Prevent XPath and CSS Based Scrapers by Using Markup Randomizer

Ahmed Diab, Tawfiq Barhoum
Islamic University of Gaza, Gaza, Gaza Strip, Palestine

Abstract *Web Scraping may consider as data theft action, several researchers have introduced some approaches for addressing this issue. These solutions could solve the problem in partial ways and sometimes, solution cannot be applicable with modern web techniques. Consequently, in our work we have introduced a new approach for stopping web scraping in an efficient way and applicable with modern web techniques called Markup Randomizer, which changes the HTML and CSS in proper way randomly in timely manner. The best feature of our model is that each web page can use it without paying any efforts or restrictions in web site markup. Experiments done over collected dataset which consist of 30 websites divided into three categories: News, Currency Rates and Weather. The proposed model based on Markup Randomizer applied over this dataset. The aim of the experimental is to measure the Similarity, File Size and the time. During testing the proposed model, we get that a change on the markup done up to 50%, file size is changed and optimized after during the process. The required time to applying the model and generating the new markup is good and up to 2 minutes. Finally, we find that our proposed markup randomizer is accepted.*

Keywords*: Anti-Scraper, Anti-Data Theft, Web Scrapers*

Received August 1, 2018; Accepted September 5, 2018

1. Introduction

Web scraping is the process of extracting the information of the web pages, this process simulates human attitude when he opens the website but it differs that it automated process done using HTTP protocol or by embedding a web browser. Web Scraping is a process like web index "search engine function" which indexes websites information using its bots. In contrast, web scraping extract specific information is related to the webpage itself but in case of search engine they take only meta tags if exists from the website [7, 16].

Due to the richness of webpages information and the increasing of need of data exchanging among the web in automated fashion, the first web scraper has developed and has inspired from search engine bot functionality.

Web scraping tools can be used in ethical and unethical way, firstly when it is used for research purposes and without taking over the privacy and copyright and the other when some people take content from some websites and repost the content on their websites particularly when the content is unique and creative.

Web Scraping is very useful technique helps researchers in many fields to improve their data and knowledge, one of the most practical fields is for weather forecasting they used the scrapers to get historical data about the weather [2].

Another usage of the web scrapers [7] is for the new Startups because of the lack of time, the need of data and the limitations of resources they do prefer to use the web scraper to scrape data from similar websites initially then they can update the scraped data whenever they need to. This is not fair for content owners who have the ownership right of the data itself such as innovative content and patents. By the time, this issue caused many losses for them in multiple fields as Data Theft, Intellectual Theft, and Economic Lose. So this type of unauthorized usage may have classified as Data Theft (is the act of stealing computer-based information from an unknowing victim with the intent of compromising privacy or obtaining confidential information) which is harmful unethical problem with destructive effects for the companies.

As a result, web scraping becomes a curial problem need to be solved and so far, to my knowledge there are few solutions proposed to solve this problem. Researchers [14] have introduced an invention for preventing scraping by using a filter that reproduces the data requested by the client in an unstructured manner, which could be understood by browsers, but a robot with scraping software can't deal with it in order to get the desired data. Other researchers [5] have introduce a compound solution based on filtering the visit to three categories (Black-List, Gray-List, White-List) and then treat with the visitor depends on his category. Gray-List contains the suspicious visitors, which are subjected to several techniques to decide whether block or not.

Other solutions [10-12] was provided as commercial tools by developers but they hide all the technical information and offer it without any documentations.

In our proposed solution, we will prevent CSS and XPath scrapers because the most of scrapers based on CSS and XPath technique in extracting data from websites. This can be done by using the markup randomizing which will change the HTML and CSS files automatically in a timely manner to be different in markup and the same in the result. Therefore, the scraper rules will be meaningless because it treats the webpage, as a new webpage and the scraper should take an action to update the rules at each time they access the webpage. Because of this technique, the scraper will stop functioning well and stop to scrape these pages.

2. Background

2.1. Web Scraping

After defining the Web Scraping term, Web scraping process contains three main processes (Web Crawling, Web Scraping, Saving the data) as shown below in Figure 1. The first part of the scraping process is Web crawling, which means the process of navigating the webpages, and finding the links in the web page, this will enable you to reach each page as well as links recursively in the page. The second process is the heart of the web scrapers, which means to extract the data from the web page by using predefined rules based on one of the following techniques (XPath & CSS, Regular expression or semantic rules). Finally, the data saving is the extracted data on file or database.

Figure 1. Web Scraper Architecture.

2.2. Web Scraping Techniques

There are a lot of web scraping techniques are used by the scrappers around the world and classified into few categories by the behavior of the scrapper and the anatomy of the data as the following:

2.2.1. Web Usage Mining

The term “Web usage Mining” refers to the automatic discovery and analysis of patterns in clickstream and associated data collected or generated as a result of user interactions with Web resources on one or more Web sites.

