Undetected ChromeDriver vs. Selenium Stealth: Which to Choose

August 26, 2024 · 7 min read

Tired of getting blocked while web scraping? Undetected ChromeDriver and Selenium Stealth are two powerful tools designed to help you bypass anti-bot measures. But which one should you choose?

This article compares Undetected ChromeDriver vs. Selenium Stealth, evaluating their effectiveness, popularity, and ease of use.

Let's dive in and see how these tools stack up against modern anti-bot systems.

Undetected ChromeDriver vs. Selenium Stealth Comparison

Undetected ChromeDriver and Selenium Stealth are both tools designed to enhance Selenium's ability to avoid detection during web scraping. Undetected ChromeDriver is a custom ChromeDriver that modifies browser properties to evade anti-bot measures, while Selenium Stealth is a plugin that alters Selenium's identifiable characteristics to make automation less detectable.

These tools share the same goal: helping your Selenium scripts bypass anti-bot systems. Both modify browser fingerprints and mask automation indicators. However, they differ in their implementation and ease of use. Undetected ChromeDriver requires replacing the standard ChromeDriver, whereas Selenium Stealth integrates as a plugin to your existing Selenium setup.

Now, let's review the key differences and similarities between Undetected ChromeDriver and Selenium Stealth in more detail.

. . .

Undetected ChromeDriver Has Higher Success Rate Against Anti-bots

Both Undetected ChromeDriver and Selenium Stealth aim to bypass anti-bot measures, but they make different claims about their capabilities.

Undetected ChromeDriver promises to avoid detection by Web Application Firewalls (WAFs) like Distill Network, Imperva, and DataDome. It does this by patching the ChromeDriver to

To compare their effectiveness, we tested both tools against CreepJS, a fingerprinting service that can detect headless browsers. Here's the code we used for these tests:

For Undetected ChromeDriver:

undetected_chromedriver_scraper.py

```python
# pip3 install undetected-chromedriver
import undetected_chromedriver as uc
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# set up the driver in headless mode
options = uc.ChromeOptions()
options.add_argument("--headless=new")
driver = uc.Chrome(options=options)

# navigate to the CreepJS test page
driver.get("https://abrahamjuliot.github.io/creepjs/")

# wait for the results to load
WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID, "headless-resistance-detection-results"))
)

# take a screenshot of the results
element = driver.find_element(By.ID, "headless-resistance-detection-results")
element.screenshot("undetected_chromedriver_results.png")

driver.quit()
```

For Selenium Stealth:

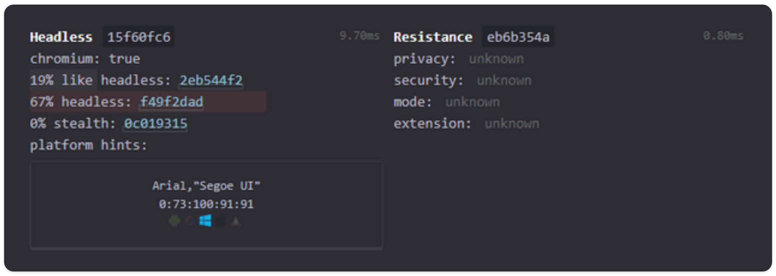
selenium_stealth_scraper.py

```python
# pip3 install selenium-stealth
from selenium import webdriver
from selenium_stealth import stealth
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# set up the driver in headless mode
options = webdriver.ChromeOptions()
options.add_argument("--headless=new")
driver = webdriver.Chrome(options=options)

# apply stealth settings
stealth(
    driver,
    languages=["en-US", "en"],
    vendor="Google Inc.",
    platform="Win32",
    webgl_vendor="Intel Inc.",
    renderer="Intel Iris OpenGL Engine",
    fix_hairline=True,
)

# navigate to the CreepJS test page
driver.get("https://abrahamjuliot.github.io/creepjs/")

# wait for the results to load
```

```
driver.quit()
```

The results from CreepJS revealed:

