oxylabs®

Block-Free Web Scraping: A Comprehensive Guide

Introduction

Web scraping is an essential practice for data-driven businesses. Acquiring and analyzing large amounts of public data can unveil surprising insights, creating new avenues for overall business strategy and growth.

A common roadblock experienced by nearly everyone who has ever tried web scraping is getting blocked. For various reasons, websites aren't too keen on letting automated scripts browse their content freely. As a result, many web scraping newcomers get blacklisted and lose access to website content.

Even when collecting data with the best intentions, getting blocked is possible. Websites have a hard time differentiating between good and bad bots, so they often enact blanket bans that stop most scripts from working.

Over the years spent in the web data industry, we have collected many technical insights that aid in avoiding blocks. We are here to share these insights with you. Following the steps outlined here, you can scrape freely and get banned less frequently.

Table of contents

How to extract data without getting blocked **5**

Check the robots exclusion protocol 5

Use a proxy server 6

Rotate IP addresses 6

Use real user agents 6

Set your fingerprint right 7

Beware of honeypot traps 7

Use CAPTCHA solving services 8

Change the scraping pattern 8

Reduce the scraping speed 8

Scrape during off-peak hours 9

Avoid JavaScript 9

Use a headless browser 9

Setting the right approach to web scraping **10**

Sessions and their importance 10

HTTP headers for web scraping 11

Fingerprinting and its relevance 12

More practical tips on web scraping 12

5 key HTTP headers for web scraping **13**

User-Agent 13

Accept-Language 14

Accept-Encoding 15

Accept 15

Referer 16

How websites detect bad bots, and how it affects web scraping? 16

What is bot traffic? 17

Good bots 18

Bad bots 18

Bot detection challenges 19

How to detect bot traffic? 20

How anti-bot measures affect web scraping 21

Overcoming anti-bot measures with improved web scraping tools 21

Web Unblocker 22

Conclusion 25

How to extract data without getting blocked

Web crawling and web scraping are essential for public data gathering. Here is a list of actions to prevent getting blacklisted while scraping and crawling websites.

Web pages detect web crawlers and scraping tools by checking their IP addresses, user agents, browser parameters, and general behavior. If the website finds it suspicious, you receive CAPTCHAs, and eventually, your requests get blocked.

Check the robots exclusion protocol

Before crawling or scraping any website, make sure your target allows data gathering from their page. Inspect the robots exclusion protocol (robots.txt) file and respect the website's rules.

Even when the web page allows crawling, be respectful and don't harm the page. Follow the rules outlined in the robots exclusion protocol, crawl during off-peak hours, limit requests from one IP address, and set a delay between them.

However, even if the website allows web scraping, you may still get blocked, so it's important to follow other steps, too.

Use a proxy server

Web scraping would be hardly possible without proxies. Pick a reliable proxy service provider and choose between the datacenter and residential IP proxies, depending on your task.

Using an intermediary between your device and the target website reduces IP address blocks, ensures anonymity, and allows you to access websites that might be unavailable in your region. For example, if you're based in Germany, you may need to use a US proxy to access web content in the United States.

For the best results, choose a proxy provider with a large pool of IPs and a wide set of locations.

Rotate IP addresses

When using a proxy pool, it's essential to rotate your IP addresses.

If you send too many requests from the same IP address, the target website will soon identify you as a threat and block your IP address. Proxy rotation makes you look like a number of different internet users and reduces your chances of getting blocked.

All Oxylabs Residential Proxies are rotating IPs, but if you're using Datacenter Proxies, you should use our Proxy Rotator feature.

Use real user agents

Most servers that host websites can analyze the HTTP request headers that scraping bots make. The HTTP request header, called User-Agent, contains information about the operating system and applications.

Servers can easily detect suspicious user agents. Real user agents contain popular HTTP request configurations submitted by organic visitors. To avoid getting blocked, make sure to customize your User-Agent to look like an organic one.

Since every request a web browser makes contains a User-Agent, you should switch the User-Agent frequently.

It's also important to use up-to-date and the most popular user agents. If you're making requests with some years-old User-Agent from a Firefox version that is no longer supported, it raises a lot of red flags. You can find public databases on the internet that show you which user agents are the most popular.

Set your fingerprint right

Anti-scraping mechanisms are becoming more sophisticated, and some websites use Transmission Control Protocol (TCP) or IP fingerprinting to detect bots.