They show that we can extract the data for web usage using web server log and show how much knowledge we can get if we analyze the data by using a specific software such as "Nihuo Web Log Analyzer". We can take a deep view of the visitor attitude and here is some reports from the analyzer.

2.2.2. Web Scraping

Converting unstructured information into structured information and stored into central database/spreadsheet. This can be done and specified by using one of the scrapers of embed a browser into application and then define the criteria and targets for extracting and grapping.

2.2.3. Semantic Annotations

Annotations or Meta data used to locate data within the document, so we can prepare a list of semantic data and define a layer for the web scraper before scraping data.

Another technique was very common in most papers and implement in most scraping tools which is DOM based manipulation and accessing data by XPath and CSS because it’s the easiest and simplest technique and supported by most programing languages and treated like the XML processing. Because of that, they encouraged to build their scrapers on those techniques and proposed their approach of scraping data on the bases of DOM manipulation and different in the architecture of the methodology, programing language or even used tools.

2.2.4. The Custom Scraper

Python based scraper consists of three parts of the process, first part is web crawler, the second is data extractor, and the last is the storing method. They have built the scraper with new concepts to full-fill the new startup need as they need very much data but with no time to collect, so they need an efficient and speed tool.

Web Crawler is a tool or a set of tools that iteratively and automatically downloading webpages also extracting URLs from their HTML and fetching them recursively [13]. So we need to have a list of URLs to be visited ,this list will be called as a seed [10], each page will be visited and also all links inside each particular page will be extracted to the list "seed" again to be visited, most of Web Crawlers contains the following parts:

1. Downloader: the process to download the pages.
2. Queue: contains the list of URL to download.
3. Scheduler: is the process to start and organize the downloader.
4. Storage: is the process to extract the Meta data of the web page and save it as well the text of the web page.

Data Extractor The process of extracting information from a single web page, although we have a lot of uses we will focus to extract specific data or predefined

rules. They achieve this goal by selecting the data using CSS Selectors or XPath patterns.

Exporting to CSV After we have crawled the pages and extracted the data, we will have a list of extracted information stored in memory, we just need to save them to CSV using Python API.

1. Scrapple
   Scrapple a Flexible Framework to Develop Semi-Automatic Web Scraper, the main purpose and contribution of Scrapple is to reduce the required modifications on the scripts to run the scraper like Scrapy.
2. Extracting Entity Data from Deep Web Precisely Researchers have proposed a model for web data extracting, this model consists of many modules:

a. Web Crawler: they have proposed an intelligent web crawler that can deep dive into the website and talk the navigation links form the static web pages as well as dynamic.
b. Pretreatment of web resources: they have developed two procedures before processing the webpages the first is to normalize the html page and the other is to eliminate the noisy information.
c. Locate and extract the entity data from Deep Web accurately: the concept of data extraction from unstructured data to structured done by DOM interface, then parsing the document using JTidy the web page is transformed to DOM tree to access each node of the webpage as an object.

2.2.5. XQUERY Wrapper

Researchers [18] have proposed a system to extract from websites, this approach was based on XQuery. Wikipedia Says : "XQuery (XML Query) is a query and functional programming language that queries and transforms collections of structured and unstructured data, usually in the form of XML, text and with vendor-specific extensions for other data formats (JSON, binary, etc.)" [15].

They have proposed a schema model for modeling both web data and user requirement; therefore, they handle all type of data (single and complex data). Figure 2 show the structure of the data in a website that emphasizes the hierarchical nature of the data.

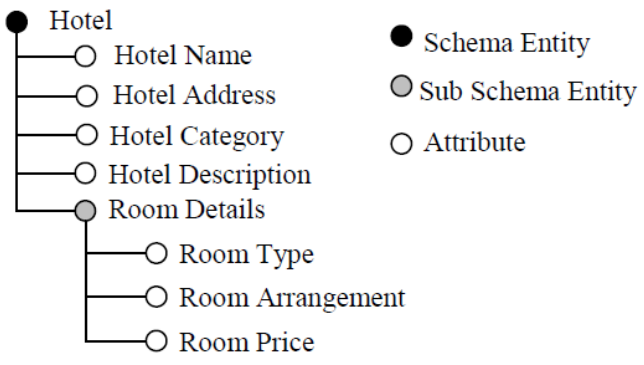

Figure 2. Proposed schema model (Nie et al., 2011).

This example of the proposed models previews the hierarchical data of the website and differentiate between the type of nodes we have (single and complex).

The annotating of data semantics they map each data value to an attribute, and then they used an exclusive path to annotate the location of the node in DOM tree. The path will be XQuery expression which is based on XPath so it will be like the following formula

$$P = /T1[p1]/\ T2[p2]/...../\ Tm[pm]/$$

3. Related Work

Many efforts addressed to mitigate and stop the Web Scraping, these efforts have classified into the following categories Legal, Developers and research efforts:

3.1. Legal Efforts

Many legal efforts lead to introduce many laws such as (Mitchell, 2015) proposed Copyright Law which stands for "Copyright is a legal right created by the law of a country that grants the creator of original work exclusive rights for its use and distribution. This is usually limited by time. The exclusive rights are not absolute but limited by limitations and exceptions to copyright law, including fair use" (Mitchell, 2015), also The digital millennium copyright act (DMCA) was proposed by [1] it implements two 1996 treaties of the World Intellectual Property Organization (WIPO) and it criminalizes all ways intended to circumvent measures that control access to copyrighted works [17]. After that, The United States Law on The Restatement (Second) of Torts § 217 [6] defines the Trespass to chattels as"Intentionally dispossessing another of the chattel, or using or intermeddling with a chattel in the possession of another".

Unfortunately, all these laws are not covered in all cases and will not force the scrapper to be down and it will be circumventing of the law due to the following reasons:

A Statistics and facts: if you publish a fact on your website about something is copyrighted it will be much fine.
B Information about copyrighted content posting frequency over the time is fine also.
C If the creative content is shared in a verbatim may not be violating copyright law if the data is prices, names, company executives or some factual piece of information.
D If you just want to store the materials into your offline database, you will be fine.
E It is fine also if you analyze that database and publishing the statistics, authors data or even meta-analysis data.

F When you select a few quotes or brief samples to your meta-analysis to make your point you should examine that "fair use".
G Lack of consent criteria of Trespass to chattels is not enough for scrappers because they treat the webpage the same as web browsers.
H Actual harm criteria of Trespass to chattels also does not apply because the scrappers are one hundred percent like the web browser so the actual damage of scraper visit is the same as browsers.

3.1. Developer Efforts

There are few business solutions developed by large companies that partially closed the gap and proposed some business products such as the following:

1. ShieldSquare is a software service that provides a Real-Time anti scraping service [8] that contains the following features :
    a. Actively detect/prevent website scraping & screen scraping.
    b. Prevent price scraping bots from competitors.
    c. Enhance your website's user experience.
    d. Get complete visibility into bot traffic on your website.
    e. See comprehensive insights on BOT types and their sources
2. ScrapeDefender is a tool to stop the web scrapers with main three functions Scan, Protect and Monitor detailed into the following points:
    a. Scan: ScrapeDefender routinely scans your site for web scraping vulnerabilities, alert you about what we find and recommend solutions.
    b. Secure: ScrapeDefender provides bullet-proof protection that stops web scrapers dead in their tracks. Your content is locked down and secured.
    c. Monitor: ScrapeDefender provides smart monitoring using intrusion detection techniques and alerts you about suspicious scraping activity when it occurs.
3. ScrapeSentry first anti-scraping solution is developed to protect sites by blocking scrapers from violating intellectual property with the ability to distinguish the good and bad scrapers whether human or bot. ScrapeSentry is a software as a service (SaaS) anti scraping service 24/7 delivered from the Sentor Security Operations Centre (SOC). These Services include monitoring, analysis, investigation, blocking policy development, enforcement, and support. Recently Distil Network acquires ScrapeSentry on January 13, 2016 [3].
4. Distil Networks is the largest and modernist bot detection and mitigation for stopping all types of bots [4]. Network blocks every Open Web Application Security Project (OWASP) automated threats such as Web Scraping, Denial of Service or even Skewing by BOT defense product they own its very excellent product because it's the first product that covers webpages, API and Mobile Apps which is distinct service.

The proposed solutions are too good but the gap still exists and it has extra efforts to eliminate the scrappers totally and here is the following points for each proposed system:

1. *ShieldSquare* is very attractive and intelligent. Due to the fast upgrade and update in the scrapers techniques this software will not survive all the time to detect the new patterns of the scrapers then the scrapers will eliminate the barriers and avoid the detection and catch techniques. Because of that, the proposed solution may mitigate the number of bots, but never helps the websites to be safe from the bots. On the other hand, this proposed solution requires each webpage or mobile app page to check if the visitor is real or bot, which means lack of performance. So we still need a paradigm to protect the whole website on the level of web server that never needs an interaction from the developers to be assured that each request will be handled without exceptions.
2. *ScrapeDefender* is a great solution, which will prevent all known scrapers by the firewall and make the content safe. But if we have a new era of web scrapers which means new attitude and patterns the firewall therefore will not prevent those scrapers. On the other hand, if the attackers exploit the DDOS and target the website then the firewall will stop then the website will be either stopped or the scrapers will continue work alone. As a result, the scrapers will access the gems and take the control over the website.
3. *ScrapeSentry* is very intelligent and excellent, and has great reviews from its clients as they list on their website. Like other solutions, they filter the request and then take an action according to the analysis of the request, so we still have the same problem that if we have a new bot with new footprints the system will be blinded and never detects it, until the security officers fix it. Another weak point also like the other is that they add a new layer for the request life cycle, which will filter the request, let us say if we face a DDOS attack to let the layer down then the scraper will scrape everything until the layer returns back. Therefore, we still need any solution based on the markup itself, which will let the scraper stop without any affect to the performance.
4. *Distil Networks* proposed a direct bot detection and mitigation process we find that they depend on how to prevent the bot to reach the web server as whole, but they never have any plan for some cases, in case that bot successfully reached the page and stole the content so it still not sufficient and not

dependable so they add the term 'Mitigation' for their proposed technology.

3.2. Researches Efforts

There are relatively few works for addressing Web Scraping issue and here we are going to discuss some of the most related works to our work.

Researchers [14] have presented an invention for preventing the scrapping of the information content of a database used for providing a website with data information. Their invention depends on using an anti-scrapping filter or filtering means. The filter is used to perform some processing on the data requested by clients before being sent to them, in order to prevent scrapping. The method of preventing the information scrapping comprises of the following steps:

1. Receiving the requested structured data record from the database.
2. Splitting all the elements or the fields of the data into data containers, called cells, in a predetermined way.
3. Giving each cell a unique sort-id, which is generated by a random number generator, and location information, which determine the location of the cell is inside the web page.
4. The cells are sorted by the sort-id to establish a new unstructured data, to be sent to the requesting client.
5. Each cell is encoded into a markup language, e.g. HTML.
6. The resulting file is delivered to the requesting client.

As a result of sorting the data containers into unstructured manner, a robot with scraping software would not be able to interpret the content, because it can deal only with structured data. On the other hand, the unstructured placement of the data containers or cells would not cause any problem for the displaying of the file as a web page. The web browser will ignore the cells structural placement in the code, which is based upon the sort-id, and will visually sort the data according to the location information. Thus, the scraping robot will be prohibited to use a file that is generated by the proposed filter.

This paper proposed a good solution because it solves a part of the problem this part is XPath based scrapers, but it is not efficient today because when we reorder the html tags within the pages the style of the page will corrupted as it randomly ordered. Another problem is HTML5/CSS3 based websites build in a way that cannot be reordered because the stylesheet is identical to the elements in html file. The proposed solution cannot deal with CSS based scrapers and the scraper will still function well because the class is not changed the change only in the order so the scraper will access the data in despite of the layout. Last weakness point is the paper never talks about the performance issues and caching for the files, so the performance of the system will be very bad and will not help the website owners.

Two researchers [5] have proposed a new model to mitigate the web scrapers based on historical analysis for the visits. They have created three lists for the visitor's IP address (Black-list, Gray-list, White-list) and deal with the visitor depending on his class. In the case of Black list, the model will block the visit and deny the session from initiation.

In white list the session will be initiated successfully without any barriers then if the visit was classified as gray listed visit the model will treat with it in may suggested solutions as listed below:

1. The model may display captcha before he views the content.
2. The model may identify the scraper through browser information that is usually not sent to browser.
3. The model may change the markup randomly to stop scraper from getting data using old CSS and XPath selectors.
4. The model may change the information to an image so that the scraper will not reach any valuable text.
5. The model may produce a frequency analysis to check if the visits number is normal or abnormal.
6. The mode may produce an interval analysis to check that if the interval analysis is similar it may be classified as gray list and to be redirected to bot-differentiating techniques like Captcha. Therefore, it may be efficient if it is used in long-term strategy.
7. The mode may produce a traffic analysis this is very necessary in these days because the modern scrapers have many IP address by these techniques they can detect those scrapers.
8. The mode may produce a URL Analysis for the visited pages to check if the ratio between data-rich pages and non-rich, so that they can identify the scrapers.
9. The model may use Honeypots and Honeynets, which is very common in networking companies like Amazon and CloudFlare.

The proposed solution is really very good as it provided a multi-tier defense, on the other hand it is not enough because the scraper may be developed and always treated as white listed so we need to focus more on the content itself. If we focus on the markup randomizer they proposed it could stop only the CSS based selectors, but if the scraper was used XPath it will not mitigated and the scraper will behave and function well. Another weakness point is not suggested idea provided to cache the generated randomized HTML markup, which means the model will generate a new randomized html file each time it accessed which will cause a harmful load on the server as well as if we have many sessions the server will get down. So the possibility of Distributed denial-of-service (DDoS) [9] will increase which is not acceptable in any way.

4. Methodology

We have proposed a model based on Markup Randomization to prevent the scrapers. The model is meant to change the markup in a way that the scraper will be prevented permanently. Next sections we will discuss the details.

4.1. Markup Randomizer Model

In this section, we are presenting the proposed model which illustrated at Figure 3.

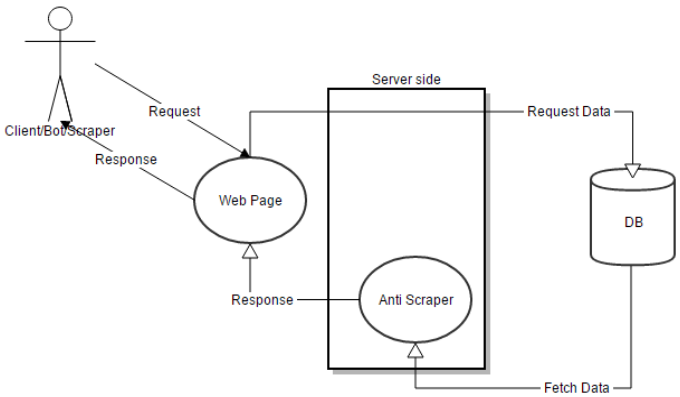

Figure 3: Deployment Model for the Markup Randomizer.

Markup Randomization is a technique to protect from XPath, CSS based web scraper, and it is a very simple and efficient model.
As we see in the model above we have the following:

The Actor: the actor, which may be Client or browser or search engine or even scraper.

- Web page: the requested web page.
- Server Side: The web server itself and its role is to get the data from database then send it to anti scraper module.
- Anti-Scraper: The module at web server before sending the response to client. Its main role is to prepare randomized version of the web page.

Our markup randomizer designed to stop most of scrapers work because we will make a crucial change on the web page markup. To clear describing the changes let us spilt the scrapers to CSS based scrapers and XPath based scrapers.

4.1.1. CSS-Based Scrapers

This type of scrapers is designed to extract the data from a webpage using CSS selectors for example, we want to extract two elements values from a webpage and the original markup is:

```
<div class="title">Data</div>
<div class="news_details">Data</div>
```

Therefore, the scraper should write the following code to extract those fields.

```
$('.title') .text();
$('. news_details).text();
```

This code will return the value of the fields then to be stored in the database. The problem is the CSS class of each field is never changed Therefore the scraper will reach the data whenever tried to access the page.

In our model the page markup as well as CSS will be changed automatically in a timely manner so, when the scraper setup the configuration to extract field by CSS classes he will figure out that scraper is stopped working and never returning data. Because of the automatic change and this is the expected result of our system to see the CSS code snippet before and after the change at figure 7 and 8.

This change in CSS required a necessary change in HTML to fit with the new CSS Rules, so we have created a dictionary file that contains the mapping between the old rule name and the new rule name and store the file temporarily to the disk. An example of Randomize HTML file as well as the original are shown below at figures 9 and 10.

4.1.2. XPath-Based Scrapers

Another type of scrapers is designed to extract the data from a web page using XPath notation selectors for example, we want to extract elements values from a web page and the original markup is:

```
<html>
        <body>
        <h1>Data</h1>
                <table>
                <tr><td>Data</td></tr>

<tr><td>Data</td></tr>

                <tr><td>Data</td></tr>
                </table>
                </body>
                </html>
```

Therefore, the scraper should write the following code to extract those fields

```
$('/html/body/h1') .text();
$('/html/body/table/tr/td') .text();
```

This code will also return the value of the H1 element as well as each td element.

To stop the scraper, we will add empty invisible tags that will be inserted into the randomized html file, because of the nature of XPath is to access any element in the document the XPath should be represented the place of the element in hierarchical format. In our example H1 element is existed on the body element so if we have wrap the H1 within DIV tag the old XPath will be meaningless and to be updated to be like this:

/html/body/div/h1.

After we generated a randomized markup we save to the disk the following files:

- Randomized CSS.
- Randomized HTML.
- Mapping File.

This will enhance the performance for the model. Cron Jobs is the ideal solution to repeat the randomized process for each webpage on the web site to ensure that the markup is unique and refreshed all the time. The following steps are executed by each run of the Cron Job.

- Delete the old cached version of the Randomized CSS, HTML and Mapping File.
- Generate the new Randomized files.

5. Applying the Model

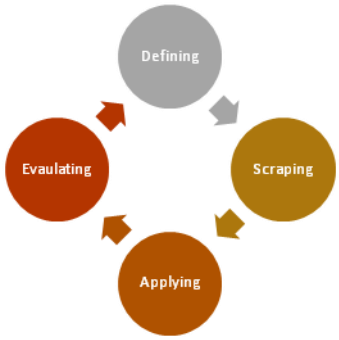

Figure 4. methodology steps breakdown.

5.1. Defining

First step is to define our websites and building our dataset that we have talked about Experiments and Results section.

We have created an offline version of each website contains all files we need like *Html, CSS and JAVASCRIPT* files.

5.2. Scraping

We have applied the scrapping for each website manually and defined the CSS-Rules for the scrapper and then ran the scraper to extract the data. We have successfully applied the scraper for each website with specific rules and get website data and stored it into a sheet shown in figure 6.

5.3. Applying the Model

We have applied our Markup randomizer for each website and created another offline version for the website that presents the enhanced page and it should stop the scraper with that rules. The following figures 7 and 8 show the a CSS code snippet before and after applying our proposed model and figures 9 and 10 show the HTML code snippet before and after applying our proposed model.

USD	GBP	EUR	JPY100	CHF	AUD	CAD	SGD	HKD100
0.0172	0.0973	2.8479	0.2956	6.0163	106.3285	3.5261	3.7602	0.2215
0.0171	0.0971	2.8449	0.2921	6.0116	106.2648	3.5242	3.7572	0.2214
3.908	5.4319	4.8193	3.6678	4.1252	3.0781	3.0431	2.9764	49.8504
12.4396	7.5431	13.3574	0.3619	0.0285	104.4169	5.6531	0.6163	2.9542
3.8965	5.4489	4.8334	3.6599	4.1318	3.0683	3.0097	2.9742	49.6907
12.4968	7.4884	13.32	0.3659	0.0284	103.897	5.653	0.6169	2.9721
0.0172	0.097	2.8618	0.2917	6.0146	106.4146	3.5288	3.7591	0.2221
0.0171	0.0967	2.8598	0.2912	5.9919	106.0864	3.518	3.7448	0.221
12.4132	7.5053	13.3149	0.3605	0.0284	103.9636	5.6536	0.6157	2.9487
3.9145	5.4042	4.8187	3.6682	4.1119	3.0457	3.0336	2.9687	49.9235
3.899	5.3781	4.8063	3.6938	4.1654	3.0215	3.0238	2.9556	49.7782
12.4745	7.512	13.3601	0.3665	0.0284	104.3714	5.6877	0.6171	2.9612
3.903	5.4222	4.8471	3.6927	4.1643	3.0459	3.019	2.9655	49.8207
0.0172	0.0971	2.8451	0.293	6.0154	106.5779	3.5341	3.7596	0.2222
0.0173	0.0975	2.8313	0.2938	6.0293	107.0077	3.5485	3.7683	0.2226
0.0172	0.0974	2.8465	0.2927	6.0034	106.6144	3.5356	3.7553	0.222
3.9305	5.4076	4.7936	3.6827	4.159	3.0395	3.0626	2.9661	50.2143
12.4756	7.5099	13.3489	0.3656	0.0284	104.1342	5.6825	0.6173	2.9544

Figure 5. Snippet from a scraped website.

```css
.WWEkHFXQvzyoaeGk {
    background: url("../images/social/flickr.png") no-repeat 50%;
}

.PHwlOHXnqQOYhskZ {
    margin-right: -15px;
}

.PHwlOHXnqQOYhskZ img {
    display: block;
    width: 100%;
}

.PHwlOHXnqQOYhskZ a {
    float: left;
    width: 57px;
    height: 57px;
    margin-right: 10px;
    margin-bottom: 10px;
    border: 5px solid #e8e8e8;
    -webkit-transition: all 200ms ease-in-out;
    -moz-transition: all 200ms ease-in-out;
    -o-transition: all 200ms ease-in-out;
    -ms-transition: all 200ms ease-in-out;
    transition: all 200ms ease-in-out;
    border-radius: 2px;
}
```

Figure 6. CSS code before applying the model.

```css
.WWEkHFXQvzyoaeGk {
    background: url("../images/social/flickr.png") no-repeat 50%;
}

.PHwlOHXnqQOYhskZ {
    margin-right: -15px;
}

.PHwlOHXnqQOYhskZ img {
    display: block;
    width: 100%;
}

.PHwlOHXnqQOYhskZ a {
    float: left;
    width: 57px;
    height: 57px;
    margin-right: 10px;
    margin-bottom: 10px;
    border: 5px solid #e8e8e8;
    -webkit-transition: all 200ms ease-in-out;
    -moz-transition: all 200ms ease-in-out;
    -o-transition: all 200ms ease-in-out;
    -ms-transition: all 200ms ease-in-out;
    transition: all 200ms ease-in-out;
    border-radius: 2px;
}
```

Figure 7. CSS code after applying the model.

```html
<div class="header-left">
    <div class="header-logo">
        <a href="index.php" class="otanimation" data-anim-object=".header-logo a.otanimation img, .header-logo a.otanimation h1" data-anim-in="flipOutX" data-anim-out="bounceIn">
            <img src="bnm2016/images/logo.png" alt="" /></a>
        <strong data-anim-in="fadeOutUpBig" data-anim-out="bounceIn"><i class="fa fa-home"></i> Homepage</strong>
    </div>
    <div class="header-socials">
        <a href="https://www.facebook.com/bnm.official"><i class="fa fa-facebook"></i></a>
        <a href="https://twitter.com/BNM_official"><i class="fa fa-twitter"></i></a>
        <a href="https://www.instagram.com/BankNegaraMalaysia/"><i class="fa fa-instagram"></i></a>
        <a href="https://www.youtube.com/user/bnm0fficial"><i class="fa fa-youtube"></i></a>
        <a href="index.php?ch=en_rss&lang=en"><i class="fa fa-rss"></i></a>
        <a href="https://www.pinterest.com/bnmofficial/"><i class="fa fa-pinterest"></i></a>
        <a href="index.php?ch=emailalert&pg=emailalert_subscribers&lang=en"><i class="fa fa-envelope"></i></a>
    </div>
</div>
```

Figure 8. HTML code snippet before applying our model.

```
<div class="vuxDYtGdNlfHcFzq "><a href="index.php" class="otanimation"
                                    data-anim-object=".header-logo a.otanimation img, .header-logo a.otanimation h1"
                                    data-anim-in="flipOutX" data-anim-out="bounceIn"><img
                src="bnm2016/images/logo.png" alt=""/></a> <strong data-anim-in="fadeOutUpBig"
                                                                   data-anim-out="bounceIn"><i
                class="YbsFe_EaqQ_SMyBv cIIlKSYHTAMXBtEn "></i> Homepage</strong></div>
        <div class="zwyqQLbKnVhqodaM "><a href="https://www.facebook.com/bnm.official"><i
                class="YbsFe_EaqQ_SMyBv yqiKwHiuzGEMBUhN "></i></a> <a
                href="https://twitter.com/BNM_official"><i class="YbsFe_EaqQ_SMyBv AbrgdqGRQrwOQMpQ "></i></a>
            <a href="https://www.instagram.com/BankNegaraMalaysia/"><i
                    class="YbsFe_EaqQ_SMyBv okeNYxNgPfaQhOBE "></i></a> <a
                    href="https://www.youtube.com/user/bnm0fficial"><i
                    class="YbsFe_EaqQ_SMyBv hVVqnVMaKbgpGHCk "></i></a> <a href="index.php?ch=en_rss&lang=en"><i
                    class="YbsFe_EaqQ_SMyBv ZqchTOz_lwZwSGbJ "></i></a> <a
                    href="https://www.pinterest.com/bnmofficial/"><i
                    class="YbsFe_EaqQ_SMyBv GsqhCHxSyQJgpaG_ "></i></a> <a
                    href="index.php?ch=emailalert&pg=emailalert_subscribers&lang=en"><i
                    class="YbsFe_EaqQ_SMyBv xwnVFTDVPpXQImWV "></i></a></div>
    </div>
```

Figure 9. HTML code snippet after applying our model.

5.4. Evaluating

To evaluate our model, we have to do the following steps:

1. Re run the scrapper to check if the scrapper is still alive or stopped totally.
2. Test the Markup by using third party to check if it is really changed and how much the percentage of the changed lines.
3. Test the file size to see if there is a change on file size per website and to detect the overall change ratio.

6. Experements and Results

To test our approach, we have experimented with our dataset which contains multiple categories and many websites.

We have built our dataset consists of websites from multiple categories as below:

1. News websites.
2. Weather forecasting websites.
3. Stock markets and currency websites.

We have chosen 30 websites randomly distributed 10 websites for each category and we intentionally checked that each website is different from the other.

A. Markup Similarity

We figure out the similarity between the page's markup for each website before applying randomizer and after applying randomizer as Table 1

Table 1. Markup similarity per each category.

Category	Similarity
News	37.75%
Currency	50.7%
Weather	34.85%
Overall	41.1%

As shown in table 1 we knew that we are changing only the CSS attribute for each page body's elements therefore the similarity then we will see there are Inverse Relationships between the page markup length and the similarity.

B. File Size

File size measurement means how much the generated files weight such as CSS and HTML. Measuring the file size is important because we will add new files to be stored on the server cache directory so it should be accepted, because it is a resource and each resource is limited on the world.

File size is changed after applying it because of length of the Randomized CSS Rule is 13 character per rule so this will reflect to the file size.

We figure that, file size is enhanced by our approach because of CSS and HTML optimization done during applying our model. The optimization done by removing unnecessary lines and comments as well as white-spaces.

C. Time

Time is the most important factor which led the performance and the model reliability and because the model is applied on timely-manner we have to be sure that it should be fast enough.

During the experiments, we see the time to apply the model on a specific page is good and it ranged from (1 second - to 2 minutes) which is efficient to stop the scraper because he needs a lot of time to setup the scrapper rules before starting scraping a targeted website.

Depending on the html markup length which is usually less than 5000. Figure 11 demonstrates the results for the experiments X-axis holds the total lines (CSS lines and HTML lines) and Y-axis holds the total seconds for applying the model.

Finally, we have got sure that system is running perfectly without any problems.

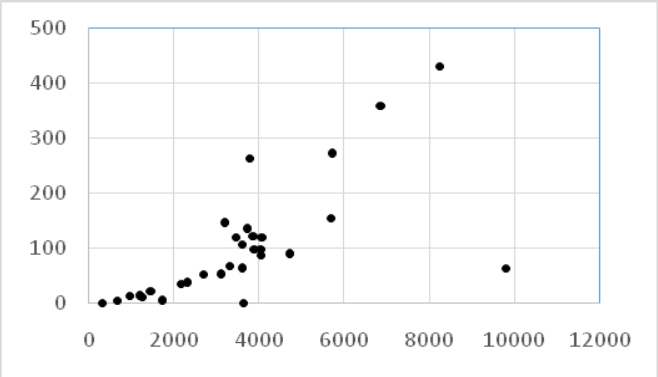

Figure 10. Experiments time.

7. Conclusion

Markup Randomizer Model stops the scrappers in the meantime by changing the CSS and HTML markups periodically which will efficiently stop the scrapers forever. Therefore, the model is tested on our dataset which was collected randomly from different categories. The test was done in three stages *scrape*, *randomize*, *re-scrape* and the results were very good. *Finally*, we got that our model is completely stopping the CSS-based scrappers and experiments show that the required time is very good for that bunch of results and it's accepted at all.

References

[1] Band J., "The digital millennium copyright act" , Available at: https://www.arl.org/storage/documents/publications/band-dmca-memo-16aug01.pdf, (accessed: 28 October, 1998), 1998.

[2] Bonifacio C., Barchyn T., Hugenholtz C., Kienzle S., "CCDST: A free Canadian climate data scraping tool", *Computers & Geosciences*, vol. 75, pp. 13-16, 2015.

[3] Distil Networks, *Distil Networks Acquires Sentor ScrapeSentry to Add 24/7 Security Operations Center and Expert Team of Analysts*. Available at: https://resources.distilnetworks.com/press-releases/distil-networks-acquires-sentor-scrapesentry-to-add-24-7-security-operations-center-and-expert-team-of-analysts, 2016.

[4] Distil Networks, *Distil Networks*. Available at: https://www.crunchbase.com/organization/distil, 2018.

[5] Haque A., Singh S., "Anti-scraping application development", in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 869-874, 2015.

[6] Henderson J., Twerski A., "Proposed Revision of Section 402A of the Restatement (Second) of Torts", *Cornell Law. Review*, vol. 77, no.6, p. 1512-1557, 1992.

[7] Mahto D., Singh L., "A dive into Web Scraper world", *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 689-693, 2016.

[8] Mathew A., Balakrishnan H., Palani S., "Scrapple: a Flexible Framework to Develop Semi-Automatic Web Scrapers", *International Review on Computers and Software (IRECOS)*, vol. 10, no. 5, pp. 475-480, 2015.

[9] Mirkovic J., Reiher P., "A taxonomy of DDoS attack and DDoS defense mechanisms" , *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39-53, 2004.

[10] ScrapeDefender, *ScrapeDefender*. Available at: http://scrapedefender.com

[11] ScrapeSentry, *ScrapeSentry*.

[12] ShieldSquare, *ShieldSquare Bot Mitigation and Bot Management solution*, Available at: https://www.shieldsquare.com.

[13] Thelwall M., "A web crawler design for data mining", *Journal of Information Science*, vol. 27, no. 5, pp. 319-325, 2001.

[14] Wetterström R.,Andersson S., "Web information scraping protection", *ed: Google Patents*, 2009.

[15] Wikipedia , *XQuery*. Available at: https://en.wikipedia.org/wiki/XQuery

[16] Wikipedia. *Web scraping*. Available at: https://en.wikipedia.org/wiki/Web_scraping

[17] Wilbur M., "The Digital Millennium Copyright Act",. *iUniverse*, 2000.

[18] Yu H.T., Guo J.Y., Yu Z.T., Xian Y.T., Yan X., "A novel method for extracting entity data from Deep Web precisely", in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pp. 5049-5053, 2014.

Ahmed Diab. Masters Researchers. He got B.Sc. Information Technology from Islamic University of Gaza - Palestine, (2007-2011), Master degree from Islamic University of Gaza (20013- 2018).

Tawfiq Barhoom. Associated Prof. at Computer Science Department, Faculty of IT, Islamic University-Gaza. He got B.Sc. Computer Science from Omdurman Ahlia University Sudan, (1991-1995), Master degree from Department of Computer Science and Engineering, Shang Hai Jiao Tong University (SJTU)– Shang Hai – China, (1996- 1999) and PhD in Applied computer Technologies, Department of computer science and Engineering, Shang Hai Jiao (2004).