DETECTING ACCESS PATTERNS THROUGH ANALYSIS OF WEB LOGS

Nilani Algiriyage

This dissertation submitted in partial fulfilment of the requirements for the Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2015

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: .... Date: ....

The above candidate has carried out research for the Masters thesis under my supervision.

....

Signature of the supervisor: Date

ABSTRACT

With the evolution of the Internet and continuous growth of the global information infrastructure, the amount of data collected online from transactions and events has been drastically increased. Web server access log files collect substantial data about web visitor access patterns. Data mining techniques can be applied on such data (which is known as Web Mining) to reveal lot of useful information about navigational patterns.

In this research we analyze the patterns of web crawlers and human visitors through web server access log files. The objectives of this research are to detect web crawlers, identify suspicious crawlers, detect Googlebot impersonation and profile human visitors. During human visitor profiling we group similar web visitors into clusters based on their browsing patterns and profile them.

We show that web crawlers can be identified and successfully classified using heuristics. We evaluated our proposed methodology using seven test crawler scenarios. We found that approximately 53.25% of web crawler sessions were from âĂIJknownâĂİ crawlers and 34.16% exhibit suspicious behavior.

We present an effective methodology to detect fake Googlebot crawlers by analyzing web access logs. We propose using Markov chain models to learn profiles of real and fake Googlebots based on their patterns of web resource access sequences. We have calculated log-odds ratios for a given set of crawler sessions and our results show that the higher the log-odds score, the higher the probability that a given sequence comes from the real Googlebot. Experimental results show, at a threshold log-odds score we can distinguish the real Googlebot from the fake.

For the purpose of human visitor profiling, an improved similarity measure is proposed and it is used as the distance measure in an agglomerative hierarchical clustering for a data set from an e-commerce web site. To generate profiles, frequent item set mining is applied over the clusters. Our results show that proper visitor clustering can be achieved with the improved similarity measure.

Keywords: access logs, crawlers, web users, web usage mining

ACKNOWLEDGEMENTS

I would like to thank all the people who gave me a tremendous support in completing my masters research project successfully. My special thanks go to the academic supervisors Prof.Sanath Jayasena and Prof.Gihan Dias for their guidance, ideas, generous support and encouragement throughout the duration of my research.

Then my heartiest gratitude goes to Mr.Kushan Sharma and Mr.Amila Perera for their ideas and support. And also I remember all the staff members at LK Domain Registry and Techcert who gave me support and encouragement.

I would like to thank the Department of Computer Science and Engineering of University of Moratuwa for giving me the opportunity to carry out this research and providing necessary resources.

I greatly appreciate the valuable comments of all members at Research and Development department of LK Domain Registry.

Finally a big thank to my beloved parents, husband and brothers who were always with me providing support and confidence when I most needed it.

CONTENTS

LIST OF FIGURES

List of Tables

LIST OF ABBREVIATIONS

Abbreviation	Description
ART2	Modified Adaptive Resonance Theory
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
RST	Rough Set Theory
SOFM	Self-Organizing Feature Map
SOM	Self Organizing Maps
UB	Usage Based
FB	Frequency Based
VTB	Viewing Time Based
VOB	Visiting Order Based
PC	Possible Crawler
RFC	Request For Comment

1 INTRODUCTION

With the continuous growth and rapid advancement of web based services, the traffic generated by web servers have drastically increased. Analyzing such data which is normally known as click stream data could reveal a lot of information about the web visitors. These data are often stored in web server access log files and in other related resources.

Web usage mining is the process of analysis and discovery of interesting patterns from web resources (data collected at server side, client side and proxy servers). Discovered patterns can be used for many purposes such as web user profiling, web page recommendation, advertising, and intrusion detection.

Web clients can be broadly categorized into two groups: web crawlers and human visitors. Many studies have been done for analyzing patterns of human visitors during the past decades. During recent past the traffic generated by web crawlers has drastically increased [1]. Web crawlers are programs or automated scripts that scan web pages methodically to create indexes. They traverse the hyperlink structure of the worldwide web in order to locate and retrieve information. Web crawler programs are alternatively known as *web robots, spiders, bots* and *scrapers*.

