

How to Bypass Cloudflare With Selenium (2025 Guide)



Rubén del Campo
November 20, 2024 · 5 min read

Table of contents

Does your web scraper get blocked by Cloudflare? One of the best ways to solve this problem is by using a headless browser like Selenium. Unfortunately, plain Selenium often gets blocked by Cloudflare's anti-bot systems.

However, there are effective methods to bypass Cloudflare with Selenium. This guide will cover the four best techniques:

- [Method #1: Undetected ChromeDriver.](#)
- [Method #2: SeleniumBase.](#)
- [Method #3: Selenium Stealth.](#)
- [Method #4: Web Scraping API to Bypass Cloudflare Every Time.](#)

Let's go!

How Does Cloudflare Detect Selenium?

[Cloudflare](#) is a content delivery network (CDN) and cybersecurity provider. It blocks malicious HTTP traffic from moving to the server and performs security checks to mitigate Layer 7 (application layer) DDoS attacks.

Unfortunately, Cloudflare's security system doesn't spare web scrapers. It can detect and block automated browsers like Selenium WebDriver, recognizing them as automated bots.

Some key methods Cloudflare uses to detect headless browsers like Selenium include:

IP Reputation

Cloudflare maintains a database of IP addresses with known behavior patterns, including suspected bots. It considers historical activities, abnormal request frequency, and other behavioral data to determine an IP's reputation. When Selenium makes requests from an IP address with a poor reputation or abnormal behavior, it is more likely to be flagged and blocked.

HTTP Request Header Analysis

Cloudflare also uses [HTTP headers](#) to differentiate between a human user and an automated request. Headers such as User-Agent, Accept-Language, Platform, and others provide clues that can expose automation. By default, Selenium's headers are incomplete or missing relevant parameters, revealing automation signatures that Cloudflare can easily detect.

TLS Fingerprinting

from a real user or an automated tool. The details sent by Selenium during its handshake may not match those of genuine browsers, making them easier to identify and block.

CAPTCHAs

Cloudflare often uses [Turnstile CAPTCHAs](#) to verify whether a visitor is human. These CAPTCHAs are a barrier to web scraping. Selenium scripts typically struggle to interact with these CAPTCHAs because they require complex visual recognition beyond the standard capabilities of the automation frameworks.

Canvas Fingerprinting

[Canvas fingerprinting](#) technique uses the browser to draw hidden images on the canvas to generate a unique fingerprint. Cloudflare uses this method to differentiate between real users and bots by analyzing subtle variations in how the browser draws the canvas. Selenium often lacks the precise graphical output of a genuine user's browser, which can be a dead giveaway to anti-bot systems.

Read our detailed general guide to [bypassing Cloudflare](#) to learn more.

Frustrated that your web scrapers are blocked once and again?

ZenRows API handles rotating proxies and headless browsers for you.

[Try for FREE](#)

. . .

How to Bypass Cloudflare With Selenium

As mentioned, vanilla Selenium can't bypass Cloudflare since it can't access sites with complex anti-bot services. Let's look at four proven methods for bypassing Cloudflare detection with Selenium.

Method #1: Undetected ChromeDriver

[Undetected ChromeDriver](#) is a modified version of Selenium's Chrome WebDriver, featuring patches to access protected sites without triggering anti-bot measures. **These patches increase your chances of [bypassing bot detection systems](#) like Cloudflare by making Selenium appear more human-like.**

To get started with Undetected ChromeDriver in Python, install it using `pip`:

Terminal



```
pip3 install undetected-chromedriver
```

Let's use this [Cloudflare-protected challenge site](#) as a demo website to show how to use the Undetected ChromeDriver.

Example

```
# pip3 install undetected-chromedriver
import undetected_chromedriver as uc
from time import sleep

if __name__ == "__main__":

    # instantiate a Chrome browser
    driver = uc.Chrome(
        use_subprocess=False,
        headless=True,
    )

    # visit the target page
    driver.get("https://www.scrapingcourse.com/cloudflare-challenge")

    # wait for the interstitial page to load
    sleep(10)

    # take a screenshot of the current page and save it
    driver.save_screenshot("cloudflare-challenge.png")

    # close the browser
    driver.quit()
```

After a few trials, the above code grabs the site's home page, demonstrating that Undetected ChromeDriver bypassed Cloudflare:

Scraping Course

Cloudflare Challenge

You bypassed the Cloudflare challenge! :D

Click to open the image in full screen

Although Undetected ChromeDriver bypassed the detection, it doesn't work at scale and may even fail if you rerun the code, as the library still leaks some bot-like attributes. It also struggles with sites that have more advanced Cloudflare security, making it unreliable for many modern websites.

