Proceedings of the Third International Conference on Electronics and Sustainable Communication Systems (ICESC 2022)
IEEE Xplore Part Number: CFP22V66-ART; ISBN: 978-1-6654-7971-4

2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC) | 978-1-6654-7971-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICESC54411.2022.9885689

Web Scraping Approaches and their Performance on Modern Websites

Ajay Sudhir Bale
Dept. of Electronics and Communication Engineering
New Horizon College of Engineering
Bengaluru, India
ajaysudhirbale@gmail.com

Naveen Ghorpade
Dept. of Computer Science Engineering
Vijay Vittal Institute of Technology
Bengaluru, India
naveen.ghorpade@gmail.com

Rohith S
School of Computer Science and Engineering
Reva University
Bengaluru, India
rohithsatheesh@outlook.com

S Kamalesh
School of Computer Science and Engineering
Reva University
Bengaluru, India
kamaleshsathyanath@gmail.com

Rohith R
School of Computer Science and Engineering
Reva University
Bengaluru, India
rohithr.9t@gmail.com

Rohan B S
School of Computer Science and Engineering
Reva University
Bengaluru, India
bsrishirohan007@gmail.com

***Abstract*—When it comes to information, the internet is a gold mine. Whether you need data for your business, school, or personal use, you may uncover a wealth of information by performing an internet search. Web Scraping (WS) is a computerized method of obtaining big amounts of information from internet sites. The bulk of this information is in the form of unstructured HTML that would be transformed to structured data in a digital document before use in diverse applications. Although web scraping is not illegal, it might go against the ethics. This paper contains research-based findings of different methods of web scraping techniques used to extract data from websites. The various approaches used in this paper to obtain the results include requests library, selenium and other external libraries. Using the results obtained the findings are visualized through different graphs having contrasting parameters as measurements. Results obtained from our paper show that there is a lot of improvement needed to protect websites from web scraping bots.**

***Keywords*—Web scrapping, unstructured, structured, HTML, Selenium**

I. Introduction

Bots account for about half of all internet traffic, according to Bot Activity Report, and two-thirds of bot traffic is malevolent [1]. WS is among the techniques bots can cause damage to organisations. Bots typically achieve this by scanning a page, gaining access to its source code, and afterwards interpreting it to extract information they seek [2-5]. They generally post stuff somewhere on the web following receiving it [6-10]. Database scraping is a much more advanced version of scraping. This really is identical to site scraping in principle, however an attacker creates a bot that communicates with the destination site's app to collect information from its database [11-13]. Whenever a bot visits a provider's website to obtain rates on policies, this is an example of database scraping [14-17].

In this research-based article we have used 7 methods of web scraping and creating bots to extract data for research purposes only, to test how the websites behave when bots are created using different approaches. For this experiment we have listed top 15 websites from 8 categories which include e-commerce, educational, real-estate, stock-data, movie, sport, news and travel websites. These websites went through rigorous tests from 7 methods of web scraping. However, no matter what approach was used, the main objective was to capture web data. We have classified each of the results in numerous parameters which are listed as follows:

a. Number of requests: Number of requests it took for a website to detect the bot.

b. Time-based results: Total time the bot was active in the website before getting detected.

c. Data-extraction results: This parameter classifies if the bot had extracted useful data from the website or not.

The study is classified into 7 parts where each part defines a web scraping approach, and at the end the results obtained is visualized through graphs for better understanding of the method and its potency.

II. Approach and Tools

A. Approaches

Understanding how web scraping works in general is essential to arrive at various approaches which are followed in this paper. The basics of it still lies in altering various parameters which are used while making a request to an endpoint and make it seem like a legitimate request. Usually, a good percentage of websites return a successful response when a simple http request is made to their server, but many of them have mechanisms in place to differentiate between an actual request and a call made through a bot user. This paper analyses various web scraping approaches starting from the least to the most successful with respect to data extraction.

a) Basic requests library tests: This is the most common technique used to scrape data from a website. Using the python requests library, a simple HTTP request is made to the list of websites and then the page contents are stored along with the response status codes.

b) Requests library with header: This test is similar to the previous one, but we add a user-agent header parameter to the request. This is used by most of the websites to identify the requesting party. If no user-agent is mentioned, the requests package by default mentions a user-agent header in the following format python-requests/package-version.