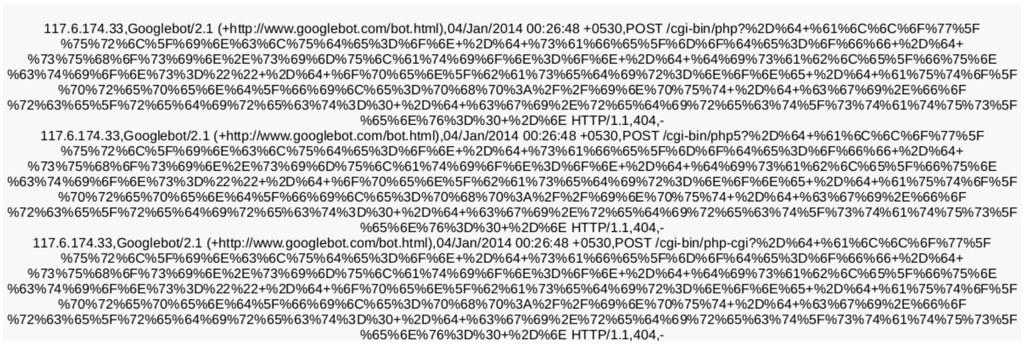
Web crawlers can be used by anyone seeking to collect information available in the Internet. Today web crawler programs as are used for various purposes. Search engines like Google [4], Yahoo [5], msn [6] and bing [7] use web crawlers to index web pages to be used in their page ranking process. Web administrators employ crawlers for automating maintenance tasks such as checking for broken hyperlinks and validating HTML codes. Some crawlers, not from search engines are designed to operate one-time only (e.g. during a particular project), although some but others are programmed for long-term operations. Business organizations, market researchers or anyone can gather specific types of information such as e-mail address, corporate news and product prices.

A recent threat with web crawlers is some try to crawl web sites hiding their own identity and pretending to be some one else. Since Google is the widely used search engine globally and web site owners do not want to block the Googlebot, imposters try to crawl sites impersonating Googlebot. The fact is they get privileged access to web sites using the identity of "Googlebot". Googlebot impersonation can lead to spamming, information theft including business intelligence, or even application level DDoS attacks. Although there were recent news items of fake Googlebots, [8] [9]current understanding of this problem is minimal. While it is possible to later identify (e.g. using web server access log files) these Googlebot imposters by using a reverse DNS lookup and a forward DNS lookup case by case basis[10], doing this real time will be much useful but challenging.

We observed multiple instances of php remote code execution vulnerability [**?**] scans by these fake Googlebots in our test data sets. (Figure 1.1)

In the web usage domain user profiling applies to establishing groups of users exhibiting similar browsing behavior. User profiling helps web site owners in multiple ways : personalization, system improvements such as load balancing, data distribution policies etc. , improve web site's structure, develop recommendation systems and for business intelligence.

Off-line or postmortem analysis of web server access log files could give a deep understanding of traffic patterns and specially to identify offensive web clients. Although the detection is after-the-fact, proactive strategies can be formulated based on the gathered knowledge.



```
117.6.174.33,Googlebot/2.1 (+http://www.googlebot.com/bot.html),04/Jan/2014 00:26:48 +0530,POST /cgi-bin/php?%2D%64+%61%6C%6C%6F%77%5F
%75%72%6C%5F%69%6E%63%6C%75%64%65%3D%6F%6E+%2D%64+%73%61%66%65%5F%6D%6F%64%65%3D%6F%66%66+%2D%64+
%73%75%68%6F%73%69%6E%2E%73%69%6D%75%6C%61%74%69%6F%6E%3D%6F%6E+%2D%64+%64%69%73%61%62%6C%65%5F%66%75%6E
%63%74%69%6F%6E%73%3D%22%22+%2D%64+%6F%70%65%6E%5F%62%61%73%65%64%69%72%3D%6E%6F%6E%65+%2D%64+%61%75%74%6F%5F
%70%72%65%70%65%6E%64%5F%66%69%6C%65%3D%70%68%70%3A%2F%2F%69%6E%70%75%74+%2D%64+%63%67%69%2E%66%6F
%72%63%65%5F%72%65%64%69%72%65%63%74%3D%30+%2D%64+%63%67%69%2E%72%65%64%69%72%65%63%74%5F%73%74%61%74%75%73%5F
%65%6E%76%3D%30+%2D%6E HTTP/1.1,404,-
117.6.174.33,Googlebot/2.1 (+http://www.googlebot.com/bot.html),04/Jan/2014 00:26:48 +0530,POST /cgi-bin/php5?%2D%64+%61%6C%6C%6F%77%5F
%75%72%6C%5F%69%6E%63%6C%75%64%65%3D%6F%6E+%2D%64+%73%61%66%65%5F%6D%6F%64%65%3D%6F%66%66+%2D%64+
%73%75%68%6F%73%69%6E%2E%73%69%6D%75%6C%61%74%69%6F%6E%3D%6F%6E+%2D%64+%64%69%73%61%62%6C%65%5F%66%75%6E
%63%74%69%6F%6E%73%3D%22%22+%2D%64+%6F%70%65%6E%5F%62%61%73%65%64%69%72%3D%6E%6F%6E%65+%2D%64+%61%75%74%6F%5F
%70%72%65%70%65%6E%64%5F%66%69%6C%65%3D%70%68%70%3A%2F%2F%69%6E%70%75%74+%2D%64+%63%67%69%2E%66%6F
%72%63%65%5F%72%65%64%69%72%65%63%74%3D%30+%2D%64+%63%67%69%2E%72%65%64%69%72%65%63%74%5F%73%74%61%74%75%73%5F
%65%6E%76%3D%30+%2D%6E HTTP/1.1,404,-
117.6.174.33,Googlebot/2.1 (+http://www.googlebot.com/bot.html),04/Jan/2014 00:26:48 +0530,POST /cgi-bin/php-cgi?%2D%64+%61%6C%6C%6F%77%5F
%75%72%6C%5F%69%6E%63%6C%75%64%65%3D%6F%6E+%2D%64+%73%61%66%65%5F%6D%6F%64%65%3D%6F%66%66+%2D%64+
%73%75%68%6F%73%69%6E%2E%73%69%6D%75%6C%61%74%69%6F%6E%3D%6F%6E+%2D%64+%64%69%73%61%62%6C%65%5F%66%75%6E
%63%74%69%6F%6E%73%3D%22%22+%2D%64+%6F%70%65%6E%5F%62%61%73%65%64%69%72%3D%6E%6F%6E%65+%2D%64+%61%75%74%6F%5F
%70%72%65%70%65%6E%64%5F%66%69%6C%65%3D%70%68%70%3A%2F%2F%69%6E%70%75%74+%2D%64+%63%67%69%2E%66%6F
%72%63%65%5F%72%65%64%69%72%65%63%74%3D%30+%2D%64+%63%67%69%2E%72%65%64%69%72%65%63%74%5F%73%74%61%74%75%73%5F
%65%6E%76%3D%30+%2D%6E HTTP/1.1,404,-
```