You can set up a [proxy with Undetected ChromeDriver](#) or change your User-Agent to increase the chances of bypassing Cloudflare. However, these techniques don't guarantee continuous success, as Cloudflare consistently strengthens its security measures to detect such tweaks.

No worries, this is just the first method. There are more techniques you can apply to bypass Cloudflare.

SeleniumBase is a web scraping and crawling tool in Python that lets you run Selenium in stealth mode using the Undetected ChromeDriver (UC). SeleniumBase is more efficient than the main Undetected ChromeDriver library because it **uses advanced browser patches to bypass anti-bot checks**.

Let's see how SeleniumBase performs against the same [Cloudflare challenge](#) site. First, install the library using `pip`:

Terminal

```
pip3 install seleniumbase
```

Start the browser instance in the GUI mode (non-headless) using UC. Then, navigate to the target page with a 6-second reconnection time to allow SeleniumBase to [bypass the initial JavaScript challenge](#).

Example

```
# pip3 install seleniumbase
from seleniumbase import Driver

# initialize the driver in GUI mode with UC enabled
driver = Driver(uc=True, headless=False)

# set the target URL
url = "https://www.scrapingcourse.com/cloudflare-challenge"

# open URL with a 6-second reconnect time to bypass the initial JS challenge
driver.uc_open_with_reconnect(url, reconnect_time=6)
```

Use the `uc_gui_click_captcha` method to click the CAPTCHA checkbox and resolve it if it appears. Note that **the `uc_gui_click_captcha` method only works in the GUI mode (non-headless)**. Finally, grab a screenshot of the web page:

Example

```
# ...

# attempt to click the CAPTCHA checkbox if present
driver.uc_gui_click_captcha()

# take a screenshot of the current page and save it
driver.save_screenshot("cloudflare-challenge.png")

# close the browser and end the session
driver.quit()
```

Combine the snippets, and you'll get the following complete code:

Example

```
# pip3 install seleniumbase
from seleniumbase import Driver

# initialize the driver in GUI mode with UC enabled
driver = Driver(uc=True, headless=False)

# set the target URL
url = "https://www.scrapingcourse.com/cloudflare-challenge"
```

```
driver.uc_gui_click_captcha()

# take a screenshot of the current page and save it
driver.save_screenshot("cloudflare-challenge.png")

# close the browser and end the session
driver.quit()
```

You should see the following screenshot upon successful execution:

Scraping Course

Cloudflare Challenge

You bypassed the Cloudflare challenge! :D

Click to open the image in full screen

SeleniumBase bypassed the anti-bot challenge!

However, it's essential to note that SeleniumBase has its limitations. While it may record a higher success rate than the main Undetected ChromeDriver library, it also doesn't guarantee success at scale.

You can also improve stealth by setting up a [proxy with SeleniumBase](#), but it can still get detected, particularly in headless mode. Additionally, since the library is open-source, Cloudflare's developers can study its mechanisms and adapt their defenses to block it.

Method #3: Selenium Stealth Plugin

The [Selenium Stealth](#) plugin is a helper that modifies Selenium with real browser fingerprints and evasion techniques, e.g., setting the WebDriver navigator property to false, replacing the HeadlessChrome User Agent in headless mode with an actual Chrome User Agent, and more.

To start with Selenium Stealth, install it using `pip :e-bypass`).

Terminal



```
pip install selenium-stealth
```

Import the library and spin the browser in headless mode. Add the driver to the stealth mode and visit the target site. Here's a basic code to use Selenium Stealth:

Example



```
# Step 2: Change browser properties
# create a ChromeOptions object
options = webdriver.ChromeOptions()

# run in headless mode
options.add_argument("--headless")

# disable the AutomationControlled feature of Blink rendering engine
options.add_argument("--disable-blink-features=AutomationControlled")

# create a driver instance
driver = webdriver.Chrome(options=options)

# enable stealth mode and set extra parameters
stealth(driver)

# navigate to the target site
driver.get("https://www.scrapingcourse.com/cloudflare-challenge")

# wait for the interstitial page to load
sleep(10)

# take a screenshot of the current page and save it
driver.save_screenshot("cloudflare-challenge-st.png")

# close browser
driver.quit()
```

Unfortunately, there's no guarantee that Selenium Stealth will work. It only partially patches Selenium and leaks several bot-like attributes, reducing its chances of bypassing advanced detection techniques of more complex Cloudflare security measures. Additionally, it doesn't handle IP bans or geo-restrictions out of the box.

Are you curious about the method that works all the time? Jump to the next section.

Method #4: Web Scraping API to Bypass Cloudflare Every Time

For guaranteed success in bypassing Cloudflare, one tried-and-true method is to use a web scraping API like [ZenRows](#).

