

Introduction

Captcha providers for the forms of derafu/form: hCaptcha, Turnstile, reCAPTCHA v3 and ALTCHA

Anonymous · 08/10/2026

Derafu: Captcha

[github](#) [derafu/captcha](#)[last commit](#) [today](#) CI [passing](#)[downloads](#) [48](#)

The captcha providers that [derafu/form](#) asks for: the widget that a form renders and the check of what the visitor solved. The application chooses one with three environment variables, **once**, and every form that is protected with the captcha (`options.captcha_protection: true`) uses it.

The providers

CAPTCHA_PROVIDER	Service	What the visitor does	Keys
hcaptcha	hCaptcha	Solves a widget.	site key and secret key
turnstile	Cloudflare Turnstile	Almost never anything: it is solved without puzzles most of the time.	site key and secret key
recaptcha-v3	Google reCAPTCHA v3	Nothing: it gives a score.	site key and secret key
altcha	ALTCHA	The browser solves a proof of work. No third party, no account, free (MIT).	only the secret key, and the package <code>altcha-org/altcha</code>
none	-	Nothing: the application decides, on purpose, not to have a captcha.	-

Install

```
composer require derafu/captcha
```

The package of ALTCHA is **not** required, because the other providers do not use it: install it only if you choose `altcha` (`composer require altcha-org/altcha`). Choosing it without the package is an error of the configuration that says so.

[derafu/foundation](#) imports its services. Otherwise, import them (they need a PSR-18 client and the

PSR-17 factories, and the parameter `app.locale`, the language of the widgets):

```
imports:
- { resource: '../vendor/derafu/captcha/resources/config/captcha-services.yaml' }
```

Configuration

Variable	Description
<code>CAPTCHA_PROVIDER</code>	<code>hcaptcha</code> , <code>turnstile</code> , <code>recaptcha-v3</code> , <code>altcha</code> or <code>none</code> . Empty: the application did not configure a captcha , and the forms that are protected with it fail with a message that says what to configure. <code>none</code> says, on purpose, that the application has none: those forms have none and there is no error.
<code>CAPTCHA_SITE_KEY</code>	The public key, that goes in the widget (not used by <code>altcha</code>).
<code>CAPTCHA_SECRET_KEY</code>	The secret key (for <code>altcha</code> , the key that signs its challenges: any long random text).
<code>CAPTCHA_MIN_SCORE</code>	Only for <code>recaptcha-v3</code> : the lowest score that is accepted, from <code>0.0</code> (a bot) to <code>1.0</code> (a person). <code>0.5</code> by default, and <code>0.7</code> with derafu/foundation , which is stricter.

A provider that is not known, that lacks its keys, or whose package is not installed is an **error of the configuration**, with a message that says what to fix; it is not a captcha that fails silently later. The service is **lazy**: the provider is made when a form first uses it (when the page of the form is rendered, or its data is processed), not when the application builds the renderer of the forms. So the error is in the page of the form, as an error page in HTML, and the rest of the site, error page included, keeps working. An application that wants to find out earlier asks for the service when it boots, or has a test that renders the page of the form (the smoke test of [derafu/foundation](#) does it with `/contact`).

The variables of each provider

Set `CAPTCHA_PROVIDER` and **only** the variables of that provider. A variable it needs and that is empty is an error of the configuration that names it.

<code>CAPTCHA_PROVIDER</code>	<code>CAPTCHA_SITE_KEY</code>	<code>CAPTCHA_SECRET_KEY</code>	<code>CAPTCHA_MIN_SCORE</code>	Extra
<code>hcaptcha</code>	Required	Required	-	-
<code>turnstile</code>	Required	Required	-	-
<code>recaptcha-v3</code>	Required	Required	Optional	-
<code>altcha</code>	Not used	Required	-	<code>composer require altcha-org/altcha</code>
<code>none</code>	-	-	-	-

Where the keys come from:

- **hCaptcha**: the dashboard of [hCaptcha](#): the *site key* is the one of the site, and the *secret* is in your account settings.
- **Turnstile**: in the dashboard of Cloudflare, *Turnstile*, add a widget for the domain: it gives a *site key* and a *secret key*.
- **reCAPTCHA v3**: in the [admin console of reCAPTCHA](#), create a site of type v3 for the domain: it gives a *site key* and a *secret key*. Keys of another version do not work.
- **ALTCHA**: there is no account and nothing to ask: the secret key is any long random text that you make, see below.

```
# hCaptcha
CAPTCHA_PROVIDER=hcaptcha
CAPTCHA_SITE_KEY=the-site-key
CAPTCHA_SECRET_KEY=the-secret-key

# Cloudflare Turnstile
CAPTCHA_PROVIDER=turnstile
CAPTCHA_SITE_KEY=the-site-key
CAPTCHA_SECRET_KEY=the-secret-key

# Google reCAPTCHA v3 (0.0 to 1.0, 0.5 by default)
CAPTCHA_PROVIDER=recaptcha-v3
CAPTCHA_SITE_KEY=the-site-key
CAPTCHA_SECRET_KEY=the-secret-key
CAPTCHA_MIN_SCORE=0.7

# ALTCHA
CAPTCHA_PROVIDER=altcha
CAPTCHA_SECRET_KEY=a-long-random-text

# No captcha, on purpose
CAPTCHA_PROVIDER=none
```

The simplest captcha: ALTCHA

If you do not want to open an account in a service, ALTCHA is enough to start: it is free (MIT), it connects to nothing, and it needs only a secret key, which is any long random text.

```
composer require altcha-org/altcha
```

Make the secret key with a random generator, for example 32 random bytes in hexadecimal (64 characters):

```
openssl rand -hex 32
```

```
CAPTCHA_PROVIDER=altcha
CAPTCHA_SECRET_KEY=the-output-of-the-command
```

The secret key **must be secret**: do not put it in the repository (it goes in the `.env` of the server, or in its secrets), and use a different one in each environment. If you change it, the challenges that were already in the pages stop being valid, which is only a form that has to be loaded again. The key signs the challenges, and with a key that is known anyone can make challenges that cost nothing. It is a weaker defense than the services that look at the behavior of the visitor (reCAPTCHA, Turnstile): it stops spam in bulk, not an attacker who is set on it.

How it checks

- **It fails closed.** If the service can not be asked (no answer, a status that is not 200, an answer that is not a JSON) the provider throws `CaptchaUnavailableException`: the form is not valid, with the message “The captcha could not be verified. Try again in a moment.”, and nobody gets in because the service is down.
- **The client is the PSR-18 one of the application**, so its timeout is the one of the client (derafu/foundation registers Guzzle with 10 seconds).
- **The action:** reCAPTCHA v3 and Turnstile are asked with the id of the form as their action, and an answer for another action is not valid. For reCAPTCHA v3 the action only has letters, numbers, `/` and `_`: anything else of the id becomes `_`.
- **ALTCHA** makes its challenge when the widget is rendered, signed with the secret key, with the id of the form and an expiry (10 minutes), and it verifies the solution by itself: it never connects to anything. A solution is not used up, it can be sent again until the challenge expires; the CSRF token of the form is what protects against a form that someone else sends.

The widgets

- **hCaptcha** and **Turnstile** render the `<div>` and the script of the service, in the language of the application.
- **reCAPTCHA v3** renders a hidden field and two scripts, one of them inline: when the form is sent, it asks Google for a token with the action of the form (the token lasts about two minutes, so it is not asked when the page is loaded) and sends the form again. A page with a content security policy that forbids inline scripts has to allow this one.
- **ALTCHA** renders the `<altcha-widget>` with its challenge inside, and loads its script from a CDN (`cdn.jsdelivr.net`): the major **3** of the npm package `altcha`, the build with the languages, which is the widget of `altcha-org/altcha 2`. The major 2 of the widget is of another protocol and does not read these challenges (it only says “Verification failed”), so an application that serves the file itself, with the `script` argument of `AltchaProvider`, must serve the version 3.

Tests

The tests of the package ask the **real services** with the keys for tests that they publish: hCaptcha (the secret and the token of the tests always pass), Turnstile (one secret always passes, another always fails) and reCAPTCHA (a secret that is not valid; the one that Google publishes for tests says success without a score, so it is not valid for v3). What a real service does not give (what is sent, a status that is not 200, an answer that is not a JSON, a score and an action) is tested with a local server of PHP that speaks the same protocol; a service that does not answer, with a closed port. ALTCHA is solved as a browser does it, with the library. The providers are also tested with the real renderer and processor of `derafu/form`. They need an internet connection.

Last updated on 08/10/2026