When scraping the web, TCP leaves various parameters. These parameters are set by the end user's operating system or the device. If you're wondering how to prevent getting blacklisted while scraping, make sure your parameters are consistent.

Beware of honeypot traps

Honeypots are links in the HTML code. These links are invisible to organic users, but web scrapers can detect them. Honeypots are used to identify and block scrapers because only robots would follow such a link.

Since setting honeypots requires a relatively large amount of work, this technique is not widely used. However, if your request is blocked, beware that your target might be using honeypot traps.

Use CAPTCHA solving services

CAPTCHAs are one of the biggest web scraping challenges. Websites ask visitors to solve various puzzles to confirm they're humans. The current CAPTCHAs often include images that are nearly impossible to read for computers.

To bypass CAPTCHAs, use dedicated CAPTCHA solving services or ready-to-use scraping tools.

Change the scraping pattern

The pattern refers to how your scraper is configured to navigate the website. If you constantly use the same basic pattern, it's only a matter of time before you get blocked.

You can add random clicks, scrolls, and mouse movements to make your scraping seem less predictable. However, the behavior should not be completely random. One of the best practices when developing a pattern is to think of how a regular user would browse the website and then apply those principles to the tool itself. For example, visiting the home page first and only then making some requests to the inner pages makes a lot of sense.

Reduce the scraping speed

To mitigate the risk of being blocked, you should slow down your scraper speed. For instance, you can add random breaks between requests or initiate wait commands before performing a specific action.

If a server has rate limiting, the target can only perform a limited number of actions on the website at a certain time. To avoid request throttling, respect the website and reduce your scraping speed.

Scrape during off-peak hours

Most scrapers move through pages significantly faster than an average user as they don't actually read the content. Thus, a single unrestrained web scraping tool will affect server load more than any regular internet user. In turn, scraping during high-load times might negatively impact user experience due to service slowdowns.

Finding the best time to scrape the website will vary on a case-by-case basis, but picking off-peak hours just after midnight (localized to the service) is a good starting point.

Avoid JavaScript

Data nested in JavaScript elements is hard to acquire. Websites use many different JavaScript features to display content based on specific user actions. A common practice is to only display product images in search bars after the user has provided some input.

JavaScript can also cause a host of other issues, such as memory leaks, application instability, or, at times, complete crashes. Dynamic features can often become a burden. Avoid JavaScript unless absolutely necessary.

Use a headless browser

One additional tool for block-free web scraping is a headless browser. It works like any other browser, except it doesn't have a graphical user interface (GUI).

A headless browser also allows for the scraping of content loaded by rendering JavaScript elements. The most widely used web browsers, Chrome and Firefox, have headless modes.

Set your browser parameters right, take care of fingerprinting, and beware of honeypot traps. Most importantly, use reliable proxies and scrape websites with respect. Then, all your public web data gathering jobs will go smoothly, and you can use fresh information to improve your business.

Setting the right approach to web scraping

Figuring out how to start web scraping can be a complicated task. To make it easier, follow this workflow:

1. **Set a preferred session**
2. **See if it works with a test query**
3. **Start scraping your target web page**

Sessions and their importance

Sessions are an essential part of the residential proxy network. They enable you to use the same IP address for multiple requests. By default, a new proxy carries out every new request that goes through the residential network, which can cause issues.

For example, if you use a full browser, bot, or headless browser to download assets from your target websites, all of them must be downloaded using the same IP address. In this case, assets mean everything that comes with the HTML – CSS, JavaScript files, images, etc.

Reliable proxy providers offer flexible and adjustable session control features, so you can be sure this part will be easily managed.

HTTP headers for web scraping

HyperText Transfer Protocol (HTTP) manages how communication is transferred and structured on the internet. It is also responsible for how web servers and browsers should respond to different requests. There are different types of HTTP headers: request header, response header, general HTTP header, entity header, and so on.

When web scraping, sending the HTTP headers, preferably in the right order, is the bare minimum. Requests without specific HTTP headers are likely to be blocked very quickly.

To start optimizing HTTP headers, we advise you to see how the browser works by itself. In Firefox or Chrome, hit the F12 button and open developer tools. Go to the Network tab and refresh the page you are on. You will see all the requests the browser made to render the page fully. Find where the HTML content was loaded, and you will see what headers were sent and in which order. Try to make this happen on your scraper, too.