Figure 1.1: PHP remote code execution vulnerability.

1.1 Problem Definition

Although it is expected that web clients will behave well while browsing the web, it is not the case always. Due to the diversity of activities performed, there exists a potential for unethical behavior such as:

- Scanning web sites for security vulnerabilities.
- Hacking servers to access unprotected or unencrypted data.
- Gather business intelligence and confidential data.
- E-mail address harvesting.

- High speed downloads causing excessive usage of bandwidth and ultimately resulting in temporary denial of services where other visitors would not be able to access the web pages at the normal speed.
- Perform crawling hiding their true identity and pretending to be some one else.

  These activities can cause significant loses to individuals and organizations in terms of data, revenue and reputation.

1.2 Objectives

Our objectives are as follows:

- Identify and characterize web crawlers through analysis of web server access log files.
- Develop a web based tool for detecting and characterization of web crawlers.
- Identify impersonation by web crawlers in web access.
  Use Googlebot as an example to develop a methodology to identify web crawler impersonation.
- Identify and profile human visitor behavior using web access logs.
  Cluster human visitors based on their similarities in web browsing behavior and study their common profiles.
- Evaluate proposed methodologies.

1.3 Methodology

Our methodology is summarized below.

- Study the methodologies for web crawler identification and web user profiling.

  Detecting web crawlers requires a lot of effort since the behavior patterns differ from one crawler to the other. As the first step we need to study previous attempts on techniques, trends and approaches for web crawler characterization.

- Pre-process web server access log files and convert them into a suitable format for further analysis. Sessionization.

  Web server log files are plain text files which contain information pertaining to web access attempts. Pre-processing of access log files is a must before any mining task, since there can be lot of irrelevant data. Further the pre-processing steps can be different from one mining task to the other.

  Sessionization is the process of grouping HTTP requests in the log files into sessions. The procedure for session identification includes grouping records in log files by IP address, user-agent and the timeout period.

- Apply supervised learning and unsupervised learning techniques to identify patterns.

  Supervised and unsupervised learning are data mining techniques, where the first one is applied for labeled training data set and the latter is applied for unlabeled data set.

- Profiling.

  Profiling is used to model web visitor behavior based on their web browsing characteristics.

2 LITERATURE REVIEW

Offline analysis of web server access log files can reveal lot of interesting patterns of web client behavior. We first (in Section 2.1) discuss the Apache combined log format of web server access log files and then surveys on web crawler patterns in Section 2.2, in the Section 2.3 about web user profiling and finally Markov Chain Models in section 2.4.

2.1 Web Server Access Log Files

Web server access log files are plain text files which contain information pertaining to web access attempts. They record all the requests processed by the web server. Format of the web server aces log file is configurable. We have used Apache combined log format which is described in the Apache HTTP Server Version 1.3 [11].

