National College of Ireland

Configuration Manual

MSc Research Project
MSCCYBETOP

Peter Byrne
Student ID: X19164131

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	PETER BYRNE……
Student ID:	X19164131……
Programme:	MSc. Cybersecurity…… **Year:** 2023……
Module:	MSc. Research Project……
Lecturer:	Raza Ul Mustafa……
Submission Due Date:	……
Project Title:	Configuration Manual……
Word Count:	1763…… **Page Count: 4**……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: PETER BYRNE……

Date: 23/06/2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Problem Statement

Peter Byrne
Student ID: X19164131

1 Problem Statement

Methodology for Automated Forensic Web Scraping of Pricing Information.

This research seeks to outline a forensic methodology for web scraping pricing data from websites using the Python language. Websites hosted on the world wide web provide an unparalleled opportunity for collection and analysis of real-time data (Diouf *et al.*, 2019). The process of collecting and analysing this data is known as *'web-scraping'*. Web-scraping broadly speaking consists of a two-software program process; collecting the data and analysing the data to extrapolate meaningful information (Farley and Pierotte, 2017). This paper evaluates existing methods and proposes a forensically sound methodology to scrape pricing information from chosen website sources using an MD5 (messaging digest 5) hashing algorithm to generate a value when the HTTP content of the page is acquired and stored. The MD5 hashing algorithm is a one-way deterministic mathematical algorithm producing a fixed length output, that is commonly used to verify integrity in computer science.

Automated approaches to web scraping are divided into three approaches; *supervised*, *unsupervised* and *hybrid* (Uzun, E., 2020). This methodology will use a supervised approach, in that certain data labels are required to parse the end results. The reason for this approach is because website architecture can change periodically, and as such code changes may be necessary (Khder, M.A., 2021).

2 State of The Art

From exploring research databases such as IEEE current state of the art papers seem to be focused on retrieval and collection of specific data from websites, rather than the forensic component of preservation of the page in question. Furthermore, proposed methodologies can also include use of third-party software or browser-extensions, which can be useful for limited scope and manual acquisition but are not always feasible for automated large scale forensics acquisition over a period (Diouf *et al.*, 2019). The paper *"Forensic Acquisition and Analysis of Webpage"* (Vidya, Saly and Balan, 2022), discusses a methodology for scraping all of the elements of a webpage and using a hashing algorithm to verify each page. This is a very powerful method for acquiring websites when looking for specific components or hyperlinks. The difference between the methodology discussed by V.Vidya et al, and my proposed methodology is that my methodology will only take the HTML code which is required for pricing data, and no other web page elements. The advantage of this is that the requirements for storage are less, however the methodology discussed by V.Vidya et al is a robust solution that could also be used, and can be advantageous in storing graphical elements of the page. There is also a modular component to my methodology which specifically allows for parsing of pricing information or other identified elements of the page.

Existing papers which I have consulted describe several processes of collecting data using available tools such as *'wget'*, or of parsing the page in real time in order to collect only the data that is required. The methodologies proposed include use of cloud-computing solutions (Khder, 2021). These approaches allow for a fast and scalable solution which can work to get a large amount of data. However, there are limitations to consider with these approaches. Such approaches to scraping and storing vast amounts of data due not have a built in data integrity check, and as such if the data was to be used in a capacity of being evidence in the legal system, there would be a weakness inherent that the data would either have to be taken on face value in good faith or an accepted risk to the integrity of the data being open to being changed or compromised. This would not be *'best-evidence'* going forward where there exists the possibility of presenting data which has been captured using a method of preserving integrity through hashing.

The main contribution of this paper is to evaluate existing methods in terms of time and efficacy, and to propose a full methodology to collect the data and offer it in a way that is forensically sound and presentable as evidence in court if needs be. The methodology I have developed in Python involves use of the MD5 hashing algorithm alongside a full download of the web page html code and a graphical screenshot. The second component of the methodology uses JSON data structures to configure the tool to parse the required data fields from each page.