Fingerprinting and its relevance

Fingerprinting is all the information that your browser gives websites about you and your computer, such as mouse input, resolution, installed plugins, and much more. Having all this information, you can make a single hash – a fingerprint. It makes it easier to identify whether requests come from a browser. Fingerprinting is becoming the primary weapon for identifying web scraping bots, increasing the chances of being blocked.

Some websites already have anti-scraping solutions that check fingerprints, but it is not universal yet. The major problem is that it still generates a lot of false positives, which might have converted to sales. More importantly, it requires tremendous hardware resources to process all the data. Overall, the chances of running into such issues are quite slim, but if you do, the best way is to use a headless browser, preferably with stealth add-ons.

More practical tips on web scraping

- Visit the home page before accessing the inner content. Regular users rarely have full links to products or articles. They first land on the home page and then browse further.

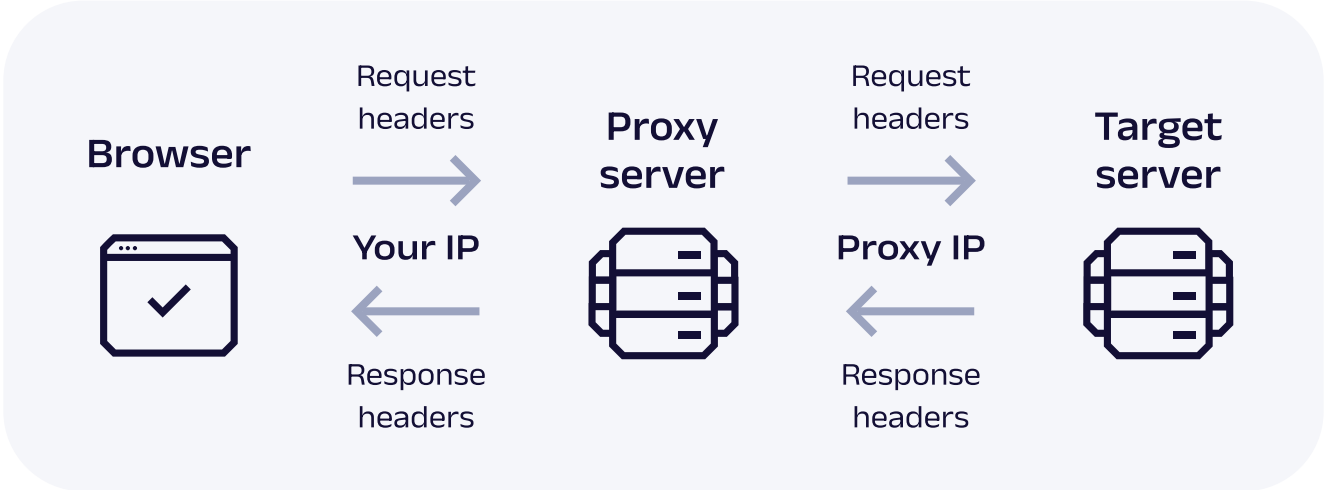
- Data under authentication or protected with a password could be considered private, and scraping such data can be illegal in some cases. Before starting web scraping, we suggest you consult legal professionals, carefully read the particular website's terms of service, or even receive a scraping license if possible.

- Choose the right proxy type for your web scraping tasks. Two of the main proxy types are residential and datacenter proxies. They are usually used for different targets.

5 key HTTP headers for web scraping

A rather overlooked technique is to use and optimize HTTP headers. The following are the essential HTTP headers that need to be used and optimized.

- User-Agent
- Accept-Language
- Accept-Encoding
- Accept
- Referer



User-Agent

The User-Agent request header passes information related to identifying application type, operating system, software, and version. It allows data targets to decide what type of HTML layout to use in response (mobile, tablet, or PC).

User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.1 Safari/605.1.15

Authenticating the User-Agent request header is a common practice by web servers, and it is the first check that allows data sources to identify suspicious requests. For instance, when web scraping is in process, numerous requests are traveling to the web server, and if User-Agent request headers are identical, it will seem as if it is a bot-like activity. Hence, experienced web scraping punters will manipulate and differentiate User-Agent header strings, which consequently allows for portraying multiple organic user sessions.

When it comes to the User-Agent request header, remember to frequently alter the information this header carries.

Accept-Language

The Accept-Language request header indicates to a web server which languages the client understands and which particular language is preferred when the web server sends the response.

Accept-Language	en-gb

It's worth mentioning that this particular header usually comes into play when web servers are unable to identify the preferred language, for example, via URL.

