patch now exists that will reach approximately nobody, because the sites running this extension were already years behind before we started looking. The security industry publishes a fixed-in version and calls the job done. The rollout is the part that actually protects anyone, and on this evidence the rollout barely happens.

If you manage Joomla sites and have not looked at your extension versions recently, that table is probably a description of your own portfolio. I have never met an administrator who decided that 4.9.4 was good enough. What I have met is a lot of people for whom updating an extension means logging into each site's admin one at a time, which is a job that grows with every client you win, so it waits until something breaks.

The root cause is a gate that was never there

Both findings come from the same architectural decision, and it is worth understanding because it predicts where the next bug in this codebase will be.

Events Booking's front end runs on the vendor's own controller framework. When a request arrives, the dispatcher takes the `task` parameter, maps it to a method on the controller, and calls it. That is a normal enough design. The problem is what the dispatcher does not do on the way: it performs **no CSRF token check and no permission check at all**. Both are helpers that each individual method has to remember to call for itself.

Opt-in security has a predictable failure mode. Every public method on every front-end controller is reachable by an anonymous visitor unless its author remembered to bolt a guard onto it, so the security of the component rests on nobody ever forgetting. Somebody always forgets.

The developers here clearly know the risk, which is what makes it instructive. Several of the front-end controllers do call the token check and a permission check, correctly, right at the top of the sensitive methods. The two endpoints we reported are the ones where that did not happen. Read that as a framework problem rather than a competence one: the guard is a line you add, so its absence is invisible. Nothing looks wrong on the screen. The reviewer has to spot a line that was never written, which is a much harder thing to ask of anyone than spotting a mistake.

That is the same shape as the AJAX endpoint problem we have written about, and the same reason it keeps producing findings across unrelated Joomla extensions. A framework that gated every task by default, and made a developer opt a method *out* to make it public, would have prevented both of these findings without anyone having to be careful.

How serious are they?

Two flaws, two different answers, and neither of them is a 10.0. We could have written this up as critical and plenty of the industry would have, but calling it a 10.0 would be

dishonest and would make it harder for you to believe us next time something really is one, as several of this month's findings were.

The anonymous upload

The Joomla CNA assigned CVE-2026-60024 and CVE-2026-60025 and has not published scores for either at the time of writing. Our own assessment puts it in the **Medium** band on a default configuration, with a **High** ceiling on any site that widened its attachment allow-list to an executable type, where it becomes unauthenticated remote code execution.

The allow-list is what keeps the default case out of the high range: no PHP, so no code execution. What stops it being a footnote is everything in the section above, and a CVSS vector captures that badly. "Integrity: low" is a thin way to describe illegal content served from your domain, under your hosting contract, in your name.

Severity here scales with how exposed your site is and how long it ran an affected version, rather than with how many users you have. A new site nobody has found yet is probably fine. A site that has been in Google for years with a public registration form has been reachable by every scanner on the internet for as long as this code has existed.

The user enumeration

We score this **CVSS 4.0 6.9 (Medium)**. An anonymous attacker gets names and email addresses, not password hashes, and cannot write to the database or execute anything through it.

Severity here scales directly with your user table. A club site with six logins has a small problem. A ticketing site with 40,000 registrants has a personal data breach involving 40,000 people, from a bug that took one afternoon to find.

What both share

Neither has any barrier in front of it. No account, no token, no user interaction, no clever timing, no waiting for an administrator to click something. Anyone with a browser and a for-loop, against any affected site on the internet, with nothing in the logs. Compare that with the Phoca Download flaw from the same run, which scored higher but needed a registered account and a non-default setting enabled to work at all.

> Update to Events Booking 5.8.1, or 4.9.5 on Joomla 3. A web application firewall will not save you here, because unlike a SQL injection there is no malicious payload for it to recognise: the attack is an ordinary, perfectly-formed request that the site is supposed to answer. The only thing wrong with it is who sent it, and your firewall has no way to know that.