Log Format: "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{“user-agent“}i\""

- %h is the IP address of the client (remote host) which made the request to the server. The IP address reported here is not necessarily the address of the machine at which the user is sitting. If a proxy server exists between the user and the server, this address will be the address of the proxy, rather than the originating machine.
- %l is the identity of the user determined by id (not usually used since not reliable)
- %u is the user name determined by HTTP authentication
- %t is the time the server finished processing the request.
- %r is the request line from the client. This log the method, path, query-string, and protocol.
- %>s is the status code sent from the server to the client. It reveals whether the request resulted in a successful response (codes beginning in 2), a redirection (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5)
- %b is the size of the response to the client (in bytes) Referer is the site that the client reports having been referred from. “user-agent“ is the identifying information that the client browser reports about itself.

- %{Referer} is the referer field which list the URL of the previous site visited by the client, which linked to the current page.
- %{user-agent} is the user agent field which provide information about the clientâĂŹs browser,the browser version, and the client‘s operating system. Importantly, this field can also contain information regarding web crawlers.

Table 2.1 lists some example log lines of Apache combined log format from our test data set.

Table 2.1: Apache combined log format.

1	86.96.110.34 - - [17/Jul/2013:00:06:13 +0530] "GET /index.php HTTP/1.1" 200 15683 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0"
2	203.94.92.226 - - [17/Jul/2013:02:03:01 +0530] "GET /infopayments.php HTTP/1.1" 200 24627 "http://www.nic.lk/index.php" "Mozilla/5.0 (Windows NT 5.1; rv:22.0) Gecko/20100101 Firefox/22.0"
3	112.135.249.52 - - [17/Jul/2013:06:44:13 +0530] "POST /lksearch.php HTTP/1.1" 200 4786 "http://www.nic.lk/index.php" "Mozilla/5.0 (Windows NT 5.1; rv:20.0) Gecko/20100101 Firefox/20.0"
4	124.43.229.73 - - [17/Jul/2013:08:35:25 +0530] "GET /index.php HTTP/1.1" 200 15683 "http://www.nic.lk/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.72 Safari/537.36"
5	66.249.85.58 - - [17/Jul/2013:09:22:09 +0530] "GET /index.php HTTP/1.1" 200 15683 "-" "Mozilla/5.0 (Windows NT 6.1; rv:6.0) Gecko/20110814 Firefox/6.0 Google (+https://developers.google.com/+/web/snippet/)"

2.2 Web Crawler Detection

Web crawlers are software programs that automatically traverse the hyperlink structure of the world-wide web in order to locate and retrieve information. In addition to crawlers from search engines, we observed many other crawlers which may gather business intelligence, confidential information or even execute attacks based on gathered information while camouflaging their identity. Therefore, it is important for a website owner to know who has crawled his site, and what they have done.

The impact of the web crawlers on web sites cannot be ignored since anyone with a little knowledge can perform web crawling as there are many tools which make web crawling an easy task. For example wget [12] for Linux environments, HTTrack [13] for windows, known web mining tool rapidminer [14] and many more. Further many programming languages like Python [15], Perl [16], Java [17] provide libraries for web crawler development.

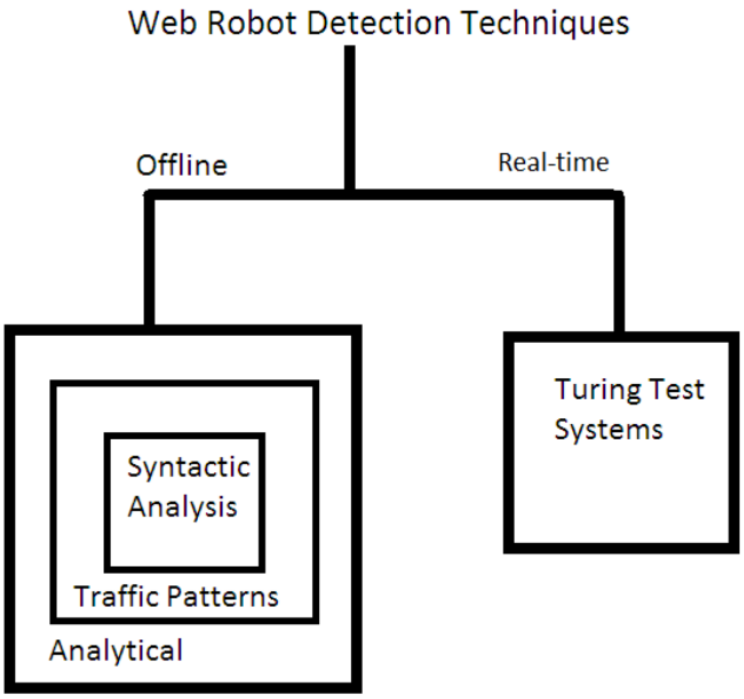


Figure 2.1: Web robot detection methodology hierarchy.[1]

In this section we survey the existing approaches of detection and categorization of web crawlers. Identification of the web crawler requests vs. human visitor requests have been addressed both in off-line and real-time context. Off-line detection often requires identification of user sessions. Sessionization is the process of grouping HTTP requests from single IP, with same user agent during a time period t.