It is essential to ensure that set languages are in accordance with the data-target domain and the client's IP location. Requests from the same client appearing in multiple languages will raise suspicions to the web server of bot-like behavior (non-organic request approach).

Accept-Encoding

The Accept-Encoding request header notifies the web server what compression algorithm to use when handling the request. In other words, it states that the required information can be compressed (if the web server can handle it) when sent from the web server to the client.

Accept-Encoding	br, gzip, deflate

However, when optimized, it allows saving traffic volume, which is a win-win situation from the traffic load perspective for both the client and the web server. The client still gets the required information (just compressed), and the web server isn't wasting resources by transferring a huge traffic load.

Accept

The Accept request header falls into a content negotiation category. Its purpose is to notify the web server what data format can be returned to the client.

Accept	test/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

It's as simple as it sounds, but a common hiccup with web scraping is overlooking or forgetting to configure the request header according to the web server's accepted format. If the Accept request header is configured suitably, it will result in more organic communication between the client and the server.

Referer

The Referer request header provides the address of the previous web page before the request is sent to the web server.

Referer	http://www.google.com

It might seem that Referer does not have much impact when it comes to blocking the scraping process when, in fact, it actually does. Think of a random organic user's internet usage patterns. This user is likely surfing the internet and losing track of their time. Hence, to make the web scraper's traffic seem more organic, specify a random website before starting a web scraping session.

The key is not to rush. Instead, take this rather straightforward step. Hence, remember to always set up the Referer request header to boost your chances of slipping under web servers' anti-scraping measures.

It's safe to state that the more you know about the technical side of web scraping, the more fruitful your web scraping results will be. Use this knowledge wisely, and it's a given that your web scraper will work more effectively and efficiently.

How websites detect bad bots, and how it affects web scraping?

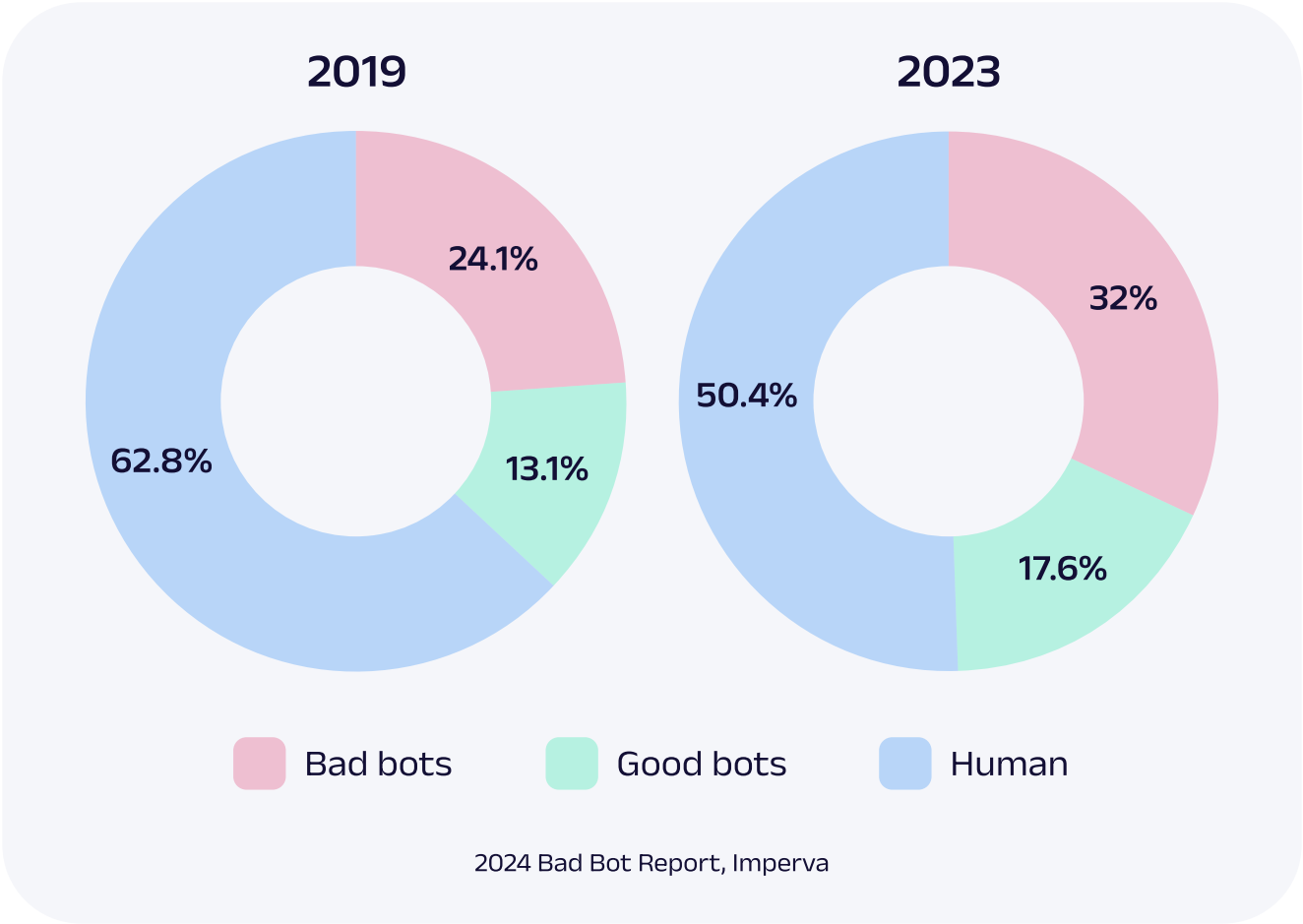
Often, we perceive the term bot as negative. However, not all bots are bad. The issue is that good bots can share similar characteristics with malicious bots. Therefore, good bots get labeled as bad and get blocked.