Many websites identify these kinds of user-agents and blacklist them. As a workaround, web scraping scripts alter the user agent parameters to values like Mozilla/5.0, Chrome/100.0.4896.127, Safari/537.36 etc. to make it seem like the request is coming from a browser. It is fascinating how a single header added to a request can vary the results.

c) Basic selenium test: Selenium is an automation tool widely used in the web scraping world, here we try to access all the websites using a chromedriver, without any additional parameters. There are different drivers available for almost all the modern web browsers to perform selenium testing, but we have used chromedriver across all the selenium tests as it is the most widely used. It is observed that selenium in general is better for web scraping as most websites today have dynamic content loaded through JavaScript, which the request library is not able to access.

d) Selenium with header: This is a slight extension to the previous test, a user-agent header is added to the selenium request made to the websites.

e) Headless selenium test: When a browser is run on headless mode, no graphical interface is shown and they run in the background. Most web scraping scripts are run using the headless mode in production as they are executed for very long durations. Headless browsers are useful as they reduce the memory consumption and increase the overall performance of automation scripts. In this test, an additional headless parameter was added while making the selenium request. This method was added to the test as many modern-day websites have started to identify headless browsers as bots.

f) Headless selenium test with window-size argument: This argument is used frequently as many websites have started to identify headless browsers as bots, the window-size argument might help in such scenarios and also provides the capability to interact with the web elements in headless mode.

g) External library test (undetectable-chromedriver): This is a relatively new library which offers a patched version of the standard chromedriver, it performs exceptionally well where the website is protected using anti-bot services and tools, so it was considered to check its effectiveness in real world scenarios. This library patches the driver in such a way that keywords which are used by anti-bot services to identify selenium drivers are modified to prevent detection.

h) Timed tests using selenium: In all the previous tests, each website was accessed only once by the scraper script. Scripts in the real world are executed for hours or even days together to gather data from websites, to simulate that behaviour, each website from the various categories were requested for a fixed number of times using the selenium library, the time and request count was calculated before the website stopped responding to the requests made.

B. Tools

Python is the most popular programming language used for web scraping in the modern world as it offers a wide range of packages for data extraction with requests and selenium library being the most popular, along with this we have also used a relatively new library called the undetected chromedriver to perform our tests.

a) Requests library: Requests is a simple HTTP library for the Python programming language which enables the use of all standard HTTP methods. It also allows you to add additional parameters like headers, data, files, form data, timeout, etc.

b) Selenium library: Selenium is an open-source project for a range of tools and libraries for supporting browser automation. The selenium web driver supports all the modern web browsers and comes with built in tools and functions to make web automation very easy.

c) Undetected-chromedriver library: Anti-bot services such as Distill Network, Imperva, DataDome, and Botprotect.io are not triggered by the optimised Selenium Chromedriver patch. Downloads and fixes the driver binary automatically [17]. Works on a range of chromium based browsers.

III. IMPLEMENTATION

To make the research relevant to real world scenarios, a list of 8 top categories of websites were selected based on their vulnerability to web scraping. 15 top websites were chosen in each category adding up to 120 and they were kept constant throughout all the tests. The tests were performed using a single machine running on the same network to maintain consistency. The flow diagram is shown in Fig.1.

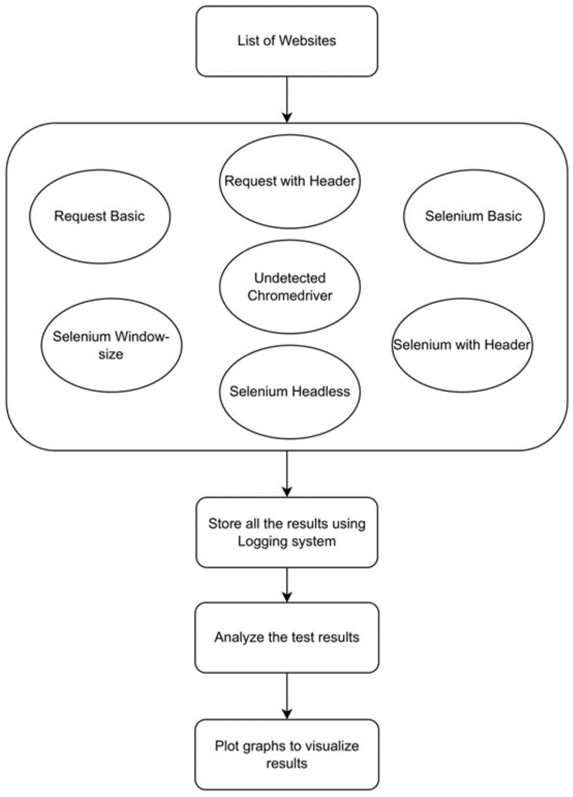

Fig. 1. Flow chart of the whole work

In the first step we are collecting top 15 websites each in 8 different categories, using these websites we are conducting tests on various methods of web scraping and after the tests have been run on these websites we are storing the results obtained from the tests through logging system, and using results obtained we are analysing the results on which parameters the results will have high impact and plotting the graphs on those parameters to visualize the results obtained.

Requests-library tests: The requests sent to the web- sites were checked for the response code sent back by the server. A common timeout of 5 seconds was set for the requests made as it is more than enough to load a website and to also maintain consistency throughout the tests. On receiving a 200 response code from the server, or encountering a timeout, the page source obtained was stored for further validation.

There were two tests performed using the requests library, one being a plain request without use of any additional parameters and the second one being the requests with header test, where an user-agent was added to the request made. Along with the source code, even exceptions which might have occurred in the process were stored. The tests on average took around 5 minutes for 120 websites.

a) Selenium and external library tests: As selenium does not provide a convenient way to check for response code sent by the server, a standard load time of 10 seconds was set for all the websites considering the load times for the chromedriver, after which the page source code was stored. In these tests as well, timeout and other exceptions have been captured.

Using the standard selenium library, 4 tests were performed which included a basic test without any header, using user- agent, using headless mode and the last one being the test with a window-size argument added to the selenium request. These tests on an average took around 20-25 minutes for 120 websites and were pretty CPU and memory intensive as chromedriver is a heavy tool.

For the undetected-chromedriver test, requests were made without using any parameters. The library is patched in such a way that it opens a new chrome session for each request made, making it significantly different that the default chromedriver used in the above tests. This test also took the same times as the other selenium based tests.

b) Timed-tests: A similar approach was followed for the timed tests as well, but the difference being that the number of requests before which the server fails to respond, along with the total time the scraper was active on a particular website was taken into account.

Top 10 websites from each category were chosen and 150 calls were made to each of them continuously. This was by far the most time consuming test, which took more than 15 hours to complete execution, conducted in 2-3 parts.

c) Evaluation of results: The scripts running various types of tests recorded all the parameters necessary for evaluation such as response code, exceptions, timeouts and so on into respective csv files. The resulting page source code was parsed using the Beautiful Soup library and stored as a html file.

These files were manually verified later to check if the source code obtained had all the data present in the actual website, and then it was determined if the data extraction was successful or not. All the data then was grouped together to form insightful graphs to show the final results.

IV. RESULTS

This section contains visualizations of the results obtained through the tests. Different graphs displayed here will be helpful in understanding the objective of this article in a much easier way.

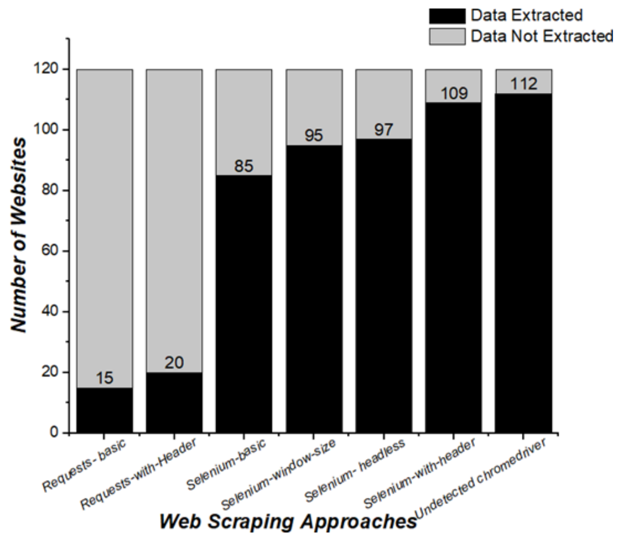

Fig. 2. Performance of Web Scraping approaches.

The Fig.2 shows the effectiveness of each web scraping method. The number of websites where data has been extracted successfully increases from left to right with requests library being the least effective and undetected chromedriver obtaining the highest amount of data extraction.

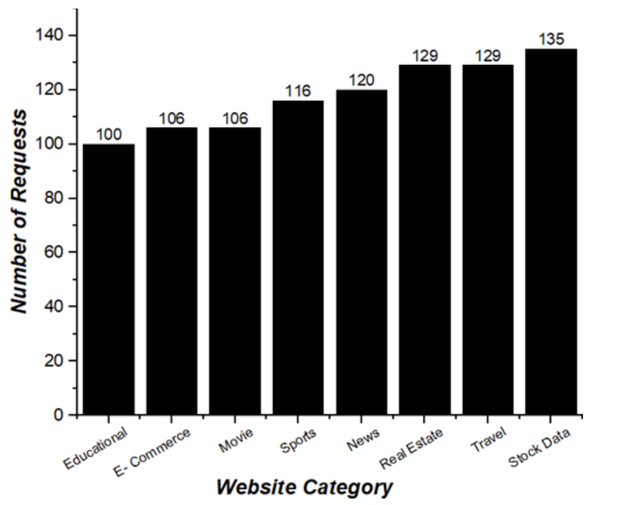

Fig. 3. Category-wise average number of requests made before detection.

The Fig.3 shows category-wise trend of the average number of requests made by the scraper before being detected by the website. A total of 150 requests were made to the individual websites in each category. It can be observed that certain categories of websites are able to identify and block bots much faster than other categories and have a significantly better protection mechanism.

The Fig.4 shows the category-wise trend for the amount of time the scraper was active on a particular website before getting blocked. The time spent by the bot on a website is an important metric as scripts in real-world scenarios are run for long durations to parse through and extract all the data available on a website. These results follow a similar trend as the previous graph, where educational and e-commerce provided the best protection compared to other categories.

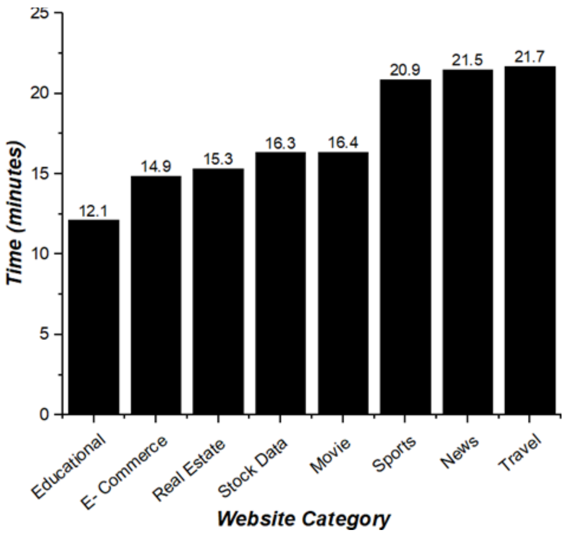

Fig. 4. Category-wise average time taken before detection.

V. CONCLUSION

This article evaluates the effects of various web scraping methods and techniques on modern day websites and reveals that a very big percentage of them are prone to these kinds of bot attacks. The results also show why selenium is the most widely used tool for web scraping as most modern-day websites render content dynamically which might not have been the case before a few years. It also brings to light the current state of protection which is present in the existing infrastructure of modern websites.

The trend depicted through the category wise results shed light towards the categories of websites that are more susceptible to bot attacks today and also the categories which are significantly better in comparison to others with respect to security.

To make the results more accurate, additional modules or libraries can be identified to perform similar tests, the results will be much more evident and distinguishable when the tests are conducted on a very large number of websites. Further- more, additional parameters can be added for selenium-based tests to check for their effectiveness, or multiple parameters can be combined together to make requests.

An automated system can be setup to check if the data extraction was successful. This will be very helpful while extending the tests to a large set of websites. With regards to timed based tests, interactivity and navigation capability can be added and the number of requests made can be significantly increased.

REFERENCES

[1] Se´ny Ndiaye, Mamadou, Babiga , Ousmane , Edouard Ngor , Rabiyatou," Web Scraping: State-of-the-Art and Areas of Application,"2019.

[2] Korawit Prutsachainimmit, Winai Nadee," Towards Data Extraction of Dynamic Content from JavaScript Web Applications," 2018 International Conference on Information Networking (ICOIN), Jan. 2018, 10.1109/ICOIN.2018.8343218.

[3] Eric C. Dallmeier," Computer Vision-based Web Scraping for Internet Forums," 2021 7th International Conference on Optimization and Ap- plications (ICOA) ,May. 2021, 10.1109/ICOA51614.2021.9442634.

[4] Erdinc¸ Uzun," A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages," IEEE Access ,vol. 20,April 2020, 10.1109/ACCESS.2020.2984503.

[5] Wahyudin Darmalaksana, Rosihon Anwar, Diah Wulandari, Agung Wa- hana, Wisnu Uriawan," Pearson Correlation Method and Web Scraping for Analysis of Islamic Content on Instagram Videos," 2020 6th Inter- national Conference on Wireless and Telematics (ICWT), Sept. 2020, 10.1109/ICWT50448.2020.9243626.

[6] Khusnul Khotimah, Nurul Fuad, Mohammad Robihul Mufid, Arif Basofi, Saniyatul Mawaddah," Risk Diagnosis and Mitigation System of COVID-19 Using Expert System and Web Scraping," 2020 International Electronics Symposium (IES), Sept. 2020, 10.1109/IES50839.2020.9231619.

[7] David Mathew Thomas, Sandeep Mathur," Data Analysis by Web Scraping using Python," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), June. 2019,10.1109/ICECA.2019.8822022.

[8] Volker Staudt, Marcel Gladen," Using a web scraping algorithm for component model generation in multiobjective optimization of power electronic applications," 2020 22nd European Conference on Power Electronics and Applications (EPE'20 ECCE Europe), Sept. 2020, 10.23919/EPE20ECCEEurope43536.2020.9215702.

[9] Monique Ross, Jia Zhu, Stephanie Lunn," Utilizing Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice," 2020 IEEE Frontiers in Education Conference (FIE), Oct. 2020, 10.1109/FIE44824.2020.9274270.

[10] Stephanie Lunn,Jia Zhu, Monique Ross," Utilizing Web Scraping and Natural Language Processing to Better Inform Pedagogical Practice," 2020 IEEE Frontiers in Education Conference (FIE), Oct. 2020, 10.1109/FIE44824.2020.9274270.

[11] Eric C. Dallmeier," Computer Vision-based Web Scraping for Internet Forums,"2021 7th International Conference on Optimization and Appli- cations (ICOA),"May. 2021, 10.1109/ICOA51614.2021.9442634.

[12] Tawfiq Barhoom, Noor Bolbol," Mitigating Web Scrapers using Markup Randomization," 2021 Palestinian International Conference on Informa- tion and Communication Technology (PICICT),"Sept. 2021, 10.1109/PI- CICT53635.2021.00038.

[13] Ankit Sanghvi, Sachin Bojewar,Dipali Shete," Survey Paper on Web Content Extraction and Classification," 2021 6th International Conference for Convergence in Technology (I2CT), April 2021, 10.1109/I2CT51068.2021.9417947.

[14] Rida Yaqoob, Sanaa, Muhammad Haris, Samadyar, Munam Ali Shah," The Price Scraping Bot Threat on E-commerce Store Using Custom XPATH Technique," April 2021, 10.1109/I2CT51068.2021.9417947.

[15] Chun Feng Lin,Sheng Chih Yang," Web Scraping to Implement Tape Reading on Taiwan Stock Periodically with GUI," 2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE), Oct. 2021, 10.1109/ECICE52819.2021.9645633.

[16] Bhavya Bhardwaj,Syed Ishtiyaq Ahmed,J Jaiharie,R Sorabh Dadhich,M Ganesan," Web Scraping Using Summarization and Named Entity Recognition (NER)," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), March 2021,10.1109/ICACCS51430.2021.9441888.

[17] "Undetected-chromedriver," PyPI [Online]. Available: https://pypi.org/project/undetected-chromedriver/ . [Accessed: 01-May-2022].