IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Effective Indexing Of Tor Network Using Distributed Scraping.

Vedprakash Upadhye[1], Abhishek Pawar[2], Yash Ingole[3], Pritish Marathe[4], Prof.H.P.Bhabad[5]

Loknete Gopinathji Munde Institute of Engineering Education and Research, Nashik

Abstract - Web scraping is the process of extracting information from websites. It can be used to collect lots of data from websites that do not provide an API or allow only limited access to data. It can be used for many purposes, including web scraping, data mining, market research, and customer service. There are many ways to do web scraping. One way is to use online services that provide web download tools.
Another way is to use an API that provides access to data from the web. Finally, web scraping can be done manually by collecting the code associated with the website. Web scraping can be a difficult and time-consuming task. However, there are many tools and services that can simplify the process. These tools and services can help speed up the web data extraction process and also help avoid the limitations of web scraping from the web. Web scraping is a powerful tool that can be used to gather information from websites. However, it is important to use web scraping ethically and responsibly. It is important not to enter sites that do not allow the site to be downloaded, and not to enter sites in a way that harms the site or its users.

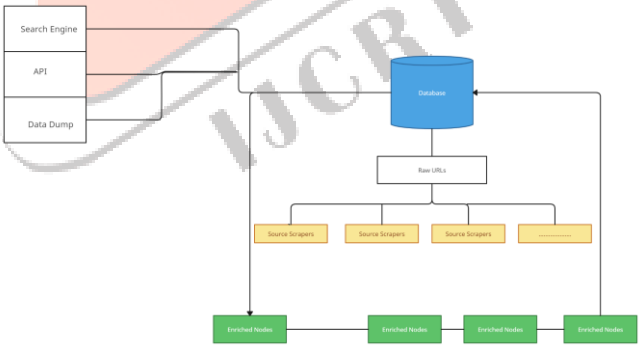
***Keywords*:** Web Scraping, Data Extraction, Tor, Onion, distributed scraping, Effective Indexing, Cybersecurity, Threat Intel, Data Mining.

1. Introduction

Web scraping for onion links using the REST API is a promising solution to solving the darknet problem that has not been analyzed much and is inaccessible by traditional research. This project aims to explore the various applications and consequences of this approach. By leveraging REST APIs, researchers, analysts and investigators can gather valuable information from the dark web to uncover insights, trends and potential hidden threats. However, it is important to consider legal, ethical, and legal considerations when web scraping for onion links.

In the previous base paper, the indexing of the onion links was not considered. This limitation hinders efficient search and retrieval of relevant information from the dark web. To overcome this drawback, this paper seeks to introduce an indexing mechanism for onion links with an added layer of enrichment. By implementing indexing, researchers, analysts, and investigators will be able to organize and categorize the scraped onion links, allowing for easier navigation and retrieval of specific information. Furthermore, the enrichment of the indexed links will enhance the analysis and understanding of the gathered data. This enrichment process may involve associating metadata, extracting relevant keywords, or applying advanced techniques such as natural language processing or sentiment analysis. These enriched indexes will provide valuable insights, trends, and potential hidden threats within the darknet. While exploring the possibilities of web scraping for onion links, it is crucial to consider the legal and ethical implications. The darknet operates within a complex legal landscape, and caution must be exercised to ensure compliance with applicable laws and regulations. Additionally, ethical considerations, such as respecting privacy and avoiding any malicious intent, should guide the implementation and usage of this approach

2. Architectural Design



2.1 Architecture Design Diagram

1. Web crawling component: This component is responsible for navigating to the target website and retrieving its HTML code. The web crawler may follow links to other pages on the same site, or to external sites, in order to gather additional data.
2. Data extraction component: This component is responsible for parsing the HTML code retrieved by the web crawler and extracting specific data elements, such as text, images, and links. This component may use regular expressions, XPath queries, or other techniques to identify and extract data.
3. Data storage component: This component is responsible for storing the extracted data in a structured format, such as a database, spreadsheet, or

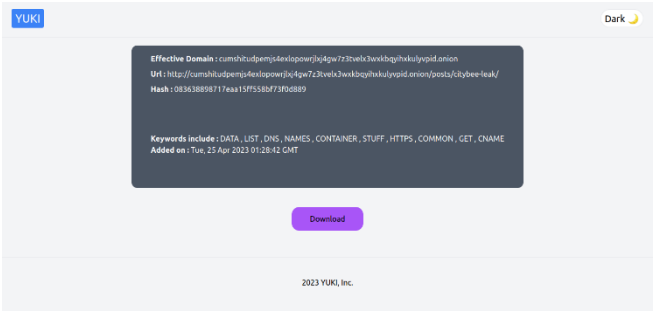
CSV file. This component may also perform data cleaning and transformation tasks to ensure that the extracted data is accurate and consistent.

4. User interface component: This component provides a graphical user interface (GUI) for the user to interact with the web scraper. The user interface may allow the user to specify the target website, select the data elements to be extracted, and configure other settings.
5. Scheduler component: This component manages the scheduling of web scraping tasks, allowing the user to specify when and how often the web scraper should run. This component may also provide notification and alerting features to inform the user of the status of web scraping tasks.

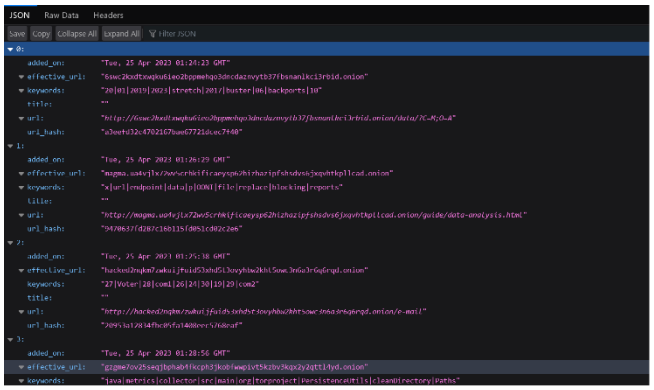
Overall, a web scraper is a software application that can be used to extract data from websites. It consists of five main components: a web crawler, a data extraction component, a data storage component, a user interface component, and a scheduler component. The web crawler is responsible for navigating to the target website and retrieving its HTML code. The data extraction component is responsible for parsing the HTML code retrieved by the web crawler and extracting specific data elements, such as text, images, and links. The data storage component is responsible for storing the extracted data in a structured format, such as a database, spreadsheet, or CSV file. The user interface component provides a graphical user interface (GUI) for the user to interact with the web scraper. The scheduler component manages the scheduling of web scraping tasks, allowing the user to specify when and how often the web scraper should run. Web scraping can be a powerful tool for gathering data from websites, but it is important to use it responsibly. Websites may have terms of service that prohibit scraping, and scraping too much data from a website can slow it down or even crash it. It is important to read the terms of service of any website before scraping it, and to be respectful of the website's resources.

3. Outcomes

3.1 Output Search Engine



3.2 Output Detail View



3.3 API Output

4. Future Scope

While the current implementation of the web scraping project is successful, there are several areas for future improvement and enhancement. Some potential future scope includes:

1. Enhancing Performance: Optimizing the web scraper to improve its speed and efficiency, especially when dealing with large volumes of data or slow websites.
2. Advanced Data Analysis: Incorporating advanced data analysis techniques to extract insights and patterns from the scraped data, providing more valuable information to users.
3. User-Friendly Interface: Enhancing the user interface to make it more intuitive and user-friendly, allowing users to easily configure scraping parameters and view the extracted data.
4. Expanded Platform Support: Extending the compatibility of the web scraper to support additional web browsers, operating systems, and platforms, ensuring a wider user base.
5. Continuous Monitoring and Maintenance: Implementing a monitoring system to regularly check the functionality and performance of the web scraper, and performing maintenance and updates as needed.
6. Integration with Other Systems: Integrating the web scraper with other systems or APIs to enhance its functionality and enable seamless data exchange.
7. Scalability: Ensuring the web scraper can handle increased data volumes and user requests by implementing scalable architecture and infrastructure.

5. Methodology

Goals and Objectives:

The project aims to create a system that scrapes onion links from the TOR browser using REST APIs and indexes them in a structured database. The system will ensure data privacy and security, and be efficient and scalable. To achieve this, the project will develop a mechanism to access the TOR browser using REST APIs. The system will then parse the retrieved data to extract onion links. The onion links will then be stored in a structured database for efficient querying and searching. The system will also ensure data privacy and security in handling the onion links. Finally, the system will create an efficient indexing system for the dark web that can handle large volumes of data and is scalable for future expansion.

1. Develop a mechanism to access the TOR browser using REST APIs.
2. Parse the retrieved data to extract onion links.
3. Store the onion links in a structured database for efficient querying and searching.
4. Ensure data privacy and security in handling the onion links.
5. Create an efficient indexing system for the dark web that can handle large volumes of data and is scalable for future expansion.

6. Working

A web scraper is a software tool that automates the extraction of data from websites. It mimics the behavior of a human user by navigating through pages and retrieving specific information. Web scraping is used in a variety of domains, including data analysis, market research, and competitive analysis. It provides a means to gather large amounts of data from different sources quickly and efficiently.

1. Sending an HTTP request to the target website: The web scraper sends a request to the website's server, usually using the HTTP or HTTPS protocol.
2. Retrieving the HTML content: Upon receiving the request, the server responds by sending back the HTML code that makes up the web page.
3. Parsing the HTML: The web scraper parses the HTML code to understand its structure, including tags, classes, and IDs.
4. Extracting the desired data: Based on predefined rules or patterns, the web scraper identifies and extracts the required data from the parsed HTML
5. Storing or processing the extracted data: The scraped data can be stored in a database, exported to a file, or processed further for analysis or integration with other systems.
6. Web scrapers can be implemented using various programming languages such as Python, JavaScript, or Ruby, and they utilize libraries or frameworks like Beautiful Soup, Scrapy, or Selenium to handle the web scraping tasks.

7. CONCLUSION

In conclusion, the web scraping project on onion links using a REST API has been successfully implemented. The project achieved its objective of
extracting data from onion links and storing it in a structured database. The software was developed following the Waterfall model, ensuring a systematic and phased approach to development. Through verification and validation processes, the acceptance and reliability of the web scraper were ensured. Functional requirements were verified, data accuracy was validated, error handling and robustness were tested, and security measures were implemented. The software met the specified requirements and performed well under different scenarios.

8. REFERENCES

1. Zhang, X., Zhang, Y., Yao, L. (2019). Web Data Extraction Based on User Interaction Recognition and Deep Learning. IEEE Access, 7,132147-132157.
2. Hsieh, M. F., Chang, C. Y., Wang, M. J. J. (2018). An Efficient Web Data Extraction Framework for Online Price Comparison. IEEE Transactions on Knowledge and Data Engineering, 30(3), 465-477.
3. Han, B., Song, L., Yu, Y. (2020). Web Data Extraction by Leveraging Human Interaction Signals. IEEE Transactions on Knowledge and Data Engineering, 32(4), 742-756.
4. Chen, J., Chen, L., Liu, J. (2020). Web Data Extraction Based on Web Page Segmentation and Template Tree Matching. IEEE Access, 8, 78315-78326.
5. Wei, X., Zhang, Z., Jiang, H. (2019). A Web Data Extraction Approach Based on Deep Learning and Conditional Random Fields. IEEE Access, 7, 52749-52758.
6. Feng, Z., Xia, F., Liu, Y. (2021). Web Data Extraction by Learning Based Hybrid Approach. IEEE Transactions on Cybernetics, 51(1),320-333.
7. Lien, J. J., Hsieh, M. F. (2019). An Improved Web Data Extraction Framework for Product Comparison. IEEE Transactions on Knowledge and Data Engineering, 31(4), 710-723.
8. Chen, H., Liu, G., Zeng, D. (2019). An Effective Web Data Extraction Approach Based on Block Structure and Dependency Analysis. IEEE Access, 7, 27358-27369.
9. Ding, H., Liu, Y., Li, G. (2020). Web Data Extraction Based on Convolutional Neural Networks and Visual Features. IEEE Access, 8,223797-223808.
10. Zhang, J., Zhang, J., Qian, Y. (2019). Web Data Extraction Based on Template Learning and Semantic Annotation. IEEE Access, 7, 137350 137360.
11. Xu, D., Chen, H., Liu, Y. (2019). Web Data Extraction Based on Hierarchical Template Matching. IEEE Access, 7, 90324-90334.

12. Luo, Y., Wang, Z., Li, Y. (2021). Web Data Extraction Based on Visual Features and Sequence-to-Sequence Model. IEEE Access, 9,27208-27220.
13. Zhu, W., Hu, Q., Zhang, L. (2018). Web Data Extraction Based on Hybrid Information Extraction and Hierarchical Wrapper Generation. IEEE Access, 6, 29552-29562.
14. Yin, S., Zhu, X., Li, H. (2019). Web Data Extraction Based on Ontology Learning and Rule Reasoning. IEEE Access, 7, 150717-150729.
15. Li, Y., Tang, Z., Wu, Y. (2020). Web Data Extraction Based on Deep Learning and Multi-Instance Learning. IEEE Access, 8, 196308- 196319