Web Scraping: Deep Dive

Mohameddin H.Bashir
LinkedIn:
https://www.linkedin.com/in/mohameddin/
Twitter:@deanjeager
Discord:_.d34n

Introduction

- Scraping - the process of extracting data from websites for various reasons like business intelligence, threat intelligence, data science etc.
- Scope and intent usually determine the nature of the scraping method/technique to be used.
- It is a legal activity but there are restrictions that are website specific.
- Before scraping it is important to read through the website’s acceptable use policy and get the do’s and don’ts from the site.

Use cases

- Market Research
- Lead generation for businesses
- Competitive analysis
- Data Analysis
- Vulnerability research
- Recon
- etc

Prerequisites to web scraping

Some of the concepts you need to understand to effectively scrape sites include:

- Javascript, html and css
- Python preferably
- Other languages like rust can be used

Static vs Dynamic websites:

Static - content remains fixed unless manually updated

Dynamic site - content changes as user interacts with the site

Python for web scraping

- Introduction
- Python is a versatile language that can be useful for web scraping.
- Benefits of using python over other languages
  - Rich community, Good library support, Integration with other Data analytic tools, Cross-platform compatibility
- Interacting with browsers can be quite convenient

NOTE: Making many repeated requests to a website's server may use up bandwidth, slowing down the website for other users and potentially overloading the server such that the website stops responding entirely.

Fundamental Libraries

- Beautiful Soup
- Requests
- Json
- CSV
- Selenium

WebDrivers and headless Browsers

- Selenium - test automation framework used mainly to spawn browser instances and can concurrently run tests.
- Pyppeteer - Python Puppeteer alternative. Rich python tool that can spawn browser instances
- Web driver - tool/API that can programmatically allow one to interact with and control behaviours of web browsers
- Headless Browser - mode of operations of browsers where you can interact without a GUI, generally faster and more efficient than non-headless browsers as no contents are loaded.

Automation Using Python

- Test cases (UI testing, integration tests, cross-browser test, performance test etc) and API interactions for developers
- Bug hunters application - Automated scanning for various vulnerabilities (XSS, CSRF), integrating with other tools and fuzzers and scope management (keeping track with updates in your scope)
- Can also be useful in testing out logins and other injection vectors
- Red teamers application - Credential harvester, attack surface enumeration, threat intelligence by getting information from various sources.
- Also useful in planning and conducting phishing campaigns

Anti-scraping evasion

- Undetected_chromedriver - chromedriver that bypass common fingerprinting techniques used by anti-scrappers and anti-bot by emulating human behaviour
- Headless browser - using puppeteer and other browser instances can be useful in evading anti-scrappers
- Rate limiting - regulate how much traffic you send at any single point. This is important to also ensure that you do not consume too much bandwidth.
- Apify - good scraping platform for common social media sites
- Proxies and VPN

Control measures

- Cloudflare captchas
- Encryption and obfuscation
- Robots.txt to disallow certain endpoints
- Rate limiting and IP blocking

Resources

https://github.com/ultrafunkamsterdam/undetected-chromedriver

https://github.com/ansani/shodan_scrape

https://www.zenrows.com/blog/web-scraping-ajax#what-is-ajax

https://console.apify.com/r/default-page

https://miyakogi.github.io/pyppeteer/

Closing Remarks