That last point is worth dwelling on, because it inverts the usual advice. With the SQL injections we found in EDocman and DPCalendar, a SQL-filtering firewall could at least buy you time, because the attack contained recognisable SQL. Here there is nothing to filter. The request is valid. That leaves updating as the whole of your defence.

Which versions are affected, and which do you need?

We audited **Events Booking 5.7.1**, the current release when we found this, and confirmed both issues in it. Being straight about the limits of that: we did not audit every older release, so we cannot tell you exactly when either endpoint first appeared. What we can tell you is that both are long-standing features rather than something new in 5.7.

Given the version spread above, treat any Events Booking release at or below **5.7.1** as affected: that is very likely every install you own.

Your Joomla	Version to be on	How to get it
Joomla 4, 5 or 6	**5.8.1** or later	Joomla's Extensions updater, or your JoomDonation account

Your Joomla	Version to be on	How to get it
Joomla 3	**4.9.5**	Not in the public downloads area. Configure a Download ID and use Joomla's updater, or open a support ticket

There is no 5.7.2: the release went straight from 5.7.1 to 5.8.0, then 5.8.1 followed within days. **Do not stop at 5.8.0.** Every one of the upload hardening measures in the table above arrived in 5.8.1, and 5.8.0 itself shipped as two different builds under one version number.

We have now reviewed both packages against what we reported. The enumeration fix in 5.8.0 is correct: the endpoint now requires the `eventbooking.registrantsmanagement` permission and throws a 403 without it. The upload fixes are as described above.

How do you update Events Booking safely?

1. **Take a backup first.** Before any extension update on a production Joomla site, back up the database and files. If you use mySites.guru, trigger a snapshot or a full backup so you have a clean starting point.
2. **Expect a real upgrade, not a patch, if you are on 4.x.** Two thirds of the installs we can see are a major version behind. Going from a 4.x build to 5.8.1 crosses a major version boundary, and on a component that handles registrations and payments you want that tested somewhere other than production. This is the tax on having skipped updates for years, and it is exactly why the sites that need this fix most will find it hardest to take.
3. **Update through Joomla's Extensions manager, or in bulk from mySites.guru.** On a single site, open the Joomla administrator, go to System, then Update, then Extensions, and let Joomla pull 5.8.1. If it does not appear, use Find Updates, or download the package from your JoomDonation account and install it over the top. If you manage more than one site, do not do this one panel at a time: use the mySites.guru mass update feature to upgrade every affected site from a single

dashboard. You can also enable auto-updates for any Joomla extension so the next fix like this reaches your sites without you lifting a finger.

4. **On Joomla 3, go and get 4.9.5.** It is not in the downloads area. Configure your Download ID and Joomla's own updater should offer it in place; if you get a 403, check whether your security extension is blocking the update URL, and fall back to a support ticket.
5. **Confirm the version.** After updating, open Components, then Events Booking, and check the version shown is **5.8.1 or later**.
6. **Check your upload folder.** Look in `media/com_eventbooking/files` for anything that does not belong to a real registration. If your site has been running an affected version on a public registration form, this is where anonymous uploads went. 5.8.1 ships an orphan file cleanup tool for exactly this.
7. **Clear caches.** Clear Joomla's cache and any CDN or page cache.

Updating closes the door. It does not undo any harvesting or uploading that already happened, so if you run a large registrant list, see the data protection point above.

How do I find every Events Booking site I manage?

The awkward question after any extension security release is which of your sites actually run the thing. Up to about ten sites you can log into each Joomla admin and read the installed extensions list. Past that you need a single view, which is why those version numbers get so old: the work of checking scales with the number of sites, so it stops happening.

mySites.guru keeps a live inventory of every extension, template and framework on every Joomla and WordPress site in your account. Search for Events Booking once and get back every connected site running it, the version each one is on, and whether an update is available. That is the same query we ran to produce the version table above, and you can run it against your own sites in about ten seconds.

View every Events Booking install across your sites

Open your Extension Inventory

Search for Events Booking across every connected Joomla site and filter for anything below 5.8.1. Not a subscriber? Sign up free and connect your sites.