Bad bots are only getting smarter, and it's hard for good bots to stay block-free. This creates many issues not only for site owners who want to ensure the healthy performance of their websites but also for the web scraping community.

What is bot traffic?

Bot traffic is any non-human traffic made to a website. It's a software application running automated and repetitive tasks much faster than humanly possible. With the ability to perform tasks quickly, bots can be used for both bad and good.

Unsurprisingly, with the advent of generative AI, overall bot traffic is steadily increasing. With such an increase, website owners are forced to strengthen their security, allowing more good bots to get wrongfully caught.



2024 Bad Bot Report, Imperva

To better understand what are good and bad bots, here are some examples:

Good bots

Search engine bots crawl, catalog, and index web pages. The results are used by search engines such as Google to provide their services effectively.

Site monitoring bots monitor websites to identify possible issues such as long loading times and downtimes.

Web scraping bots collect publicly available data for research, identifying and pulling down illegal ads, monitoring brand mentions, and much more.

Bad bots

Spam bots create fake accounts on forums, social media platforms, and messaging apps. They are used to build a social media presence and create more clicks on a post.

DDoS attack bots take down websites. They usually leave just enough bandwidth available to allow other attacks to penetrate the network, pass weakened network security layers undetected, and steal sensitive information.

Ad fraud bots automatically click on ads, siphoning money from advertising transactions.

A good bot performs useful or helpful tasks that aren't detrimental to a user's experience on the internet. In contrast, a bad bot is the opposite and usually has malicious or illegal intentions.

Bad bot traffic is predicted to increase each year. As for good bot traffic, the chance to avoid getting mixed in with the bad crowd is slowly dwindling. Amongst good bots, there are many web scrapers that use gathered data for research, pulling down illegal ads, market research, etc. All of them may get flagged as bad and blocked.

Bot detection challenges

Distinguishing bots from human behavior online has become a complex task in itself, and the bots on the internet have evolved dramatically over the years. Currently, there are four different generations of bots:

First-generation bots are built with basic scripting tools and mainly perform basic automated tasks like scraping, spam, etc.

Second-generation bots mainly operate through website development, hence ending up with the name web crawlers. They are relatively easy to detect due to specific JavaScript firing and iframe tampering.

Third-generation bots are often used for slow DDoS attacks, identity thefts, API abuse, etc. They are relatively difficult to detect based on device and browser characteristics and require proper behavioral and interaction-based analysis to identify.

Fourth-generation bots perform human-like interactions, like nonlinear mouse movements. To detect such bots, advanced methods, often involving AI and machine learning, are required.

The fourth generation of bots is tough to differentiate from legitimate human users, and basic bot detection technologies are no longer sufficient.

How to detect bot traffic?

Websites have set up various bot detection techniques to prevent bad bots. Some of these are very likely to be encountered daily (CAPTCHAs).

