# CVE-2021-35368 – CRS Request Body Bypass (Update)

By **Christian Folini** / June 30, 2021

There is a severe security issue in our rule set. It has been present since the release of CRS 3.1.0 and was recently brought to our attention.

Here is the official advisory that we are also publishing as CVE-2021-35368 via MITRE (as usual, MITRE will take a few days until they publish this).

## Offical Advisory for CVE-2021-35368

**The OWASP ModSecurity Core Rule Set (CRS) is affected by a request body bypass that abuses trailing pathname information. A backend vulnerability can thus be exploited despite being protected with the CRS Web Application Firewall rule set when an application server accepts additional path info as part of the request URI. All known CRS installations that offer the predefined CRS rule exclusion packages are affected. This applies to end-of-life CRS versions 3.0.x, 3.1.0, 3.1.1 as well as the currently supported versions 3.2.0 and 3.3.0. Integrators and users are advised to upgrade to 3.1.2, 3.2.1 and 3.3.2 respectively.**

## Link to releases, Release Notes, etc.

* **CRS v3.3.2 release** / **CHANGES**
* **CRS v3.2.1 release** / **CHANGES**
* **CRS v3.1.2 release** / **CHANGES**

Please note that **CRS 3.3.1 was never released**. We started with the release process, but we stopped after encountering severe problems with the first release candidate. CRS 3.3.2 builds on 3.3.0 again.

The Daily Swig has also **covers this vulnerability in an article**.

## CVSS Score (8.0 or higher)

It is tricky to assign a CVSS score to a bypass when exploitability depends on backend settings and weaknesses on the backend. If the backend is broken and configured with the correct trailing pathname information setting (more on that further down below), then anything is possible. If the backend looks into the trailing path info as it should, then you are on the safe side.

So depending on the backend, the CVSS is very high, certainly 8.0 or even higher.

## The Vulnerability in Detail

First, let's explain what trailing pathname information is.

Here is a simple URI path used by a PHP server:

```
/cms/index.php
```

This path does not have any trailing pathname information.

Now look at this:

```
/cms/index.php/admin/content
```

Here, */admin/content* is the trailing pathname information, short: path info. PHP will execute the code in */cms/index.php* and decide on the exact behavior based on the path info.

Very often, servers attempt to beautify the path as follows:

```
/cms/admin/content
```

Here, the */index.php* is hidden and the path info is again */admin/content*. If you are on the backend, you can make that distinction. But for the web application firewall in front, it's all part of the path. In fact, ModSecurity and CRS are not making the distinction between path and path info at all unless they run on the webserver.

With that being clear, let's look at the vulnerability.

CRS introduced rule exclusion packages in 3.1.0, short REs. REs are ModSecurity rules that are loaded, but disabled by default. Disabling works via a skip mechanism that is being applied unless you set a variable that tells ModSec you want to execute the REs in question.

As you probably know, ModSecurity runs in multiple phases. The request rules happen in phase 1 and in phase 2. It's like the rule set would be executed twice, once for the phase 1 rules, and once for the phase 2 rules. So if you want to skip a part of the rule set – the REs! – then you need to define a skip for phase 1 and a separate skip for phase 2. All the REs do that. Well, all the REs with one exception: The Drupal RE package in REQUEST-903.9001-DRUPAL-EXCLUSION-RULES.conf lacks the skip for phase 1. This is a bug that sadly evaded our review process, which was still maturing around the 3.1 release stage.

This results in a situation where the Drupal REs in phase 1 are always active, whether you have enabled them or not.

However, if your installation comes without the RE files or this particular RE, then you are on the safe side. In fact, many of the commercial CRS offerings do not include REs.

Okay, so far we are in a situation where too many rules are executed which brings a minimal performance impact. 3-4 percent or so, I guess. But it's getting worse. The problem is that on top of this blunder, three Drupal RE rules are playing with fire: They *disable request body scanning* for a handful of specific requests: 9001180, 9001182 and 9001184. There is a comment in the code that reads "Extensive checks make sure these uploads are really legitimate." Extensive: yes, comprehensive: no.

We think it's a good idea to install your software into a sub-folder. Like installing Drupal under */cms*, since this will hide it from a lot of security scanners that do not even follow redirects. However, this also leads to a situation where CRS won't know your exact installation path. That's why the extensive tests quoted in the comment can not be exact. CRS opted for anchoring the pattern at the end like this regular expression: */admin/content/assets/add/[a-z]+$*. The URI path starts with anything and ends with this pattern.

As we have seen above, this could be a path info in a specific URI. If an application outside Drupal looks at the path info, then this would result in a 404 or some other error. However, if the application ignores the path info, then it will simply execute the request with its base URI path. CRS in front of the application can not know this, so it has to assume it's a single URI path and interprets it accordingly, assumes it's facing a request to Drupal *content->assets* and disables request body access. That means the request will pass without the request body being inspected by the WAF. But when the backend

receives the request, it will throw away the path info and we have a WAF Bypass!

