Passing AntiBot

Getting Past Cloudflare's Anti-Bot / Challenge Page — The Full Toolbox

Cloudflare layers **many** signals on top of each other. No single trick works universally; you often need to combine several. Below is the full stack, from easiest to most aggressive.

---

1. The Signal Stack Cloudflare Actually Inspects

Before diving into tricks, know what's being scored:

Layer	What it checks
IP reputation	Datacenter vs residential vs mobile, ASN, geo, prior abuse
TLS fingerprint (JA3 / JA4)	Cipher order, extensions, curve list, ALPN — unique per library/browser
HTTP/2 fingerprint	SETTINGS frame values, pseudo-header order, WINDOW_UPDATE size
HTTP headers	Which headers, their *order*, values (UA, Accept-Language, etc.)
JavaScript execution	The actual challenge: a PoW hash, DOM checks, or Turnstile
Browser fingerprint	Canvas, WebGL, AudioContext, fonts, screen size, plugins, `navigator.*`
Behavioural signals	Mouse movement, scroll, timing of clicks, focus events

Layer	What it checks
Cookie state	`cf_clearance`, `__cf_bm`, `__cf_chl_*` — tied to IP + UA + TLS
Rate / volume	How fast you're hitting, from how many IPs

A request that looks fine at layer 3 but has a Python-`requests` TLS fingerprint at layer 1 will get flagged.

---

2. Quick / Low-Effort Fixes

- **Just wait 5–10 s.** The JS challenge is often a proof-of-work that resolves automatically in a real browser.
- **Use a real, up-to-date browser** (Chrome, Firefox, Edge, Safari). Not a headless container, not an old one.
- **Disable ad-blockers / privacy extensions** (uBlock, Privacy Badger, etc.) that strip or reorder headers.
- **Enable JS, cookies, and third-party cookies** for the site.
- **Make sure your system clock is correct** (cert validation depends on it).
- **Try a different browser or an incognito window.**
- **Don't hammer the page.** Reload at most once or twice, then wait.

---

3. IP / Network Layer

This is often the #1 reason you keep getting challenged.

- **Avoid datacenter / cloud IPs** (AWS, GCP, DigitalOcean). Cloudflare flags them aggressively.
- **Use a residential proxy** (IPRoyal, Smartproxy, SOAX, Decodo, etc.) or a **mobile proxy**. They rotate IPs that look like real home/mobile users.
- **Avoid widely-used free VPNs** (Proton, Windscribe free tier, etc.) — Cloudflare has their ASN on a blocklist.
- **Geo-consistency:** your proxy IP's country should match your `Accept-Language`, timezone, and the ASN. A US IP sending `Accept-Language: de-DE` looks odd.
- **Stick to one IP per session.** Rapidly rotating IPs for the same "user" triggers the rate/volume check.

---

4. TLS & HTTP/2 Fingerprint (the "invisible" wall)

Even with a residential IP, `python-requests`, `curl`, or `Node fetch` have a **distinct TLS handshake** that a real browser doesn't. Cloudflare fingerprints this.

Tools that impersonate a real browser's TLS + HTTP/2 fingerprint:

Tool	Language	Notes
`curl_cffi`	Python	`pip install curl_cffi` – impersonates Chrome / Safari / Edge / Firefox TLS. Easiest
to start.		
`curl-impersonate`	C (CLI)	Patched `curl` binary that clones Chrome/Safari handshake byte-for-byte.
`tls-client` (bogdanfinn)	Go	Very popular in the scraping community. Supports cookie jars, proxy
rotation.		
`cycletls`	Go	Fork of tls-client, actively maintained.
`utls`	Go	Low-level; you build the handshake yourself.
`httpx` + `curl_cffi` backend	Python	Modern async HTTP with the right fingerprint.

Example (Python, curl_cffi):

```
from curl_cffi import requests

session = requests.Session(impersonate="chrome124")
session.proxies = {"https": "http://user:pass@residential-proxy:port"}

r = session.get("https://protected-site.com", headers={
    "Accept-Language": "en-US,en;q=0.9",
})
```

```
print(r.status_code, r.text[:200])
```

This alone clears a surprising number of 403 / challenge pages that vanilla `requests` can't.