Derek Doren [1] has visualized (Figure 2.1) web robot detection as a hierarchy. According to the increasing strength of each technique, off-line detection includes syntactical log analysis, traffic pattern analysis and analytical learning model. However we noticed some have used combination of these techniques which gave them good results. In Derek hierarchy, real-time detection include Turing test systems. Today analytical models are also used to detect web robots real-time.

In the following subsections these three techniques are discussed in detail.

2.2.1 Syntactical Log Analysis

Syntactical log analysis considers each entry recorded in the log file. Based on the technique used, this can be in three types. The simplest is to detect crawlers based only on the information recorded in the log itself. Popular commercial log-analysis tool AWstats [18] uses this technique. They uses access of “robots.txt“, certain words and characters in the “user-agent“ field such as ’bot’ followed by a space or one of the following characters + :, .; / and term “crawl“.

In the second approach each request is compared against a database of known web crawler “user-agents“. Today there are lots of “user-agent“ parsers in many programming environments [15] [16]. They have a database of “user-agents“ and returns whether the request is from a web crawler or not.

The third approach identifies web robots based on multiple facets and hence can be called as multifaceted log analysis. Huntington et al.[19] talk about using multiple aspects in the detection process starting with the IP address. They have found 0.5% of the robot traffic could be identified using only IP address. Then they have used DNS look up and 16.1% of the traffic has been identified as web robots. For undeclared robots they have used a database of IP addresses and identified 6.5% of the traffic as from web robots. Ultimately the usage of multi-faceted log analysis has resulted in detecting 32.6% of the traffic as from web robots in their test data set. Any way they have declared that their method is only successful about 80% of the time in locating robots.

2.2.2 Traffic Pattern Analysis

This technique considers traffic characteristics such as frequency, bandwidth usage, server status and referrer. First the authors study distinguishing the traffic patterns of web crawlers and then formulate techniques to identify web robots. The first technique is to use the access of “robots.txt“ file. Martijn Koster [20] has proposed Robots Exclusion Protocol to allow web administrators to give instructions to the web crawler using “robots.txt“ file. Whenever a web robot visits a web site, say `http://www.example.com`, it should first check for `http://www.example.com/robots.txt`. The following entry in “robots.txt“ forbids all crawlers from accessing xyz.html.

```
"user-agent":*
Disallow:/xyz.html
```

Although this suggests web robots can be easily detected from “robots.txt“, there are two important considerations when using it. Firstly web crawlers can ignore “robots.txt“ file. Secondly since “robots.txt“ is publicly available file, anyone can see what you are protecting that you don‘t want robots to access. Due to the first fact depending on “robots.txt“ access for web robot detection is unreliable.

2.2.3 Analytical Learning

Analytical techniques use a trained model to predict the class of the new data set. During the training process this technique requires a labeled data set. Data labeling is again a difficult task and analytical learning can be further classified according to the modeling paradigm and techniques used in data labeling process.

Pan-Ning and Vipin Kumar [3] proposes an approach for web robot detection using analytical learning. They discuss the standard approach of user session generation by grouping IP address with “user-agent“ will always not give proper results. That is one IP/“user-agent“ pair can be with multiple users (when users browse through a proxy server). So in the first stage they have introduced a new session generation algorithm as follows. To match log entry lj to its corresponding session they partition the list of currently active sessions H into four groups. First group candidateSet(1) contains the sessions with same IP and “user-agent“. candidateSet(2) contains sessions with same “user-agent“ as lj but with different IP addresses. They have used a reverse DNS look up to resolve host names. Third group candidateSet(3) contains same user agent and prefix IP address as lj. Last group candidateSet(4) contains sessions with same IP address but with different user agents as lj. Then they have used 24 different properties to identify web robot sessions Table 2.2. After deriving the session features classification model has been built using C4.5 decision tree algorithm to achieve their two main objectives; (1) find a good model for predicting web robot sessions based on their access features, and (2) to detect robotic activities as soon as possible. According to the experimental results web robots can be detected more than 90% accuracy after 4 requests.

Table 2.2: Summary of attributes derived [3].