Once you know which sites need it, the mass updater handles the rollout. When a fix ships we add the affected range to our vulnerability database, so every connected site still on a vulnerable version of Events Booking is flagged automatically. You should not have to read about a Joomla security issue in the news and then audit a portfolio of client sites by hand to find out whether it is your problem.

Booking and registration extensions deserve a closer look

Every extension that accepts input from anonymous visitors is attack surface, but registration and booking components bring a heavier consequence than most, and it is worth being explicit about why.

A page builder flaw leaks your site. A booking component flaw leaks your customers. The whole purpose of the extension is to collect personal details from the public and store them, which means the data at risk was never yours in the first place, it was entrusted to you. The same logic applies to the newsletter tools, form builders and document managers where we keep finding this pattern: AcyMailing, Balbooa Forms, EDocman, DPCalendar. They are all built to talk to strangers, so a missing check in one of them talks to strangers too.

None of which is an argument for avoiding these extensions. Events Booking is a capable, long-established component, both issues are now closed, and much of the code we reviewed was in better shape than most of what we audit. The payment verification in particular is done properly, which we cannot say about every Joomla extension that takes money. We were harder on the handling above than on the code, and that distinction is intended: the bugs themselves are ordinary, the sort any busy codebase grows. What is not ordinary is spending several days telling a reporter that an anonymous write path into your customers' web servers is not a security issue.

Stay ahead of the next one

There will be another one, because Joomla runs on thousands of third-party extensions and the ones that accept anonymous input keep producing bugs like this. The hard part was never the update itself. It is knowing a fix exists, knowing which of your sites are affected, and getting to them before someone else does.

That is the job mySites.guru does. It keeps a live inventory of every extension on every Joomla and WordPress site in your account, flags the ones with a known vulnerability, and lets you push the update to all of them from one screen. When something like this is disclosed you see exactly which sites are exposed in seconds, without opening a single admin panel.

Get free email alerts when a Joomla vulnerability breaks

We email a plain-English alert the moment a serious flaw like this one is disclosed, with the affected versions and what to do. No charge, unsubscribe any time.

Subscribe to security alerts

Want the alerts and the tooling to act on them? Start with a free audit on one site and see your full extension inventory, or sign up for mySites.guru to get vulnerability alerts and one-click updates across every site you manage.

What to do right now

- Update every Joomla 4, 5 and 6 site running Events Booking to **5.8.1** or later, one at a time or in bulk from one dashboard
- On **Joomla 3**, get **4.9.5**: set a Download ID and use Joomla's updater, or open a JoomDonation support ticket. It is not in the public downloads area
- Do not stop at 5.8.0. The upload hardening is in 5.8.1, and 5.8.0 shipped as two different builds under the same version number

- Check `media/com_eventbooking/files` for files with no matching registration, and clear them out
- If you are on a 4.x build, plan and test the major version upgrade rather than clicking update on production and hoping
- Do not rely on a firewall for this one. There is no payload to filter, so there is nothing for it to block
- If you manage multiple sites, let mySites.guru show you which ones are still exposed rather than checking by hand
- If you run a large registrant list and were on an affected version for a long time, treat the names and email addresses as potentially harvested and talk to whoever owns data protection where you work
- If you suspect something worse, find any hacked files and backdoors and follow our guide to fixing a hacked Joomla or WordPress site

Disclosure and Severity

6.9
CVSS 4.0

MEDIUM

Our own assessment of the user enumeration; three CVEs assigned by the Joomla CNA

Both flaws are unauthenticated and exploitable over the internet in a single request with no user interaction. The enumeration is limited to reading names and email addresses, and the upload is constrained by the attachment allow-list on a default configuration, which is what keeps both out of the high range.