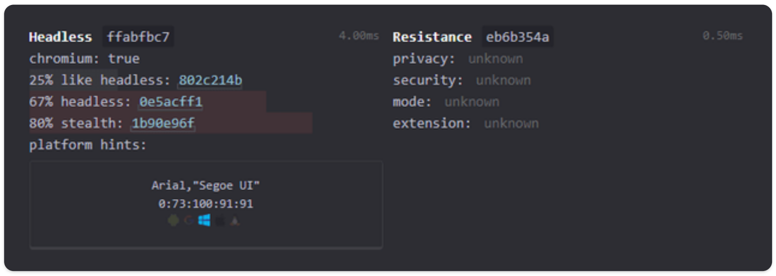
Undetected ChromeDriver:



```
Headless 15f60fc6                9.70ms   Resistance eb6b354a        0.80ms
chromium: true                            privacy: unknown
19% like headless: 2eb544f2               security: unknown
67% headless: f49f2dad                    mode: unknown
0% stealth: 0c019315                      extension: unknown
platform hints:
          Arial,"Segoe UI"
          0:73:100:91:91
```

Click to open the image in full screen

Selenium Stealth:



```
Headless ffabfbc7                4.00ms   Resistance eb6b354a        0.50ms
chromium: true                            privacy: unknown
25% like headless: 802c214b               security: unknown
67% headless: 0e5acff1                    mode: unknown
80% stealth: 1b90e96f                     extension: unknown
platform hints:
          Arial,"Segoe UI"
          0:73:100:91:91
```

Click to open the image in full screen

Undetected ChromeDriver exhibited fewer characteristics typically associated with headless browsers. Surprisingly, Selenium Stealth scored 80% in stealth, suggesting it didn't effectively mask its automated nature.

These results show that Undetected ChromeDriver outperformed Selenium Stealth.

However, it's crucial to note that neither tool gave satisfactory results against CreepJS (67% headless). This highlights an important point for web scraping at scale: while these tools can improve your chances of avoiding detection, they don't guarantee complete invisibility to all anti-bot systems.

When choosing between Undetected ChromeDriver and Selenium Stealth for large-scale scraping projects, consider that you may still encounter some blocks or detection, regardless of which tool you use. While Undetected ChromeDriver showed better performance in our tests, particularly in stealth, the effectiveness of these tools can vary depending on the specific anti-bot measures employed by different websites.

> **Frustrated that your web scrapers are blocked once and again?**
>
> ZenRows API handles rotating proxies and headless browsers for you.
>
> Try for FREE

boasts over 9,300 GitHub stars and more than 580,000 monthly downloads on PyPI.

In contrast, Selenium Stealth has around 620 GitHub stars and averages about 122,000 monthly downloads. This significant difference in stars and download counts indicates that Undetected ChromeDriver has captured a larger share of the developer community's attention and trust.

The popularity of an open-source tool like Undetected ChromeDriver brings several benefits. A larger user base means more people test the tool in different environments, helping find and fix bugs faster. Popular projects often get more volunteers who contribute new features and updates. This is crucial for web scraping tools, as they must keep up with constantly changing anti-bot measures.

Additionally, widely used tools usually have more tutorials, guides, and forum discussions. This makes it easier for newcomers to get started and for experienced users to solve problems.

. . .

Undetected ChromeDriver Is Better Maintained

Undetected ChromeDriver shows a much more active development cycle compared to Selenium Stealth. Looking at their GitHub commit histories, Undetected ChromeDriver has had regular updates throughout 2023 and 2024. These updates include bug fixes, feature additions, and compatibility improvements.

In contrast, Selenium Stealth's last commit was in November 2020. This stark difference in update frequency suggests that Undetected ChromeDriver is being maintained and improved, while Selenium Stealth appears to be in a state of low maintenance or potential abandonment.

The frequency of updates is crucial for anti-bot evasion tools. Tools that aren't updated frequently can quickly become ineffective as new detection methods are implemented.

. . .

Undetected ChromeDriver Is Easier to Use

Undetected ChromeDriver is easier to start with and use than Selenium Stealth. Its installation process is straightforward. Implementation is equally user-friendly, with minimal code changes needed to switch from standard Selenium usage. You can replace the existing `webdriver.Chrome()` initialization with `uc.Chrome()`, and the tool will handle most of the complex anti-detection measures automatically.

In contrast, while Selenium Stealth isn't overly complex, it does require more setup and configuration. After installation, you need to import the stealth module and manually apply various stealth settings. This process involves more code and a deeper understanding of browser fingerprinting techniques.