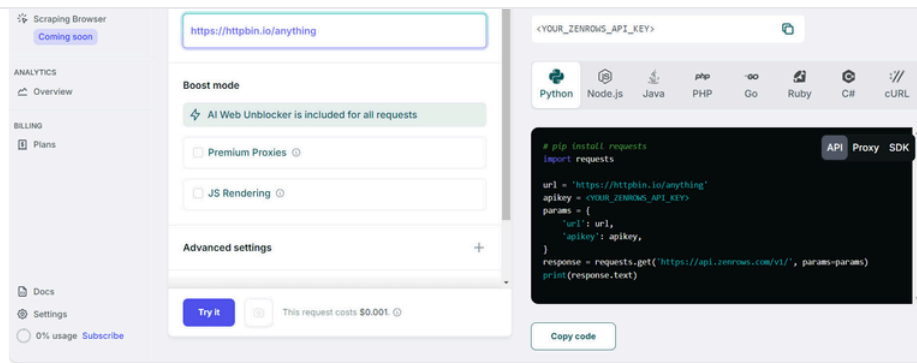
ZenRows provides all the tools you need to scrape any protected website at scale without limitations. It features auto-rotating premium proxies, auto-parsing, CAPTCHA and anti-bot auto-bypass, and more.

With ZenRows, you only need to make a single API call with your chosen programming language and watch it do the bypassing job behind the scenes. ZenRows also acts as a headless browser to replace Selenium, featuring various JavaScript instructions for interacting with web pages like humans.

To see how ZenRows works, let's use it to access and scrape the previous Cloudflare challenge page.

[Sign up](#) to open the ZenRows Request Builder. Paste the target URL in the link box and activate Premium Proxies and JS Rendering.

Select Python as your programming language and choose the API connection mode. Copy and paste the generated code into your Python script.



Click to open the image in full screen

The generated code should look like this:

```
Example

# pip install requests
import requests

url = 'https://www.scrapingcourse.com/cloudflare-challenge'
apikey = '<YOUR_ZENROWS_API_KEY>'

params = {
    'url': url,
    'apikey': apikey,
    'js_render': 'true',
    'premium_proxy': 'true',
}

response = requests.get('https://api.zenrows.com/v1/', params=params)
print(response.text)
```

The above scraper accesses the protected website and scrapes its full-page HTML, as shown:

```
Output

<html lang="en">
<head>
  <!-- ... -->
  <title>Cloudflare Challenge - ScrapingCourse.com</title>
  <!-- ... -->
</head>
<body>
  <!-- ... -->
  <h2>
    You bypassed the Cloudflare challenge! :D
  </h2>
  <!-- other content omitted for brevity -->
</body>
</html>
```

Congratulations! Your scraper now bypasses the highest Cloudflare protection with ZenRows.

In this Selenium Cloudflare bypass guide, you've learned four ways to improve vanilla Selenium and avoid Cloudflare detection. Using open-source solutions such as Undetected ChromeDriver, SeleniumBase, and Selenium Stealth, as well as even setting them up with proxies, offer varying levels of success by targeting specific aspects of Selenium's operation.

However, the only method that always works is employing a web scraping API, such as ZenRows. This solution handles all anti-bot bypass technicalities behind the scenes while you focus on your scraping logic.

Try [ZenRows for free](#) now without a credit card!

Share



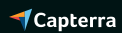
Ready to get started?

Up to 1,000 URLs for free are waiting for you

[Try ZenRows for Free](#)



ZenRows® web scraping API handles all anti-bot bypass for you with rotating proxies, headless browsers and more.



Company

[Web Scraping Blog](#)

[Pricing](#)

[API Documentation](#)

[Legal](#)

[Affiliate](#)

[Contact](#)

[Press](#)

[Customers](#)

[Careers](#)

Ready-to-use Scrapers

[Youtube Scraper](#)

[Zillow Scraper](#)


[Indeed Scraper](#)

[All Popular Web Scrapers](#)

[Web Scraping in NodeJS](#)[Web Scraping in Java](#)[Web Scraping in PHP](#)[Web Scraping in R](#)[Web Scraping in Ruby](#)[Web Scraping in Golang](#)[Web Scraping in C#](#)[Web Scraping in Rust](#)[Web Scraping in C++](#)[Web Scraping in C](#)[Perl Web Scraping](#)[Scrapy Python Web Scraping](#)[cURL Converter](#)[Selenium Web Scraping](#)[Playwright Web Scraping](#)[Puppeteer Web Scraping](#)[Bypass Akamai](#)[Bypass PerimeterX](#)[Bypass DataDome](#)[Web Scraping Without Getting Blocked](#)[Avoid Getting Blocked in Python](#)[Solve CAPTCHAs](#)[Web Scraping Proxy](#)

How we compare

[ScrapingBee Alternative](#)[BrightData Alternative](#)[ZenRows Alternative](#)

 All services are online

2025 ZenRows®. All rights reserved.

[X](#) [in](#)