No login needed | Exploitable over the internet | On by default | Anonymous file write | No firewall mitigation

Field	Detail
CVEs	CVE-2026-58149 (user enumeration), CVE-2026-60024 (default unauthenticated upload), CVE-2026-60025 (missing CSRF token on upload), all assigned via the Joomla CNA

Field	Detail
Component	Events Booking for Joomla (`com_eventbooking`)
Vendor	JoomDonation / Ossolution (joomdonation.com)
Type	Unauthenticated file upload (CWE-434), missing CSRF protection (CWE-352), and authorisation bypass through a user-controlled key (CWE-639)
CVSS 4.0	6.9 (Medium) for the enumeration, `AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N` (our own assessment). No public score yet for the upload CVEs
Impact	Anonymous file writes to the web server, and anonymous read of the name and email address of every user on the site
Finder	Phil Taylor, mySites.guru
Affected versions	5.7.1 and earlier, and the Joomla 3 line below 4.9.5
Fixed in	5.8.0 (both flaws), 5.8.1 (substantial upload hardening), 4.9.5 (Joomla 3, by support ticket)

We follow a fix-first, publish-second process: private disclosure to the vendor, a window for a patch, then a public write-up without a proof-of-concept.

The full timeline is at the foot of this post.

Timeline

2026-07-13 — **We give JoomDonation a heads-up while disclosing a different flaw**

While reporting the unrelated EDocman SQL injection to the same vendor, we mention that Events Booking appears to have two security

issues of its own, pending a closer look. JoomDonation re-reviews the source in response.

2026-07-14

A fix appears in the download package with no changelog entry

The enumeration flaw is quietly patched in the shipped package with no announcement. Customers query the undocumented change, and a fellow Joomla developer emails us having spotted a new build appear that day.

2026-07-15

We disclose both issues in full, copying the Strike Team

We audit the Events Booking 5.7.1 source, confirm both issues and report them privately with all public detail withheld. JoomDonation confirms the enumeration flaw and disputes the upload as invalid. The Joomla Security Strike Team backs the CSRF finding in writing.

2026-07-15

10:02 Events Booking 5.8.0 is published

The announcement credits mySites.guru, states the first issue "was fixed before he reported", and calls the second "not really a security issue". JoomDonation asks us to delay publication by two days. We agree and publish nothing.

2026-07-15

11:02 the announcement is edited three minutes after we challenge it

Both statements are removed, with no correction note.

2026-07-15

16:10 the 5.8.0 package is rebuilt and re-published under the same version number

The changelog thread advises anyone who updated before this time to download and install again. A customer replies on the same thread that the release "does not appear to fix the issues".

2026-07-17

Three CVEs are assigned and the vendor confirms the default to the Strike Team

JoomDonation confirms the upload was enabled by default before 5.8.0 and that several additional protections have been implemented. The

Joomla CNA assigns CVE-2026-58149, CVE-2026-60024 and CVE-2026-60025, all crediting Phil Taylor of mySites.guru.

2026-07-20 ● **5.8.1 ships, and we publish**

JoomDonation emails customers calling the upload issue "a security improvement rather than a vulnerability", with no CVE numbers in the mail. We diff the 5.8.0 and 5.8.1 packages, confirm the fixes and the hardening, flag every connected site below the fix, and publish this write-up with no proof of concept.

2026-07-20 ○ **We privately report a third, separate flaw in 5.8.1**

An insecure direct object reference in the invoice download, held back pending a fix. It became CVE-2026-63047, fixed in 5.8.2 the following day.

Further Reading

- These Events Booking flaws were part of a month of Joomla extension vulnerabilities we found and disclosed - the full roundup of the run.
- Why AJAX endpoints are a CMS security blind spot - the recurring public-endpoint pattern that produced both of these findings.
- Our EDocman SQL injection disclosure - the same vendor, a different bug class, and a changelog that badly understated it.
- When a vendor abandons Joomla 3 - for contrast, what happens when the Joomla 3 fix is never written at all.
- CWE-434: Unrestricted Upload of File with Dangerous Type, CWE-352: Cross-Site Request Forgery and CWE-639: Authorization Bypass Through User-Controlled Key - the canonical references for these weakness classes.
- ISO/IEC 29147 and ISO/IEC 30111, the CERT Guide to Coordinated Vulnerability Disclosure, FIRST's multi-party coordination guidance and the OWASP

Vulnerability Disclosure Cheat Sheet - the standards and practitioner guides behind the disclosure process described above.