Of course, there are more bot detection techniques to take into account. So, how to detect bots? Here are some ways to do that:

Browser fingerprinting – the main approach is to check for the presence of attributes added by headless browsers like PhantomJS, Nightmare, Puppeteer, Selenium, and others.

Browser consistency – checking the presence of specific features that should or should not be in a browser. This can be done by executing certain JavaScript requests.

Behavioral inconsistencies – nonlinear mouse movements, rapid button and mouse clicks, repetitive patterns, average page time, average requests per page, starting browsing from inner pages without collecting HTTP cookies, and similar bot-like behavior.

CAPTCHA – a challenge-response type test that often asks you to fill in correct codes or identify objects in pictures.

Once a website identifies bot-like behavior, it blocks them from further activity.

How anti-bot measures affect web scraping

We have covered this topic in our blog post on fingerprinting and its impact on web scraping. According to Allen O'Neill, a full-stack big data cloud engineer:

> The only way (to overcome anti-bot measures in web scraping) will be to build guided bots, i.e., construct personas that will need to have their web-footprint schedules. Just like regular internet users, they will have to show their organic behavior to visited websites. Only then will it be possible to mix within the internet crowd, and consequently, slip under implemented anti-bot detection means.

Overcoming anti-bot measures with improved web scraping tools

O'Neill's predictions were quite accurate. With the fourth-generation bots evolving into fifth, Oxylabs focused on developing a tool to overcome these challenges.

Web Unblocker

Web Unblocker is an AI-powered proxy solution offering a system of processes that support block-free web scraping. The system automates unblocking by managing proxies, browser fingerprinting, automatic retries, CAPTCHA bypass, and JavaScript rendering.

As an intermediary between your own web scraper/crawler and a target website, Web Unblocker allows you to gather publicly available data from the most complex websites without the risk of detection from advanced anti-bot solutions.

Core Features

Simple integration – Web Unblocker uses a simple backconnect proxy (a single-entry node) and is compatible with your existing code.

ML-driven access to public data – with the power of ML algorithms, Web Unblocker selects, rotates, and evaluates the most suitable proxies for a specific site to provide the highest possible success rate along with the lowest response time.

Dynamic fingerprinting – ensures organic traffic resemblance. The system selects the right combination of headers, cookies, browser attributes, JavaScript fingerprints, and proxies to appear as a real user, not triggering CAPTCHAs and bypassing target website blocks.

ML-powered response recognition – the tool creates an effective feedback loop between the scraping results and the experimentation engine to determine outcome quality.

Auto-retries – in case a scraping request fails, the system chooses another combination of client device parameters and sends the request again.

JavaScript rendering – the tool supports HTTP(S) requests to dynamic websites that use JavaScript for rendering content.

To improve the tool further and make better decisions, Oxylabs has formed a board of advisors that includes experts in data science, engineering, and AI.

Pujaa Rajan
Technical Staff at Anthropic, ex-Deep Learning Engineer at Node.io, USA Ambassador at Women In AI, and ML Expert at Google

Adi Andrei
Director at Technosophics, ex-Senior Data Scientist and Consultant at NASA, Unilever, British Gas

Jonas Kubilius
Co-founder and CTO at HERlab, Co-founder at Three Thirds, Artificial Intelligence Researcher

Ali Chaudhry
Founder at Veracious, PhD, Artificial Intelligence at UCL

Gautam Kedia
Head of Fraud Intelligence at Stripe, ex-Applied Scientist Lead at Microsoft, ex-Head of Applied Machine Learning at Lyft, ex-Engineering Manager at Stripe

Conclusion

Web scraping is a complex data acquisition process. As websites look for new ways to differentiate between bots and humans, those looking to scrape public data must discover an approach to help their scripts meld with regular internet users.

We have provided several ways to accomplish such a goal. These simple yet effective procedures will allow both veterans and newcomers to get more out of each web scraping script. Of course, these methods are not an exhaustive list. Creating one is nearly impossible as each new target has its own specificity.

However, by applying our tips and understanding the anti-bot measures utilized by your targets, you can significantly increase the data acquisition success rate. If constantly keeping track of website changes and the newest developments in web scraping seems futile and you would rather spend time analyzing data rather than acquiring it, contact us! Oxylabs are always ready to help businesses supercharge their data acquisition process.

oxylabs®

Want to Know More?

If you would like to know more about any of the topics mentioned in this white paper or learn about our products, please get in touch! Our team is ready to answer any of your questions and offer you the best solution for your business needs.

Get in touch with Oxylabs