---

5. The `cf_clearance` Cookie (the "master key")

When a real browser successfully passes the challenge, Cloudflare sets a **`cf_clearance`** cookie. It is:

- Tied to your **IP + User-Agent + TLS fingerprint**.
- Valid for roughly **30 min – a few hours**.
- Reusable for subsequent requests *without* re-solving the challenge.

Strategy: **solve once, reuse many**

1. Open the site in a **real Chrome** on a machine with a **residential IP**.
2. Let the challenge auto-resolve (or click Turnstile).
3. Extract the cookies:
   - DevTools → Application → Cookies, **or**
   - A browser extension (e.g., *Get cookies.txt*, *EditThisCookie*).
4. Replay those cookies in your script **with the same UA, same IP, same TLS fingerprint**.

```
# Reuse the clearance cookie
cookies = {
    "cf_clearance": "abcdef...",
    "__cf_bm": "123456...",
}
r = session.get("https://protected-site.com/data", cookies=cookies)
```

> ⚠ If the IP, UA, or TLS fingerprint changes even slightly, the cookie is rejected.

Automate the "solve once" step:

- **Flare Solver** – a small Chrome extension + REST API. You POST the URL, it opens the page in a real Chrome, waits for the challenge, and returns the cookies.
- **Playwright / Puppeteer** with a **real (non-headless) Chrome** + stealth plugin (see next section).
- Run a **real Chrome on a VPS** with an Xvfb display server and a residential proxy, drive it with Selenium/Playwright.

---

6. Headless Browser + Stealth (when you must script)

If you need a full browser (for dynamic content, SPAs, etc.):

Key stealth plugins / configs

- **Playwright**:

```
from playwright.sync_api import sync_playwright
from playwright_stealth import stealth_sync

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, args=[
        "--disable-blink-features=AutomationControlled",
        "--no-sandbox",
    ])
    page = browser.new_page(user_agent="...")
    stealth_sync(page)          # patches navigator.webdriver, etc.
    page.goto("https://protected-site.com")
    page.wait_for_timeout(8000) # give the challenge time
    # page.wait_for_selector("#content", timeout=15000)
```

- **Puppeteer (Node)**: use `puppeteer-extra` + `puppeteer-extra-plugin-stealth`.
- **Selenium**: use **`undetected-chromedriver`** (Python) which strips the `--enable-automation` flag and patches `navigator.webdriver`.

Things the stealth plugins patch:

- `navigator.webdriver = false`

- Removes `chrome.runtime` / `cdc_` artifacts
- Fixes `window.length` vs `window.frames.length` mismatch
- Patches `Intl.DateTimeFormat` timezone
- Removes automation-related CDP listeners

Extra hardening:

- **Don't run `headless=True`** if you can avoid it. Use `xvfb-run` or `--headless=new` (Chromium's new headless mode is much less detectable).
- Set a **real viewport**, **real screen size**, real **timezones**.
- Inject a small **userscript** (via Tampermonkey or `page.add_init_script`) to randomise Canvas / WebGL / AudioContext outputs.
- Add **human-like interaction**: random mouse moves, scroll a little, wait 2–4 s before clicking.
- **Don't load extensions** that leak (or use a consistent set every time).

---

7. Cloudflare Turnstile (the "new" captcha)

Turnstile replaced the old I-AM-HUMAN checkbox in many deployments. It's a small widget that:

- Runs a JS proof-of-work.
- Checks your fingerprint.
- In "managed" mode, may show no UI at all (invisible) or a checkbox.

How to handle it:

- In a **real browser**, it usually just resolves silently. Wait.
- In **Playwright/Puppeteer**, you can:
  - Wait for the iframe `challenges.cloudflare.com` to appear.
  - Click the checkbox if visible: `page.frame_locator("iframe[src*=challenges.cloudflare.com]").locator("input[type=checkbox]").click()`.
  - Wait for the `cf-turnstile-response` to be set.
- If it keeps failing, your **fingerprint or IP is being flagged** → go back to layers 3–4.