Id	Attribute Name	Remark	Purpose
1	TOTALPAGES	Total number of pages requested	Feature
2	%IMAGE	%of image pages(.gif/.jpg) requested	Feature
3	%BINARY DOC	%of binary documents(.ps/.pdf) requested	Feature
4	%BINARY EXEC	%of binary program files(.cgi/.exe) requested	Feature
5	ROBOTS.TXT	No of times the robots.txt file is accessed	Labeling
6	%HTML	%of HTML pages requested	Feature
7	%ASCII	%of ASCII files(.txt/.java) requested	Feature
8	%ZIP	%of compressed files(.zip/.gz) requested	Feature
9	%MULTIMEDIA	%of multimedia files(.wav/.mpg) requested	Feature
10	%OTHER	%of other file formats requested	Feature
11	TOTALTIME	Temporal sever session length	Feature
12	AVGTIME	Average time between clicks	Feature
13	STDEVTIME	Standard deviation oftime between clicks	Feature
14	NIGHT	%of requests made between 2am to 6am	Feature
15	REPEATED	Reoccurance rate of file requests	Feature
16	ERROR	%of requests with status>=400	Feature
17	GET	%of requests made with GET method	Feature
18	POST	%of requests made with POST method	Feature
19	HEAD	%of requests made with HEAD method	Labeling
20	OTHER	%of requests made with other method	Feature
21	WIDTH	width of the traversal(in the URL space)	Feature
22	DEPTH	depth of the traversal(in the URL space)	Feature
23	PATHLENGTH	server path length(no of requests)	Feature
24	REFERRER="-"	requests with referrer=""	Labeling

Dusan Satavonovic et al. [21] uses seven well-established data mining classification algorithms on a static web server access log in order to (1) classify web user session as from web robots or human visitors and (2) identify web robots sessions exhibit malicious behavior which could be potential participants of a DDoS attack. Based on previous work they have adopted seven different features and new two futures of their own to distinguish between browsing patterns of web robots and humans. These features as click number, HTML-to-image ratio, and percentage of PDF/PS file requests, percentage of 4xx error responses, percentage of HTTP requests of type HEAD, percentage of requests with unassigned referrer, "robots.txt" file requests, standard deviation of requested page depth and percentage consecutive sequential HTTP requests. They have done two types of classification experiments. For the experiment #1 they have obtained a labeled data set of human sessions and well-behaved crawler sessions. Aim of the first approach is to examine whether human sessions and well-behaved crawler sessions can be separated by classification algorithms. For the data labeling they have used a database of "user-agent" fields of crawlers and browsers.

For the experiment #2 they have grouped human sessions and well-behaved crawler sessions into one class and malicious web crawlers or unknown visitors into the other. As in the previous case same database is used to label the sessions. After the session labeling the have used C4.5, RIPPER, k-nearest neighbor, naive Bayesian, Bayesian network, Support Vector Machine and neural network classification algorithms to predict the class label of the test data set. To evaluate the performances of the algorithms they have used recall, precision and F1 score. According to the results classification accuracy of neural network, C4.5, RIPPER and k-nearest neighbor is high and they have concluded that there is a significant different between behavior of web crawlers and human users. However identifying crawler who mimics the behavior patterns of human users is difficult with their approach.

Stassopoulou and Dikaiakos [22] used a Bayesian network approach to distinguish between web crawler and human user sessions. After the identification of sessions they have used the six features based on the findings by Dusan et al. [21] : maximum sustained click rate, session duration, percentage of image requests, percentage of pdf/ps requests, percentage of requests with 4xx errors and "robots.txt" file requests. Using these features they have formed the Bayesian network. To train the Bayesian network they have created a labeled data set based on the following four heuristics: (1) IP address of known crawlers, (2) "robots.txt" file requests, (3) session duration values extending over period of three hours, and (4) HTML to image ratio more than 10 HTML files per image file. Experiments were performed on the test data set. Results show that 0.95 recall can be achieved using Bayesian network in the situations where the training set contains an equal number of crawler and human sessions.

2.2.4 Real-time Web Crawler Detection

The most common example for real time crawler detection is the CAPTCHA test. Von Ahn et al. [23] introduced Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). CAPTCHA codes are very hard to read and understand by conventional software programs and normally contain *.gif* images with scrambled words or recordings. Over the past years this technique has been successfully used to detect humans from web crawlers. Additionally hackers and spammers were blocked using CAPTCHA tests. Figure 2.2 shows an example CAPTCHA test on a web site.

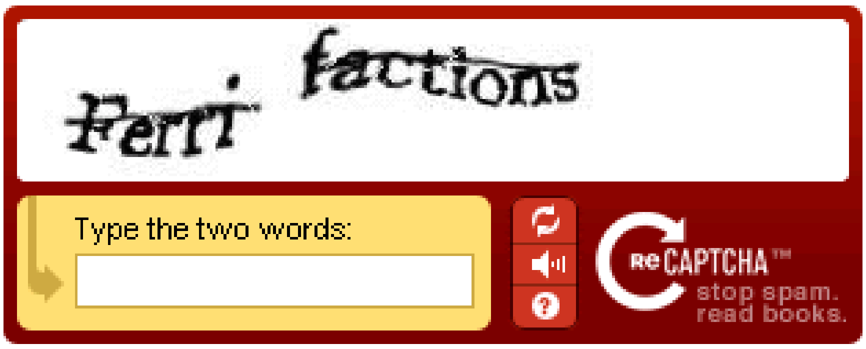


Figure 2.2: An Example of a CAPTCHA test.

