



PHP2Shell ModSecurity Fix Blocking WordPress and Joomla Saves on cPanel

The cPanel ModSecurity fix for PHP2Shell turns on body inspection server-wide, so old rules block WordPress, Elementor and Joomla saves. How to check and fix.

Phil E. Taylor | 26 October 2026

This post is for server administrators running WHM/cPanel, or anyone running ModSecurity with the OWASP Core Rule Set. If you look after a site on shared hosting rather than the server itself, the symptoms below will still help you recognise the problem, and the question to put to your host is at the end.

If your editors started getting blocked when saving in WordPress, Elementor or Joomla on a cPanel server in the days after 21 October 2026, and you or your host applied a ModSecurity mitigation for PHP2Shell, this is very likely why. The mitigation itself is sound. One line in it, `SecRequestBodyAccess On`, changes what the rest of the firewall inspects, and the side effect is that the OWASP Core Rule Set starts reading editor saves it had never seen before. Background on the two PHP flaws is in [our earlier post on the PHP security release](#).

We know because it happened on two cPanel servers we look after, both hosting a mix of WordPress and Joomla sites. We applied the mitigation on 21 October, and over the following five days editors on both servers found they could not save. We found it, traced it, and fixed it with narrow exclusions that keep the front end fully protected. As of 26 October 2026 we have found no public report connecting the mitigation to broken saves, so this is the write-up we wanted to read on day two.

Quick answer

The PHP2Shell mitigation for cPanel adds `SecRequestBodyAccess On` to `modsec2.user.conf`. ModSecurity 2 defaults that setting to Off, so turning it on makes every rule on the server inspect every POST body for the first time, including cPanel's OWASP Core Rule Set 3.0.2. Old XSS and injection rules then fire on WordPress, Elementor and Joomla saves and deny them, and five of those within an hour gets the editor's IP banned by CSF. The fix is a handful of narrow rule exclusions on the admin save paths, further down this page.

What the Blocked Saves Look Like in WordPress, Elementor and Joomla

The symptoms differ by CMS, and the most serious one does not look like a firewall problem at all.

In WordPress, editor saves go through `/wp-admin/admin-ajax.php` and fail. ModSecurity logs a 403, but cPanel serves the deny using the site's own 404 page, which makes Elementor the hardest to diagnose: clicking Publish shows "Server error 404 not found" for a page that exists.

In Joomla, saving an article in the administrator fails, and so does saving Global Configuration > Permissions. Both go through `/administrator/index.php`, and both are blocked.

The worst case is the site owner whose whole site appears to be down. It will not load on their laptop. It will not load on their phone, on the same home broadband. Everyone else can see it. CSF and Ild have counted the ModSecurity denies from their IP and banned it at the firewall, so from their house the server no longer answers at all. More on that below.

Why Did SecRequestBodyAccess On Break Editor Saves?

Because it turned on request body inspection for every rule on the server, not only the new one. The [cPanel mitigation](#) that circulated during the emergency adds a ModSecurity rule, id 2026102101, which returns a 403 for any multipart request with a part named `PHP_SESSION_UPLOAD_PROGRESS`. Blocking that field at the web server was a reasonable emergency measure. The rule cannot see the field unless ModSecurity reads the request body, and WHM has no switch for that, so the gist tells you to add `SecRequestBodyAccess On` to `/etc/apache2/conf.d/modsec/modsec2.user.conf`.

ModSecurity 2 defaults `SecRequestBodyAccess` to Off, and the directive is server-wide. Once it is on, every rule on the server can read POST bodies, including cPanel's OWASP3 vendor rule set, which on our servers is OWASP Core Rule Set 3.0.2. Those rules were always loaded; they had never had a body to look at. From 21 October they

were reading every POST, page builder saves and editor saves included, for the first time on those servers.

So the rules that blocked our editors are old rules, not new ones, and what changed is what they could see. The upstream OWASP CRS project has been adding PHP2Shell detections this week, but those go into CRS 4.x, not into cPanel's 3.0.2 feed. The 3.0.2 rules files have a 2006–2016 copyright header, and cPanel's distribution of them on our servers was published on 30 September 2026 and has not changed since.

