

Forum:[Hack Forums](#)

Thread: [Building a Bruteforce Script in Python](#)

reb00t_inc - March 21, 2025, 11:09 am

I am attempting to write a bruteforce script in Python for a login page that is protected by security measures from Nudata and Arkose Labs, including sensitive CAPTCHA triggers and request rejections designed to prevent unauthorized access.

Login Request Details:

To successfully interact with the login API, the request requires the following parameters in the payload:

```
Code {  
  "email": "string",  
  "password": "string",  
  "nds_pmd": "string",  
  "nudata_session_id": "string"  
}
```

- `nds\_pmd` and `nudata\_session\_id` are crucial values for the request to go through.

Headers and Cookies:

In addition to the payload, the request requires specific headers, which are dynamically generated from other forms of validation, including:

- `device\_intelligence`
- `challenge\_context`

These values are obtained through individual requests to the site.

The request also requires valid cookies for authentication, which I have attempted to generate using different methods. However, these generated cookies have not been accepted, leading to failed login attempts.

Current Attempts:

1. I've tried generating these values using requests made to the site in an attempt to mimic the site's login process.
2. I have also attempted to generate cookies randomly, but the cookies I'm using for my requests are invalid.

Next Steps:

My current thought is to use Selenium or another headless browser library to extract the cookies directly from the browser for repeated use. However, before I proceed with this method, I am wondering if there is an alternative solution to using a headless browser that would allow me to extract valid authentication keys and tokens programmatically by mimicking the site's request process in my Python script.

holofernes - March 22, 2025, 8:44 pm

I would honestly use something like cURL with a shell script and the "-b" flag to supply a cookie to brute-force the login. Python can be a bit slow, but your bottle-neck might be at the website level instead of the operating system level. You're right in that the trick is to get a valid cookie; does the login screen reject you if you want to proceed without a cookie? Is it the same cookie for a limited time, or is it one cookie per attempt? You could write a subroutine to grab cookies, and then another to make attempts, make them work on separate threads, and it should work.

Tyr - March 23, 2025, 12:40 am

Using Selenium? Simulate the login flow using cookie extraction

```
Code from selenium import webdriver
from selenium.webdriver.common.keys import Keys
```

```
# Initialize the WebDriver (using Chrome in headless mode)
options = webdriver.ChromeOptions()
options.add_argument("--headless")
driver = webdriver.Chrome(options=options)
```

```
# Visit the login page
driver.get("https://example.com/login")
```

```
# Simulate login form submission
email_input = driver.find_element_by_name("email")
password_input = driver.find_element_by_name("password")
email_input.send_keys("your_email@example.com")
password_input.send_keys("your_password")
password_input.send_keys(Keys.RETURN)
```

```
# Wait for the page to load and capture cookies
cookies = driver.get_cookies()
print(cookies)
```

```
# Extract the necessary session values and headers
session_cookie = next(cookie for cookie in cookies if cookie["name"] == "session_id")
# Now you can use the session cookie in subsequent requests
```

But you need to get the data first. Once you've identified the exact headers and cookies required, you can extract the values and include them in your requests using the requests library and also fingerprinting mechanisms (ie device intelligence or challenge context) u should also consider which would require mimicking the exact request headers generated in a real session

HorseShit - March 23, 2025, 10:56 am

Tyr knows wats up. Use selenium or even just the python web call depending on your skill and how much time you want to spend on this, but here's a good tip that works: Use an AI chat bot to help you write it. Just dont be dumb about it and say "oh hey im building this captcha bypass" call it something else and if it figures out what you're doing just lie and it'll believe you.

katzadmin - March 23, 2025, 2:43 pm

You can use Javascript + Puppeteer + webdriver bypass, it's mostly the best way to scrape or bypass any captcha. If you want to use Python with Selenium, basically download a Chromium build, then launch it with debugging port, after connect with the debugging port. or use with basic ws control, if you do this thing with webdriver bypass, you can easily bypass Turnstile, recaptcha invisible, etc.

reb00t_inc - March 28, 2025, 8:10 am

(Mar 22, 2025 08:44 PM)holofernes Wrote: [\[CITE\]](#) I would honestly use something like cURL with a shell script and the "-b" flag to supply a cookie to brute-force the login. Python can be a bit slow, but your bottle-neck might be at the website level instead of the operating system level. You're right in that the trick is to get a valid cookie; does the login screen reject you if you want to proceed without a cookie? Is it the same cookie for a limited time, or is it one cookie per attempt? You could write a subroutine to grab cookies, and then another to make attempts, make them work on separate threads, and it should work.

(Mar 23, 2025 12:40 AM)Tyr Wrote: [\[CITE\]](#) Using Selenium? Simulate the login flow using cookie extraction

...

Thanks for the reply guys! I managed to extract the valid cookies from the site, which allowed me to mimic the requests and retrieve tokens and keys as needed. However, I'm encountering a new issue now.

The cookies let me replicate requests that return the necessary tokens, but the site employs Arkose Labs anti-bot protection. For one of the requests, it requires a fingerprint (BDA) to validate the request. I found an online bypass package "azureflow [\[.\] github \[\\[.\\] io/arkose-fp-docs/arkose_re_docs.html\]\(#\)" that claims to be able to repackage the BDA fingerprint for reuse in other requests.](#)

Unfortunately, I can't seem to get it to work. Has anyone run into this issue and solved it or have an alternative suggestions on how to resolve it?

Tyr - March 28, 2025, 2:20 pm

(Mar 28, 2025 08:10 AM)reb00t_inc Wrote: [\[CITE\]](#) Thanks for the reply guys! I managed to extract the valid cookies from the site, which allowed me to mimic the requests and retrieve tokens and keys as needed. However, I'm encountering a new issue now.

The cookies let me replicate requests that return the necessary tokens, but the site employs Arkose Labs anti-bot protection. For one of the requests, it requires a fingerprint (BDA) to validate the request. I found an online bypass package "azureflow [\[.\] github \[\\[.\\] io/arkose-fp-docs/arkose_re_docs.html\]\(#\)" that claims to be able to repackage the BDA fingerprint for reuse in other requests.](#)

Unfortunately, I can't seem to get it to work. Has anyone run into this issue and solved it or have an alternative suggestions on how to resolve it?

Hmm, cannot say I've experienced the same issue but what I do know is that some sites generate BDAs that are time-sensitive. If you're capturing and replaying an old fingerprint, it might be invalid by the time you reuse it. Do you know the TTL value by any chance? You could also try intercepting and analyzing the fingerprint request with mitmproxy or Burp to see what specific data the site expects.

Best of luck friend.