---

8. Alternative Endpoints (cheeky but effective)

Sometimes the main site is behind a heavy challenge, but:

- `api.example.com` – JSON API, lighter protection.
- `amp.example.com` – AMP pages.
- `example.com/feed` or `rss.xml` – XML feed.
- `sitemap.xml` – often served with less bot protection.
- The **Cloudflare Worker** that powers the site (if it's a SPA, the data comes from an API behind the Worker, which may have different rules).
- **CDN-cache paths**: if the page is cached at the edge, a conditional `If-None-Match` / `If-Modified-Since` request might get a 304 before the challenge.

Check the site's `robots.txt` and look for API docs.

---

9. If You Own / Control the Site

You (or the site owner) can simply **turn the dial down**:

- **Cloudflare Dashboard → Security → Bots → Bot Fight Mode**: toggle off, or lower the sensitivity.
- **Security → WAF**: add a rule to **skip the challenge** for your IP / IP range.
- **Security → Settings → Browser Integrity Check**: toggle off (lets you skip the JS check).
- **Use Cloudflare Access / Access Policies** with an auth token or JWT instead of the public challenge.
- **Create an API Token** and call the origin directly (bypassing the CDN) or through the Cloudflare API.
- **Add your bot to the "Verified Bot" list** (Cloudflare → Security → Bots → Verified Bots).
- **Exclude specific paths** (e.g., `/api/*`) from the managed challenge via a WAF rule or a Page Rule.

---

10. Nuclear / "For All Costs" Options

If nothing above works:

1. **Run a real Chrome on a residential-IP VPS** (e.g., a $5/month VPS in a datacenter + a residential proxy on the outbound), drive it with Playwright + stealth. This is what most scraping farms do.
2. **Use a managed scraping service** (Bright Data, Oxylabs, ScrapingBee, Apify, Zyte). They've already solved the fingerprint + IP + challenge dance and sell you the clean HTML/JSON.
3. **`curl-impersonate` + a real `cf_clearance` cookie**: the closest you can get to "invisible" without a full browser.
4. **Solve the PoW manually** (advanced): Cloudflare's old managed challenge is a hash-based proof-of-work. You can:
   - Intercept the JS with a breakpoint.
   - Extract the challenge parameters (`cf_chl_opt`).
   - Run the hash loop in your own code.
   - POST the result back and collect the `cf_clearance` cookie.
   - (This is what libraries like **`cloudscraper`** tried to do, but Cloudflare keeps rotating the algorithm, so it breaks periodically.)
5. **Use Cloudflare's own edge**: if you own the zone, add a **Cloudflare Worker** that sets a custom header or rewrites the response before it hits the challenge logic.

---

11. A Practical Decision Flowchart

```
GET request → 200?  → Done □
                → 403 / challenge page?
                    ├─ Are you on a datacenter IP?  → Switch to residential
                    ├─ Are you using raw requests/curl? → Switch to curl_cffi / tls-client
                    ├─ Is your UA / headers consistent with IP geo?  → Fix
                    ├─ Can you solve it once in a real browser and reuse the cookie? → Do that
                    ├─ Do you need dynamic content / a SPA?  → Playwright + stealth + residential proxy
                    ├─ Is it Turnstile?  → Real browser, wait, click checkbox if shown
                    └─ Still stuck?  → Managed scraping API, or ask the site owner to whitelist you
```

TL;DR: For most people, the combo that "just works" is:

> **Residential proxy + `curl_cffi` (or `tls-client`) + a real browser's UA + solve the challenge once in Chrome to grab `cf_clearance` + reuse the cookie.**

That covers ~90 % of cases. The remaining 10 % (heavy behavioural checks, Turnstile with visible UI, rate-limited APIs) need a full stealth-browser session or a paid scraping service.

Want me to write a concrete script for a specific target or toolchain?

Revision #1
Created 11 September 2026 15:07:49 by Togoboi
Updated 11 September 2026 15:10:57 by Togoboi