This is because the WAF and the application don't have the same perception of the URI and they act differently based on the request. It's a typical web application firewall problem, but here it is impacting us in an exemplary way.

Why do the Drupal REs disable the rule engine anyways? All the other REs are more conservative in their approach. Honestly, we do not remember. It probably has to do with a lack of time and with a lack of review.

When attempting to fix this double shortcoming, we tried to reproduce the requests that the rules 9001180, 9001182 and 9001184 address. However, we failed. Despite clicking around in Drupal for hours, we could not issue a request meeting the regular expressions of said rules. It is therefore very likely the requests are aimed at a particular Drupal module and are not standard Drupal requests. It could be a popular module, but our Drupal RE actually only targets the Drupal core installation, so it is possible that we saw false positives on an installation that included a few rare modules and we wanted to get rid of it quickly when we were planning to release the first batch of rule exclusions.

## The Fix

Given we can not reproduce the requests behind 9001180, 9001182 and 9001184, we do not know the individual false positives that these rules try to avoid anymore. We can not continue switching off the request body access either, so removing these rule exclusions seems like the best option as of this writing.

For Drupal users, this might mean that you install the update and face new false positives. If that's the case, you will need to tune them away yourself. If you happen to encounter these URIs on your Drupal installation and you find false positives that you connect to these URIs, then please get in touch via our GitHub issues page; we would be really interested in more information here.

So rules 9001180, 9001182 and 9001184 are being removed. And we are adding a phase 1 skip. We try to bring the Drupal REs in sync with the other REs. So we shift the existing rule 9001000 to phase 1 and add a new rule 9001001 in phase 2 to perform the RE skip in phase 2.

**Update:** We are not releasing an update for the 3.0.x release line that has been EOL for several years. If you are stuck on version 3.0 and you do not need the Drupal rule exclusions, then simply remove the file `REQUEST-903.9001-DRUPAL-EXCLUSION-RULES.conf`. If you do need them, then simply copy the **Drupal rule exclusion file** from the new version over your old file.

## A Word About CRS Rule Exclusions

Any Web Application Firewall worth its investment comes with false positives (FPs) for certain requests. If it does not, then it's probably because it lacks teeth. At CRS, we try to fight FPs with multiple approaches. Improving the detection patterns is an obvious one. Rule exclusion packages are another one. With an RE we accept that some rules are being triggered by a certain application. We group these rules together and assemble a configuration that disables these rules from being executed for this application or for certain parameters on certains URIs of the application. That's what we call a rule exclusion package.

In CRS v3.3, we included REs for

- Drupal
- WordPress
- Nextcloud / Owncloud
- Dokuwiki
- CPanel
- Xenforo

and more of them are in the making for the next release.

When we introduced REs with 3.1, we kind of hoped that the communities and vendors behind these tools would embrace the REs and help us with maintenance, maybe help to extend them to modules, etc. Ideally, other communities would then get in touch and ask how they can contribute REs for their software and more and more traction would build up.

Yet it seems, that's all been wet dreams of open source developers. Meanwhile we are aware that the REs are somewhat outdated and still very limited. Yet very few contributions happen and we are struggling to maintain the ones we have, especially when nobody in the CRS development team uses the software in question. Necessary reviews of existing REs are not happening regularly either, so the situation is unsatisfying and we need to do something about this.

## Indicators of Compromise

We are are not aware of this weakness being exploited in the wild. You should be able to find out for yourself by checking for the following patterns in your logfiles.

If you are not running Drupal, yet you see successful HTTP status codes for the following URI paths,

then you might be facing a successful exploit:

- `/admin/content/assets/add/`
- `/admin/content/assets/manage/`
- `/file/ajax/field_asset_`

If you are able to confirm an attack (that is the requests above against non-Drupal installations) regardless of success or not, then we would be very interested to learn about it.

## Thanks

The CRS team would like to thank Andrew Howe from Loadbalancer.org for bringing this weakness to our attention. Andrew has been very helpful discussing the impact and fix for this vulnerability and we hope to work with him in the future.

Meanwhile Andrew has also written a separate **blog post** about this vulnerability.

## Timeline

- 2021-06-10: Initial message by Andrew Howe
- 2021-06-10: Confirmation of vulnerability
- 2021-06-15: Vulnerability submitted to MITRE
- 2021-06-23: Announcement of upcoming security release to known integrators
- 2021-06-26: Mitre assigns CVE-2021-35368
- 2021-06-28: Announcement of upcoming security release to community
- 2021-06-30: Release of CRS 3.1.2, 3.2.1 and 3.3.2.

If you are an integrator of CRS and you would like us to put you on our list for future communication, then please get in touch.

Christian Folini for the CRS team

## Updates to this page

- 2021-07-01: Added link to Andrew Howe's blog post.
- 2021-07-01: Added releases 3.0.x to the list of vulnerable releases and a remark under Fix how to address this.
- 2021-07-01: Updated the blog post to mention that CRS 3.0 (out of support) is also affected.

- 2021-07-06: Link to Daily Swig article

← Previous Post

## Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment »