ERSC
ENGENHARIA DE REDES E SISTEMAS DE COMPUTADORES
ESTG-IPVC

Portugal WordPress Fingerprint: Enhancing Website Analysis

Submited on 11/06/2023
Submited by
Romilson Monteiro nº 28891, Francisco Oliveira nº 22252, Marco Da Luz nº26476
Supervised by
teachers Hugo Almeida and Pedro Pinto

***Index Terms*—crawler, Google Hacking,Scrapy, Wordpress, spider .**

***Abstract*—This article presents the development of a web crawler designed to identify Portuguese websites developed using WordPress and extract information about the WordPress versions utilized. Additionally, a user-friendly web interface is built to facilitate the application's usage. The objective of this research is to provide a valuable tool for website administrators, developers, and security professionals to gather information about WordPress usage in the Portuguese web landscape.**

I. Introduction

The web crawler employs a systematic approach to search for Portuguese websites by leveraging specific characteristics commonly found in URLs and web page content. By examining the HTML source code, the crawler identifies WordPress-specific markers and extracts the version information. The gathered data is stored in a structured format for further analysis. To enhance usability and accessibility, a web interface is designed to interact with the crawler. Users can obtain a comprehensive list of Portuguese websites built with WordPress. The interface also allows users to filter and sort the results based on various criteria, such as WordPress version or website popularity. Additionally, the crawler verifies if a user-inputted site is developed in WordPress, providing valuable information about its platform. The proposed solution aims to provide valuable insights into the usage and distribution of WordPress in the Portuguese web ecosystem. In the figure (1), we can see the diagram of our project.

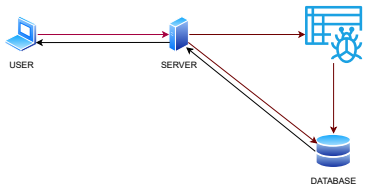

Fig. 1. Diagram

II. State of art

A. Crawler[1][2]

A crawler, also known as a spider, is an automated program or bot that systematically navigates the internet, crawling and analyzing web pages in a methodical manner. The primary goal of a crawler is to gather relevant information from a large number of websites in an automated fashion. Crawlers are widely used for various purposes, such as content indexing for search engines, data collection for analysis, monitoring changes in web pages, and more. The versatility and automation capabilities of crawlers make them valuable tools for efficiently exploring and extracting information from the vast expanse of the internet

B. Google Hacking[3]

Google Hacking is the practice of using advanced search operators and specific queries to uncover sensitive or hidden information that may not be easily accessible through normal search methods. It involves leveraging the power of Google's search engine to find vulnerabilities, exposed data, and security issues. It is commonly

employed by security professionals to identify potential risks and assess the security of websites, networks, and systems.

C. Scrapy

Scrapy is an open-source web scraping framework written in Python. It provides a comprehensive set of tools and functionalities for extracting data from websites in an automated and structured manner. With Scrapy, developers can create web spiders, which are specialized programs that navigate through websites, send HTTP requests, parse HTML or other response formats, and extract desired data. Scrapy supports data export to various formats, including CSV, JSON, and databases. It also integrates well with other Python libraries and frameworks, making it a popular choice for web scraping projects.

D. spider

In the context of a crawler, a "spider" refers to a component or program responsible for navigating the web, collecting information, and indexing web pages. It is used to describe a computer program that automatically traverses the internet, follows links, and visits various web pages.

E. WordPress

WordPress is a popular open-source content management system (CMS) for creating websites, blogs, and online stores. It offers a user-friendly interface and doesn't require advanced programming skills. With its intuitive dashboard, users can easily manage content, themes, and plugins.

III. Design and Implementation of Crawlers[4]

To develop the crawlers, we utilized the Python programming language along with various libraries and frameworks. Python is a versatile and widely used language known for its simplicity and efficiency in web scraping and data extraction tasks. In order to optimize the performance and avoid potential issues like Google's request denial due to excessive requests, we made use of the Scrapy API and ScrapeOps.

The crawlers were designed to fulfill specific objectives and provide valuable insights for our users. We carefully planned and implemented each crawler to ensure they effectively extract the desired data and provide accurate results. The following sections provide an overview of the design and implementation of each crawler.

A. Crawler: PT WordPress Fingerprint

To develop this web crawler, we utilized the **Scrapy** library, a powerful tool for web data extraction. The main objective of our web crawler is to search for Portuguese websites that use the **WordPress** platform as their content management system and verify the specific version of WordPress they are utilizing.
To implement this functionality, we divided the web crawler into two parts, each executed by a different spider.Figure(. 2)

1) Spider:pt_fingerprint_wp: The first spider is responsible for sending a request to the Google search engine, using specific "Google hacking" techniques to narrow down the results to Portuguese websites that use WordPress. These techniques include queries such as 'site:.pt inurl:wp-content', site:.pt inurl:wp-login.php','site:.pt inur:wpadmin', and 'site:.pt "Powered by WordPress". The spider receives the response from the request and extracts the domains of the found sites, storing the results for later use.

2) Spider:version: The second spider utilizes the results obtained by the first spider. It sends a request to each discovered domain and verifies if the site indeed uses WordPress. If the usage of WordPress is confirmed, the spider calls a specialized function to extract the specific version of WordPress in use. Additionally, the second spider collects other relevant information about the domain, such as IP address and associated name servers. All this information is stored in a database for further analysis. Below we can see the figure(2) illustrating the structure of the spider's code:

In this way, our web crawler is capable of searching and verifying Portuguese websites that use WordPress, extracting their specific versions, and collecting additional information about the domain. This process enables a more detailed analysis of the WordPress landscape in Portugal, providing valuable insights for researchers, developers, and professionals in the field.

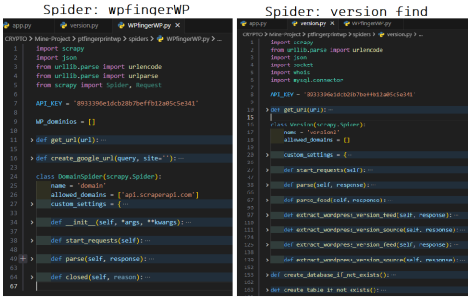

Fig. 2. Sourcecode: pt_fingerprint_wp

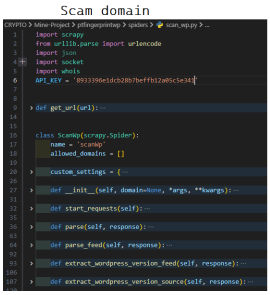

Fig. 3. Sourcecode, Find WordPress in website

B. Crawler:Find WordPress in website

Initially, our idea was to create a single crawler with the purpose of searching for Portuguese websites that use the WordPress platform. However, we realized that it would be interesting to implement a second crawler that performs a more focused and specific verification. In other words, a crawler that searches for a specific website, provided as a parameter, and determines whether it uses WordPress or not. Additionally, this second crawler would also check the WordPress version if the use of the platform is identified.

To develop this crawler, we designed a spider that sends a request to the given domain and analyzes the response and the site's source code. This way, it is possible to verify whether the website in question is using WordPress or not. Moreover, this verification is carried out meticulously, considering various aspects of the provided domain when the crawler is executed.

Once the use of WordPress is identified, the crawler extracts the specific version of WordPress being used on the website. Furthermore, the crawler returns relevant information about the domain, such as IP address, name servers, administrators, and other pertinent details. This provides a more comprehensive insight into the analyzed website. Below we can see the figure(3) illustrating the structure of the spider code:

Therefore, with this crawler, it is possible to obtain a detailed analysis of a specific website, checking whether it uses WordPress and providing additional information such as the version of WordPress used and relevant data about the domain.

C. UserFriendly Web Interface for Crawlers

In order to make the usage of our crawlers more convenient, we have developed a web interface using technologies such as HTML, CSS, JavaScript, and the Python Flask server. With this approach, we provide users with a user-friendly and intuitive interface for utilizing our crawlers.

The construction of this web interface was carefully planned to ensure an enhanced user experience. By providing the desired information, the interface interacts with the Flask server, which executes the crawlers in the background. Once the execution of the crawlers is completed, the results are sent back to the interface, where users can view them in a clear and organized manner.

Through this web interface, we offer users a simplified and pleasant platform for interacting with our crawlers. By utilizing this interface, they have the ability to submit the necessary information in an easy and intuitive way, as well as receive the results in an accessible format. The image(4) below illustrates the appearance of our web interface. In summary, our web interface represents a

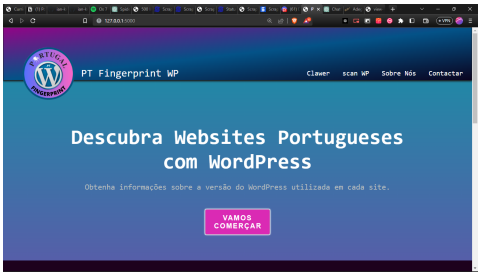

Fig. 4. Web Interface for Crawlers

significant step towards enhancing the user experience of our crawlers. It provides users with a pleasant and simplified environment where they can interact intuitively, submit their requests, and receive the results in a clear and organized manner.

D. Server "app"

To connect the web interface with the crawlers, we have developed a Python server using the Flask framework to provide a robust infrastructure.

The server has been structured to handle different HTTP requests. We have implemented specific routes and endpoints to execute the functionalities of the interface. It also incorporates error handling mechanisms to ensure a smooth user experience.

The server interacts with the crawlers by triggering their execution in the background. We have implemented security measures, such as input validation, to protect against potential threats. The server enables asynchronous interaction between the interface and the crawlers, playing a crucial role in their coordination.

The "app" server serves as the foundation of our web interface, delivering a reliable and efficient infrastructure. Below is an image(5) depicting the code structure of the server. In summary, our server acts as a bridge

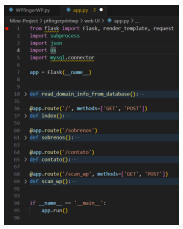

Fig. 5. App server

between the web interface and the crawlers, providing a robust infrastructure. It handles various HTTP requests, incorporates error handling mechanisms, and facilitates asynchronous interaction. The server plays a crucial role in coordinating the interface and the crawlers, ensuring a secure and efficient user experience.

IV. Crawler Results

A. Crawler for Finding Portuguese Websites that Use WordPress

We have implemented a specialized crawler within the graphical interface to search for Portuguese websites that utilize WordPress. Upon clicking the "Let's Get Started" button, the server initiates the execution of the crawler and subsequently sends the obtained results back to the user. An example output from this particular crawler is depicted below figure. 6:

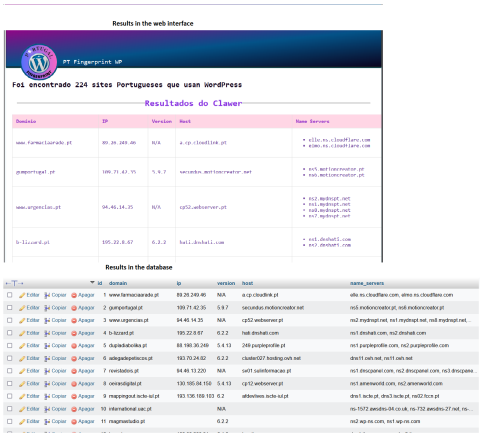

Fig. 6. Crawler for Finding Portuguese Websites

B. Crawler for Analyzing Website Usage of WordPress

Another crawler focuses on analyzing whether a website implements WordPress. When a user submits a request, the provided domain is forwarded to the server, which then proceeds to execute the crawler with the domain as a parameter. The server subsequently relays the results back to the user. An example of the output results is displayed below figure. 7:

Fig. 7. Crawler for Analyzing Website Usage of WordPress Results

These crawlers are diligently designed to furnish users with pertinent information, thereby assisting them in gaining insights into the websites they are researching. The web interface seamlessly facilitates users in accessing these functionalities and acquiring the desired results.

V. Conclusion

Developing a crawler to identify Portuguese websites built with WordPress and obtaining their versions, along with creating a web interface, was an exciting challenge for us. Initially, we had to deal with a completely unfamiliar programming language, Python, which posed some difficulties. However, we didn't let that discourage

us. We sought appropriate learning resources and dedicated ourselves to studying the syntax, control structures, and essential libraries of Python. With patience and perseverance, we gained a solid understanding of the language. When tackling the challenge of identifying Portuguese websites built with WordPress, we used specific dorks such as "site:.pt inurl:wp-content" or "site:.pt inurl:wp-login.php" to refine our searches and obtain relevant results. We learned web scraping techniques and how to extract important information from web pages, such as the versions of WordPress being used. Furthermore, we took it a step further and created a web interface for the application. This allowed us to interact with the crawler in a more user-friendly manner and visualize the results in a clear and organized way. We explored Python frameworks and libraries such as Flask to build a functional and intuitive interface. Throughout the process, we encountered difficulties and obstacles, but we didn't give up. With perseverance and the support of the Python developer community, we overcame the challenges and honed our skills. Continuous practice and the search for solutions have made us more confident and experienced programmers. In conclusion, the development of this crawler and web interface was a journey filled with challenges and learning. Dealing with an unfamiliar language required dedication and effort, but it also provided us with a sense of accomplishment when overcoming the difficulties. We are proud of the results achieved and look forward to applying our knowledge in future projects.

References

[1] DigitalOcean. "Como fazer crawling em uma página web com scrapy e python 3 (pt)." (), [Online]. Available: https://www.digitalocean.com/community/tutorials/como-fazer-crawling-em-uma-pagina-web-com-scrapy-e-python-3-pt.

[2] YouTube. "Web crawler fácil e básico em python." (), [Online]. Available: https://www.youtube.com/watch?v=NGBgAzuC1vc&t=2s&ab_channel=Matias.

[3] HackingVision. "Google dorks – find vulnerable wordpress sites." (), [Online]. Available: https://hackingvision.com/2017/07/21/google-dorks-find-vulnerable-wordpress-sites/.

[4] I. Kerins. "Google scraper - python scrapy." (), [Online]. Available: https://github.com/ian-kerins/google-scraper-python-scrapy.

VI. Authors

Fig. 8. Marco da Luz, Self-assessment - 18

Fig. 9. Romilson Monteiro,Self-assessment - 19.7

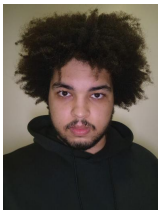

Fig. 10. Francisco Oliveira, Self-assessment -