interweb

Generated on 15/11/2026 00:08

KNOWLEDGE BASE · PUBLIC DOCUMENT

Do exec and shell_exec work here? Why no PHP function is disabled

PHP Versions & Settings

CODE	VERSION	COUNTRY · LANGUAGE
RLX-AO-KB-342	v2026.09.28	Angola · English

Interweb Angola
RLX Angola, Lda. · Group RLX Holdings
Tax ID (NIF) 500 260 5724 · Commercial registration no. 26464-25/250724

https://www.meu.interweb.ao/knowledgebase/342

Do exec and shell_exec work here? Why no PHP function is disabled

PHP Versions & Settings Created on **September 11, 2026** Updated on **September 28, 2026** 4 min read



Anyone installing a tool that converts images, generates PDFs or shells out to some command eventually hits this question: *does* `exec()` *work on shared hosting?* With most providers, no. **Here it does** - and the reason it does is more interesting than the answer.

The usual practice, and why we do not follow it

Almost every shared host disables a set of PHP functions - `system`, `exec`, `shell_exec`, `passthru`, `proc_open` - through a list called `disable_functions`. The idea is simple: if PHP cannot run commands, a compromised site cannot run commands.

The trouble is that the list is a remedy for one specific illness: **servers where every account shares the same filesystem**. There, a loose `shell_exec` could go and look at the neighbours' folders, and the blacklist becomes the last line of defence. It was never a good defence - it was just the one available.

What we have instead: a jail per account

Our servers run **CloudLinux**, and each account runs inside its own **jail**. That changes the nature of the problem: a script of yours running a system command **only sees your account**. It does not see other customers' folders, it does not see the server's user list, and it cannot reach the system configuration.

A disabled-function list	A per-account jail
Forbids some functions	Limits **what can be seen at all**, whichever function is used.
Can be worked around	There is always another way to run a command; a blacklist chases function names, not the problem.
Breaks legitimate software	Breaks nothing. Tools that need to run commands run.
Protects the server	Protects the server **and** separates customers from one another.

So here **no PHP function is disabled** - and you do not need to ask for anything to be enabled. If you came from a provider where you had to ask, you can simply use it.

What this unlocks, in practice

What now works	Everyday examples
Image tooling	Conversions and resizing that call ImageMagick on the command line rather than the extension.
PDF and document generation	Libraries that invoke an external binary to compose the file.
Backup plugins	The ones using system commands to compress and export databases - usually a good deal faster that way.
Your own cron scripts	Scheduled tasks calling utilities - see **which technologies run here**

Before assuming you need a command.
Many libraries ship two paths: one using a PHP extension and one calling a binary. The first is nearly always faster and more predictable. Check whether the extension you need is enabled before reaching for `exec()` - see **viewing and adjusting your PHP configuration**.

What the jail still prevents

«No functions disabled» does not mean «access to everything». There are still limits, and they are what makes this safe:

1. **You only see your own files.** Leaving your account is not possible, with any function.
2. **You only run permitted binaries.** The jail holds a curated set of tools - not the whole system.
3. **You cannot install software into the system.** For that you need a server of your own - see **shared hosting vs VPS**.
4. **Resources stay capped per account**: processor, memory, number of processes and disk access. A heavy command hits those limits, not the neighbours.

There are two memory limits, and confusing them costs time.
PHP's `memory_limit` is **per request**; the account limit covers **everything running at the same time**, including commands your script launches. Raising the first does not lift the second - the difference is explained in **PHP limits**, and real consumption is visible in **monitoring performance in cPanel**.

The part that stays yours

The jail protects the **other** customers from you, and protects you from them. What it does not do is protect you from yourself: an out-of-date plugin able to run commands can still wreck **your** account. The usual rules still apply - see **what we do about security and what stays on your side**.

1. **Never pass something that came from a visitor into a command** without validating it. That is the classic way to turn a form into somebody else's control panel.
2. **Use the language's escaping functions** before assembling any command.
3. **Keep everything updated.** It remains the first defence, by a wide margin.

If you suspect something is already running in your space that is not yours, **how to tell if your site has been compromised** has the signs; and the **PHP error log** usually holds the first clue.

If you need more than the jail allows

There are cases where shared hosting stops being enough: installing a service of your own, a custom-compiled version, or something that listens on a port. Then the route is a server of your own, with the freedom and the responsibility that brings - see **keeping your VPS secure** and **what you can host here**. And if all you need is an outbound port, that is solved without changing plan: **which ports are open**.

NEED A HAND?

Your developer needs to run a command and is not sure?

Ask us →

interweb

Interweb Angola

RLX Angola, Lda. · Group RLX Holdings

TAX ID (NIF)	500 260 5724
COMMERCIAL REGISTRATION NO.	26464-25/250724
REGISTERED OFFICE	Rua Direita do Patriota, Bairro Dangereux Belas, Luanda, Angola
CONTACT	info@interweb.ao · +244 942 011 191 · www.interweb.ao
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

RLX-AO-KB-342 · v2026.09.28 · Generated on 15/11/2026 00:08

Online version of this document