The gist is not alone. The same advice is now all over the web.

Multiple Articles on the Web Recommend SecRequestBodyAccess On

Every source below was published on 21 October 2026, the day of the PHP release, and each one sets or requires `SecRequestBodyAccess On` so a rule can see the upload-progress field. None of them warns that the directive also changes what every other rule on the server inspects, and only the first is written for cPanel. We checked each one on 26 October.

- [The cPanel gist](#) (GitHub, 09:38 UTC): "WHM has no switch for request-body inspection. Without `SecRequestBodyAccess On`, the new row still lets this field through." Its script inserts the line as line 1 of `/etc/apache2/conf.d/modsec/modsec2.user.conf`, which is server-wide. It has an Undo section, but no warning about side effects.
- The most-starred PHP2Shell proof-of-concept repository on GitHub (over a thousand stars; we are not linking an exploit repository): its `modsecurity-php2shell.conf` opens with `SecRequestBodyAccess On`. Its only caveat is that a WAF parses multipart differently from PHP, so the rule is "an extra layer, not a silver bullet".
- [PasqualePillitteri/php2shell-hardening](#) (GitHub): the same directive at the top of its ModSecurity file, with "Test every rule in DetectionOnly mode first." That is the closest any of them comes to a warning.

- [Lexsah/php2shell-detection](#) (GitHub): "Requires SecRequestBodyAccess On for the body rules." Its rules ship in log-only mode, with advice to switch them to deny after a tuning period.
- [ZEST Security](#): "Both rules need request body inspection on (SecRequestBodyAccess On)."
- [PenTestIT](#) (updated 26 October): "This rule needs SecRequestBodyAccess On." It reports that normal uploads and page loads still returned 200 in its testing.
- [CVE Tools](#): its ModSecurity rule is marked "(needs SecRequestBodyAccess On)".
- [A NodeSeek forum post](#), a Chinese-language rewrite of the PenTestIT rule that lists turning on **SecRequestBodyAccess On** as a deployment requirement.

The one different approach we found is [plob881/php2shell-shield](#) (GitHub, in Dutch), which does not set the global directive. Its rule uses **ctl:requestBodyAccess=0n** in phase 1 to turn body access on per request, only for multipart requests, and for cPanel it goes in **/etc/apache2/conf.d/includes/pre_main_global.conf**. Scoped that way, any save that is not a multipart request should stay outside the Core Rule Set's body inspection. We have not tested it on our servers.

None of these authors did anything unreasonable. A WAF rule for an unpatched remote code execution flaw has to see the request body, and on the day of the release that was the priority. The side effect shows up when the server also runs the OWASP Core Rule Set, which on WHM/cPanel is the OWASP3 entry under ModSecurity Vendors.

Which Core Rule Set Rules Fired, and on What

Every block below is a 949110 anomaly-score deny: the Core Rule Set adds up the scores of the individual rules that matched and blocks once the total crosses the threshold.

What was being saved	CRC
WordPress editor save, <code></code> markup in <code>ARGS:html</code>	941130 (script tags, encoded on the fly)
Elementor editing a page with Vimeo <code><script></code> embeds, <code>ARGS:actions</code>	941130 941130 941130
Elementor Publish (<code>save_builder</code>), <code>ARGS:actions</code>	941130
Joomla article save, <code>ARGS:jform[articletext]</code>	941130 XSS rule and 930120 on the product system ("
Joomla Global Configuration > Permissions, the field name <code>jform[rules][jce.profiles][5]</code>	930120

Two of these need explaining. Rule 941130's keyword list includes `xmlns` , `base64` , `formaction` and `@import` , words that turn up in ordinary page builder data (an inline SVG icon has `xmlns` , an embedded image has `base64`), and it was not in our first exclusion, which is why Publish failed the morning after editing had been fixed. Rule 930120 is a list of operating system file names, and `.profile` matched the field name `jform[rules][jce.profiles][5]` , not any value an editor typed.

On the large Elementor payloads, rules 941140, 941310, 942190 and 942240 also logged "PCRE limits exceeded" and fail open. That is noise in the log, not the cause of the block.

How CSF and Ifd Turn a Blocked Save Into a Site That Looks Down

On both of our servers CSF has `LF_MODSEC = 5` , `LF_MODSEC_PERM = 1` and `LF_INTERVAL = 3600` . That tells Ifd to block an IP at the firewall, permanently, after five ModSecurity denies from it within an hour. One Elementor editing session produced five denies in about 25 seconds. From that moment nothing from the editor's house reached the server: laptop and phone on the same home broadband both failed, which looks exactly like the site being down.

Meanwhile our uptime monitoring, which checks from outside, saw a 200 every five minutes all evening, and other members used the site normally. That external check told us within minutes that the site was up and the problem was one visitor's IP, which turned a "the site is down" email into a `csf -g` lookup. Our uptime monitoring explainer already covers allow-listing the monitor in ModSecurity, Fail2Ban, Imunify360 and CSF, and the same allow-listing applies to a regular editor's home IP.

How Do I Check Whether My cPanel Server Is Affected?

These commands answer it: denies on the admin save paths, the rules and arguments that fired for one editor's IP, whether CSF has banned that IP, and whether the rule set itself changed.

```
# 949110 denies on WordPress and Joomla admin save paths
grep -a 'id "949110"' /etc/apache2/logs/error_log | grep -aE 'uri "[^"]*(a

# which rules fired, and on which arguments, for one editor's IP
grep -ah "EDITOR_IP" /etc/apache2/logs/error_log | grep -ao '\[id "[0-9]*'
```

```
# is that IP banned by CSF?
csf -g EDITOR_IP

# did the vendor rule set actually change? (look for "already up to date")
grep -a modsec_vendor /usr/local/cpanel/logs/error_log | tail

# which Core Rule Set version cPanel is serving
head -3 /etc/apache2/conf.d/modsec_vendor_configs/OWASP3/rules/REQUEST-941
```

The fourth command is the one we should have run first. cPanel's nightly vendor check writes "The vendor OWASP3 is already up to date" to that log, and it also gives the rules files a new modification time every night, whether or not anything changed. A rules file dated last night is not evidence of a rule update. We were caught by exactly this.

The Fix We Used: Narrow ModSecurity Exclusions for WordPress and Joomla

The exclusions go in `modsec2.user.conf` after the gist's lines. They run in `phase:1`, so the `ctl` actions apply before the phase 2 Core Rule Set rules run, and they are scoped to the admin save paths, so the front end keeps full protection.

```
# Joomla administrator saves
SecRule REQUEST_FILENAME "@endsWith /administrator/index.php" \
    "id:2600100,phase:1,pass,nolog,t:none,\
    ctl:ruleRemoveById=933160,\
    ctl:ruleRemoveById=941100,\
    ctl:ruleRemoveById=941110,\
    ctl:ruleRemoveById=941160,\
    ctl:ruleRemoveById=941200,\
    ctl:ruleRemoveById=941350"

# Joomla Global Configuration permissions: 930120 on field names only
SecRule REQUEST_FILENAME "@endsWith /administrator/index.php" \
    "id:2600102,phase:1,pass,nolog,t:none,ctl:ruleRemoveTargetById=930120,"
```

```
# WordPress editor AJAX (ARGS:html, and Elementor's ARGS:actions), logged-
SecRule REQUEST_FILENAME "@endsWith /wp-admin/admin-ajax.php" "id:2600101,
    SecRule REQUEST_HEADERS:Referer "@contains /wp-admin/" "chain"
        SecRule REQUEST_COOKIES_NAMES "@beginsWith wordpress_logged_in_" \
            "ctl:ruleRemoveTargetByTag=attack-xss;ARGS:html,ctl:ruleRemove"
```

Then `apachectl -t && apachectl -k graceful` .

We started the WordPress rule as a list of rule IDs and kept finding the next one: 941130 blocked Elementor Publish the morning after editing was fixed. So we switched to removing the whole `attack-xss` tag from those two arguments only. Test publishing as well as editing, because they are different requests.

The cookie condition checks the cookie name, not its value, so it narrows the exclusion rather than authenticating anyone; WordPress still checks the nonce and the login. We verified it on the server: a logged-in-style `save_builder` payload containing `<svg xmlns=...>` and a `data:image/png;base64` image produced no ModSecurity log lines, and the same payload without the cookie was still denied (941100, 941130, 941160, 941170). For 930120, a field name containing `.profile` now passes, while a field value of `/home/user/.profile` is still denied.

Finally, allow-list a regular editor's home IP in CSF with `csf -a` and add it to `csf.ignore` so lfd cannot ban it.

When Can SecRequestBodyAccess On Come Out Again?

Once every PHP version serving sites on the server is on a fixed release (8.2.35, 8.3.36, 8.4.27, 8.5.12 or later), removing `SecRequestBodyAccess On` as described in the gist's Undo section removes these false positives at a stroke. It is a trade-off: body inspection also lets the Core Rule Set catch real attacks in POST bodies.

Keep body access on while any end-of-life ea-php still serves sites

cPanel does not fix end-of-life PHP builds such as ea-php81. If any site on the server still runs one, the ModSecurity rule is one of the few layers left between it and the upload-progress exploit path, and the rule does not work without `SecRequestBodyAccess On`. Use the exclusions above until the last end-of-life PHP is gone, then remove the line.

The PHP-side mitigation, `session.upload_progress.enabled = 0`, is unaffected by any of this and should stay in place: [the advisory post](#) has the one-line change. And [PHP is not out of the woods yet](#), which is the case for leaving mitigations where they are for now rather than unwinding them early.

If you look after several cPanel servers or sites across several hosts, [the agency post](#) covers raising this with each host as one ticket per host. The thing to ask is simple: did you add `SecRequestBodyAccess On` for PHP2Shell, and if so, what exclusions did you put in for WordPress and Joomla admin saves?

Further Reading

- [The cPanel ModSecurity gist for the upload-progress field](#) - the mitigation discussed in this post, including its Undo section.
- [OWASP Core Rule Set issue tracker](#) - where the upstream project's PHP2Shell detections for CRS 4.x are being discussed.
- [CloudLinux on PHP2Shell and Imunify360](#) - the view from a hosting stack that ships its own web application firewall.
- [Beaver Builder: 403 Forbidden or Blocked error](#) - a page builder vendor's own documentation of ModSecurity blocking saves, the same class of problem from the other side.

Not sure whether a host-side change is what is blocking your editors? A [free audit](#) of one site reports its PHP version and hosting environment, and uptime monitoring tells you within minutes whether a site is down for everyone or for one IP.

Frequently Asked Questions

Why does Elementor say "Server error 404 not found" when I click Publish?

On a cPanel server, ModSecurity's deny is served using the site's own 404 page, so a blocked Publish request shows up in Elementor as a 404 rather than a 403. Check the Apache error_log for rule 949110 on /wp-admin/admin-ajax.php at the time of the click; if it is there, the firewall blocked the save.

Did a ModSecurity rule update cause the blocked saves?

Not on our servers. The rules that fired are old OWASP Core Rule Set 3.0.2 rules that had never inspected a POST body before. Adding SecRequestBodyAccess On for the PHP2Shell mitigation let them read every POST for the first time. The rules files get a new modification time every night even when nothing changes, so file dates are not evidence of an update; the cPanel log line "The vendor OWASP3 is already up to date" is.

My site is down on my laptop and my phone but fine for everyone else. What happened?

Most likely CSF banned your home IP. With LF_MODSEC = 5, five ModSecurity denies from one IP within an hour gets that IP blocked at the firewall, and one page builder editing session can produce five in under half a minute. Ask whoever runs the server to run csf -g on your IP, and to allow-list it so lfd cannot ban it again.

Should I just remove SecRequestBodyAccess On?

Only once every PHP version serving sites on the server is on a fixed release (8.2.35, 8.3.36, 8.4.27, 8.5.12 or later). While any end-of-life ea-php such as ea-php81 still serves sites, keep it and use the narrow exclusions in this post instead. The PHP-side setting session.upload_progress.enabled = 0 is unaffected and should stay either way.

AI MODIFIED

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

mySites.guru

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

mysites.guru