Balla et al. [24] has proposed a methodology to detect web crawlers real time. The system reads the incoming HTTP requests and feeds them to a hashing table. They have trained a Bayesian classifier for the classification of IP addresses. Their results have shown that the accuracy of detection algorithm is more than 80%.

2.2.5 Suspicious Web Crawler Detection

Suspicious web crawlers are a challenge for web site security. Since "suspicious" is qualitative measure, identifying them may be different upon the context and user. We discussed possible features of suspicious web crawlers in Section 1.1. Few studies have been done on detection of suspicious web crawlers. Dusan and Natalija [25] used unsupervised clustering and observed a good separation between suspicious and non-suspicious behavior. They have collected university server's access log files and identified web crawler sessions based on traditional features and two novel features. These two features are Consecutive Sequential HTTP Request Ratio and Standard Deviation of Page Request Depth. They have applied both Self Organizing Maps(SOM) and Modified Adaptive Resonance Theory(ART2) unsupervised neural network algorithms. According to their results suspicious web crawlers exhibit a range of browsing patterns while human users follow rather similar browsing patterns. Importantly 52% of suspicious crawlers exhibit human-like browsing behaviour. Algiriyage et al. [26] have used hidden links and other features like IP blacklist checks, anomalies in the "user-agent" field to identify suspicious web crawlers.

Nong Ye et al. [27] has used a Markov chain model to detect intrusions in a computer and network system. In their study temporal behavior of the normal profile is learnt from historic data and new observed behavior is analyzed to derive probabilities. Testing results show that intrusive activities have a very low probability compared to the normal activities.

DeXiang et al. [28] have developed algorithms using a combination of machine learning and advanced metrics to detect suspicious crawlers. In their work they have proposed two approaches (1) TSSNBS (Too Simple Sometimes Naive Blocking Schema) and (2) ABS (Adaptive Blocking Schema) to block malicious crawlers. According to their results after about 60 seconds, the detection ratio is stabled at about 86% which is a fair amount. Today there are also many crawlers that impersonate well-known web crawlers. For example, it has been observed that Google‘s Googlebot crawler is impersonated to a high degree. This raises ethical and security concerns as they can potentially be used for malicious purposes. However detection of Googlebot impersonation is not addressed in the research community yet, possibly because this is a recent trend.

2.3 Web User Profiling

Mobarsher et al. [2] introduced the taxonomy of web mining (Figure 2.3) for the data mining activities performed on web data. There are major two types of web mining which is refereed as web content mining and web usage mining. Web content mining engages in automatically searching information resources in web pages and web usage mining is related to discovering user access patterns from web usage data. We have focused on web usage mining in the context of web visitor profiling.

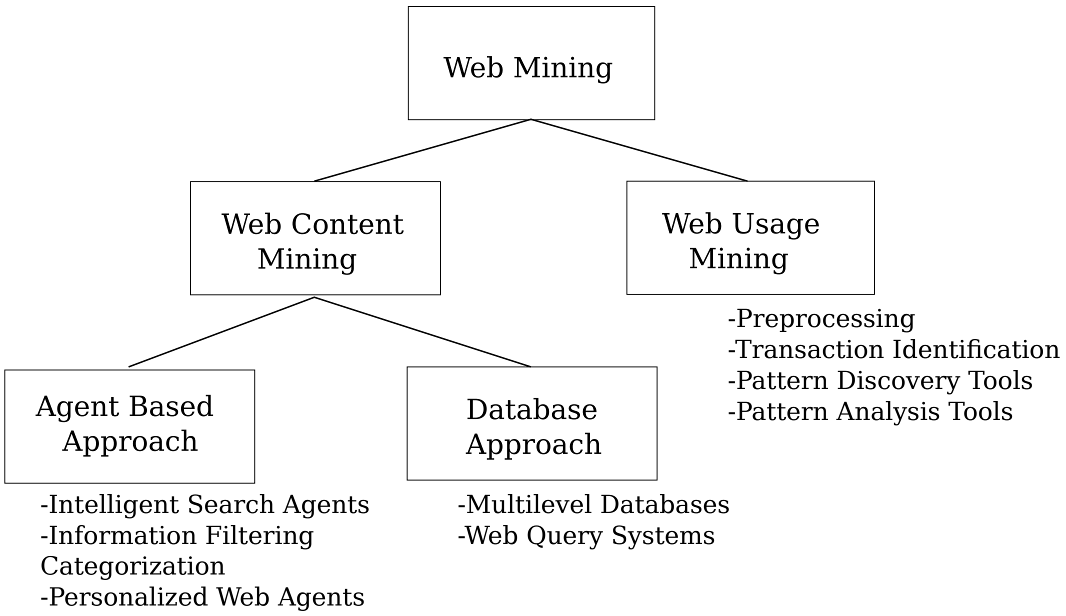


Figure 2.3: Web mining taxonomy [2].