- Events Booking product page - where 5.8.1 is available to download.

Frequently Asked Questions

What are the Events Booking vulnerabilities?

There are two, both unauthenticated, both found by mySites.guru in Events Booking for Joomla by JoomDonation. The first is a file upload endpoint that accepted files from anonymous visitors, on by default on every install, with no login and no CSRF token. The second is an access control flaw where a public endpoint returned any user's name and email address for any user ID it was handed. Three CVEs were assigned via the Joomla CNA: CVE-2026-60024, CVE-2026-60025 and CVE-2026-58149. Both issues were addressed in Events Booking 5.8.0, with substantial further hardening in 5.8.1.

Which Events Booking version do I need?

Update to Events Booking 5.8.1 or later. 5.8.0 fixed the two reported flaws, but 5.8.1 adds the upload rate limiting, per-field permission checks, image content validation and randomised filenames that 5.8.0 lacked. 5.8.0 also shipped as two different builds under the same version number, so skipping straight to 5.8.1 avoids the ambiguity. If you run Joomla 3, the fixed version is 4.9.5, which is not in JoomDonation's public download area: configure a Download ID and use Joomla's own updater, or open a support ticket to request the package.

Was the anonymous file upload really enabled by default?

Yes, and the vendor confirmed it in writing to the Joomla Security Strike Team: 'It was enabled by default in previous versions. From 5.8.0, it will only be enabled if admin creates custom field to allow registrants to upload file when they register for the event.' The Joomla CNA assigned CVE-2026-60024 for precisely this, describing it as Events Booking prior to 5.8.0 allowing unauthenticated users to upload media assets by default.

Could this have led to remote code execution?

Not on a default configuration. The upload is filtered against the configured attachment allow-list, and the shipped default of bmp|gif|jpg|png|swf|zip|doc|pdf|xls contains no PHP extension, so an attacker could not simply upload a web shell. What they could do was write files to your server without an account, and then have the site serve them back to anyone: fill your disk, host their own content on your domain and your hosting bill, distribute malware in the zip, doc and xls types the default list permits, or plant illegal material on infrastructure registered in your name. On a site whose administrator has widened the allow-list to an executable type, it becomes straightforward unauthenticated remote code execution.

The vendor's email says this is a security improvement, not a vulnerability. Who is right?

The Joomla CNA settled it independently by assigning CVE-2026-60024 for the default unauthenticated upload and CVE-2026-60025 for the missing CSRF token, three days before that email was sent. Parts of the email are accurate: on a default configuration this is not remote code execution, because the shipped attachment allow-list contains no PHP extension, and we have said so throughout. The absolute claims are the problem. The allow-list is an administrator-editable setting, so 'attackers could not upload executable files' is true of the defaults rather than of every install, and the same 5.8.1 release added image content validation and a nosniff header to the file-serving path, which are defences against the scenario the email says could not happen.

Does 5.8.1 fully close the anonymous upload?

No, and it is not meant to. Public registration forms need to accept uploads from people who have not logged in, so the endpoint remains reachable by guests where a File field is set to Public access. What 5.8.1 adds is limits: 100 files per session and 20 per minute by default, optional folder size and file count caps that ship disabled, real image validation, and randomised filenames. The per-session limits are enforced against a session cookie, so an attacker who discards their cookie between requests gets a fresh allowance.

How do I find every Events Booking site I manage?

mySites.guru keeps a live inventory of every extension and version on every connected Joomla and WordPress site. Search for Events Booking once and get back every site running it, the version each is on, and whether an update is available. Every connected site still on a vulnerable version is flagged automatically against our vulnerability database, so you do not have to audit a portfolio of client sites by hand.