Best Alternative to Undetected ChromeDriver and Selenium Stealth

While Undetected ChromeDriver and Selenium Stealth offer valuable solutions for bypassing anti-bot measures, it's important to understand that **neither tool nor any other open-source solution can guarantee consistent access to the data you need**. This is because anti-bot technologies are in a constant state of evolution, with frequent updates and increasingly sophisticated detection methods.

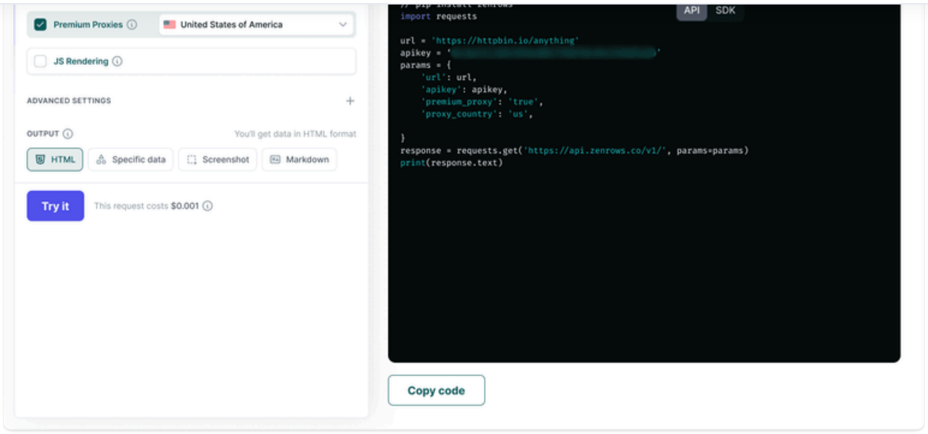
Anti-bots employ a wide array of techniques to identify and block automated access, from analyzing browser fingerprints to detecting patterns in user behavior. **As these detection methods become more advanced, even the most up-to-date open-source tools can struggle to keep up**.

For those seeking a more reliable and hassle-free approach alternative to Undetected ChromerDriver or Selenium Stealth, a dedicated web scraping API like ZenRows offers a compelling alternative. **ZenRows provides a comprehensive toolkit designed to overcome even the most robust anti-bot systems**. It goes beyond simply masking automation signatures, offering a suite of features that can replace the need for plugins, headless browsers, and complex configurations.

ZenRows not only stands strong against various detection methods but also streamlines the entire web scraping process, handling challenges like JavaScript rendering, CAPTCHAs, IP rotation, and more automatically.

To illustrate ZenRows' capabilities, let's look at how it handles scraping a heavily protected webpage like G2 Reviews.

Sign up for ZenRows, and you'll be automatically redirected to the Request Builder page. In the URL to Scrape box, input the target URL. Select the Premium Proxies and JS Rendering checkboxes. Choose Python as your programming language and select API mode.



```
import requests

url = 'https://httpbin.io/anything'
apikey = '[redacted]'
params = {
    'url': url,
    'apikey': apikey,
    'premium_proxy': 'true',
    'proxy_country': 'us',

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

Click to open the image in full screen

Here's the generated code snippet for scraping G2 Reviews using ZenRows:

Example

```
# pip install requests
import requests

url = "https://www.g2.com/products/jira/reviews"
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)
```

You'll get the following output on running this code:

Output

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <link href="https://www.g2.com/images/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <title>Jira Reviews</title>
    <!-- ... -->
</head>
<body>
    <!-- other content omitted for brevity -->
</body>
```

Awesome! This example demonstrates how ZenRows effortlessly bypasses G2's protection measures. It delivers the desired HTML content without the need for complex setups.

With ZenRows, you can focus on extracting and analyzing the needed data, rather than battling with anti-bot systems.

. . .

ChromeDriver showed better results and more active development, both tools have limitations. They may work in some scenarios but can't guarantee consistent success across all websites.

For a more reliable solution, consider ZenRows. This web scraping API offers a comprehensive approach to overcoming anti-bot measures without constant adjustments. Try ZenRows for free!

Web Crawling Webinar for Tech Teams

zenrows

Start Free Trial

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

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