This paper will compare these methods and highlight issues with current methodologies. The issues identified in my research so far is that many sites block HTTP access from known cloud IP address ranges and block some methods of making HTTP GET requests. So far, I have carried out tests using a controlled environment, using six (6) methods of scraping using various Python libraries which are suggested for use in state of the art papers (Persson, E., 2019).

3 Implementation and Evaluation

The Web scraping script itself is written using Python and will use the Selenium library for downloading and the Beautiful Soup library for parsing.

Test environments for these various methods are Ubuntu Server 22.04 LTS updated to the same version, and running on a local server test environment, Amazon AWS, Azure and Linode cloud environments. Each of the methods will have the same computational resources allocated to them. A limitation of the research is that there are extrinsic variables influencing performance on each cloud test environment, such as bandwidth, network infrastructure, ping response times etc. as well as the source websites that are subject to change and outside of control. For this reason, I will take a snapshot of the results at roughly the same time and present the findings. The real-time nature of these changes may result in different outcomes when taken at a different point in time, however although the results may be contemporaneous the results and the testing methodology should still be of value in the future.

To easily accomplish the testing in a scalable way, I have developed a test Python script to test each of the following Python library methods of downloading sites:

- URL Lib

- Session Requests
- Selenium
- HTTP Lib 2
- Wget
- Pyppeteer

This script measures the success of each method in each test environment as well as the time taken to complete each method. An issue that presented during testing, was that measuring the efficacy of the method using HTTP response codes was not always accurate. Libraries like Selenium, and Pyppeteer do not allow HTTP Response codes through their native library API and a HTTP response code of 200 (OK) could be returned by a third-party cyber security solution (such as Cloudflare, which attempts to block DDoS or similar attacks). Instead, the page was downloaded and if the size was above 1000 bytes then it was deemed to be a real web page.

The aim of this testing is to have a scientific basis to recommend my methodology over the others.

Once the testing phase and metrics are presented in the paper, the methodology of a forensic web scraper will be presented. The web scraper consists of a downloader and a parser. The downloader is responsible for acquiring web pages in a forensic way. It downloads the page and stores it in its own subfolder, along with a log file which stores a timestamp of the acquisition, an MD5 hash of the page content, and the method of acquisition used.

The final part of web scraper is the parser. This uses a JSON configuration file for each website, that tells the script what fields you are interested in scraping and outputs in a tab-separated CSV format. The methodology is designed so that any information obtained from the web scraper should be verifiable and adhere to the highest standards of digital forensics.

References

Diouf, R. *et al.* (2019) 'Web Scraping: State-of-the-Art and Areas of Application', in *2019 IEEE International Conference on Big Data (Big Data)*. *2019 IEEE International Conference on Big Data (Big Data)*, pp. 6040–6042. Available at: https://doi.org/10.1109/BigData47090.2019.9005594.

Khder, M. (2021) 'Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application', *International Journal of Advances in Soft Computing and its Applications*, 13(3), pp. 145–168. Available at: https://doi.org/10.15849/IJASCA.211128.11.

Milev, P. (2017) 'Conceptual Approach for Development of Web Scraping Application for Tracking Information', *Economic Alternatives*, (3), pp. 475–485.

Persson, E. (2019) *Evaluating tools and techniques for web scraping*. Available at: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-271206 (Accessed: 7 June 2023).

Upadhyay, S. *et al.* (2017) 'Articulating the construction of a web scraper for massive data extraction', in *2017 Second International Conference on Electrical, Computer and*

Communication Technologies (ICECCT). *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–4. Available at: https://doi.org/10.1109/ICECCT.2017.8117827.

Uzun, E. (2020) 'A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages', *IEEE Access*, 8, pp. 61726–61740. Available at: https://doi.org/10.1109/ACCESS.2020.2984503.

Vidya, V., Saly, K. and Balan, C. (2022) 'Forensic Acquisition and Analysis of Webpage', in *2022 2nd International Conference on Intelligent Technologies (CONIT)*. *2022 2nd International Conference on Intelligent Technologies (CONIT)*, pp. 1–6. Available at: https://doi.org/10.1109/CONIT55038.2022.9848303.