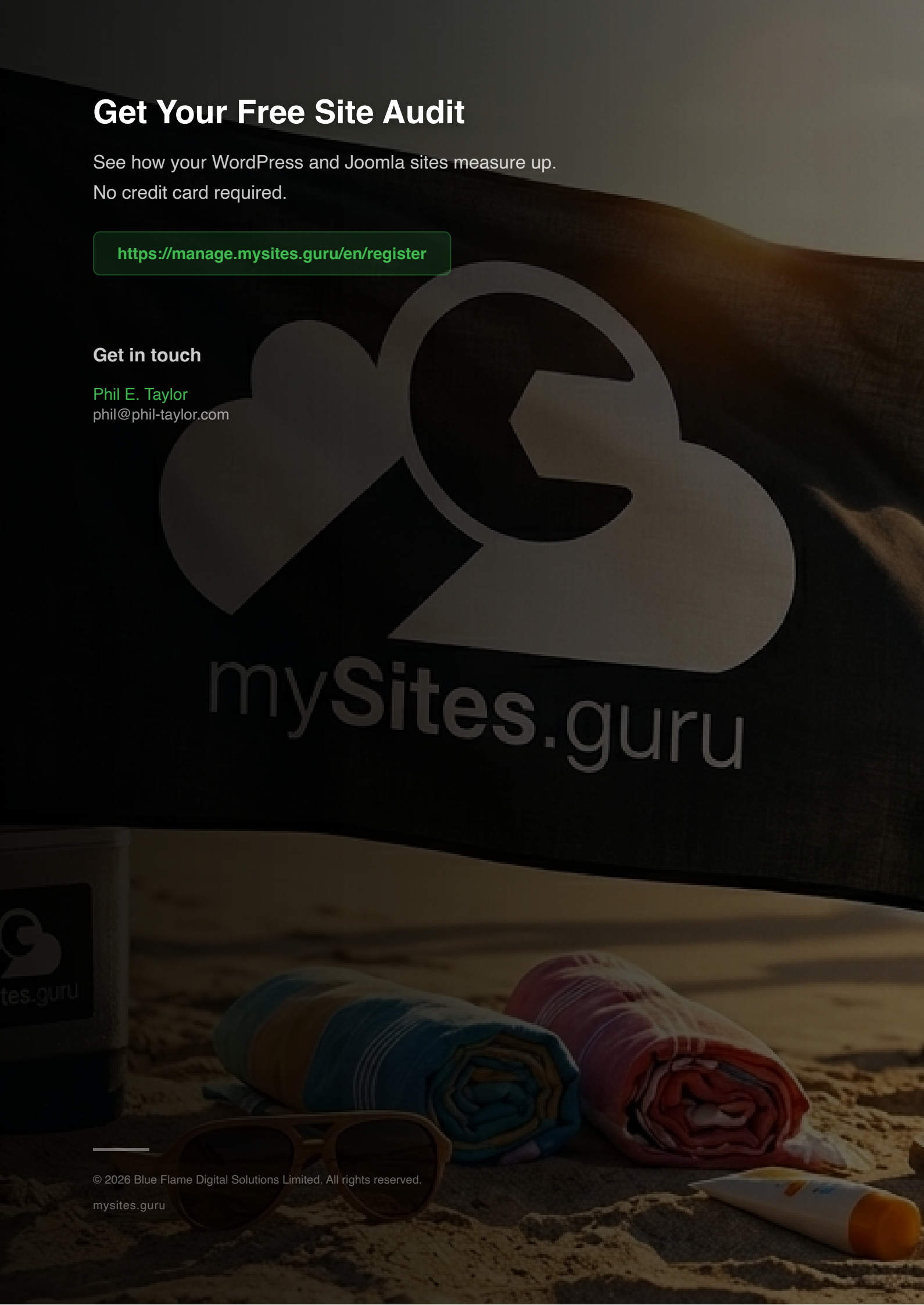
Written and edited by a human, with AI assistance. Our approach to AI

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

© 2026 Blue Flame Digital Solutions Limited. All rights reserved.