Statistical analysis and various data mining techniques have been successfully applied over web data to extract useful patterns. The study of Stermsek et al. [29] is an example for using statistical analysis and graph analysis for user profiling. After deriving user sessions they have measured statistics such as how many web pages have been visited in a particular session, significant web pages and accumulated time spent on each page. Adjacency matrix is created to identify pages requested by the user and how the user got into different pages. Derivation of user profile is based on both graph mining and statistical analysis. Other than statistical analysis, association rule mining, sequential pattern mining, classification, clustering have been applied to group web visitors with similar characteristics.

Generally a profile contains facts about someone's interests and behavior. The content of the user profile can be changed based on the context. However most common contents of a user profile can be user's interaction preferences, user's knowledge, user's interests and background and skills. In the web usage domain user profiling applies to establishing groups of users exhibiting similar browsing behavior. User profiling helps web site owners in multiple ways : personalization, system improvements such as load balancing, data distribution policies, improve web site's structure, develop recommendation systems and business intelligence.

Xie et al. [30] has proposed a distance measure for clustering based on Dempster-ShaferâĂŹs theory of combining evidence. After the general pre-processing steps they have calculated the basic probability assignment (bpa) for each session. They have proposed belief function as the similarity measure to be used in greedy clustering. A user session is assigned to the cluster based on the similarity measure. After assigning all user sessions to different groups, DempsterâĂŹs rule of combination has been applied to get the common user profiles.

In another research Xu et al. [31] has tested the feasibility of applying k-means algorithm to cluster web users. First they have generated the the matrix of web users and web pages. To calculate the similarity between vectors of two users they have applied standard cosine similarity. They have used standard k-means algorithm in the following way: (1) Place *k* points to represent initial group centroids (2) Assign each object to the group that has the closest centroid (3) When all objects have been assigned, recalculate the positions of the *k* centroids (4) Repeat Steps 2 and 3 until the centroids no longer move. Their results show a clear separation of the clusters.

Sharma et al. [32] has used roughest clustering for a proxy log data set. Rough Set Theory (RST) is an approach to aid decision making in the presence of uncertainty. They have used cookie data to identify users. In their algorithm they have four main modules: *preprocessing_log, RST, threshold_calculator and matcher*. The *preprocessing_log* module takes the log file as input and produces transaction set while *RST* inputs transaction set and outputs clustered set. *Threshold_calculator* calculates the total number of pages in various sessions and *matcher* finds the equivalence set. Ultimately they get user sessions clustered based on maximum pages visited.

Fuzzy clustering has been applied in multiple research works. The idea behind the fuzzy clustering is to enable the generation of overlapping clusters. This has been successfully applied in the web usage mining process by Nasraoui et al. [33] Suryavanshi et al. [34] and Castellano et al. [35].

In order to understand similarity between two visitors, similarity measures are required. Jitian et al. [36] talk about different similarity measures that can be applied for web log data. The following are the measures they discuss:

For a given web site there are m sessions $S = \{s_1, s_2, ..., s_m\}$accessing n web pages $P = \{p_1, p_2, ..., p_n\}$. For each page p_i, and each session s_j, a *usage value* is associated, denoted as $use(p_i, s_j)$ and defined as:

$$use(p_i, s_j) = \begin{cases} 1 & \text{if } p_i \text{ is accessed in } s_j \\ 0 & \text{otherwise} \end{cases}$$

The first measure is based on common pages and total pages accessed by both sessions. This one is described as a usage based measure(UB).

$$sim(s_i, s_j) = \frac{\sum_k (use(p_k, s_i) * use(p_k, s_j))}{\sqrt{\sum_k use(p_k, s_i) * \sum_k use(p_k, s_j)}} \tag{2.1}$$

Second one is a frequency based(FB) measure where $a_w(p_k, s_i)$ is the total number of times that the user of session s_i, accesses the page p_k at site w. This is based on total number of times they access common pages at all sites.

$$sim(s_i, s_j) = \frac{\sum_{kw}((a_w(p_k, s_i) * a_w(p_k, s_j))}{\sqrt{\sum_{kw}(a_w(p_k, s_i))^2 * \sum_{kw}(a_w(p_k, s_j))^2}} \tag{2.2}$$

Thirdly similarity between two sessions can be measured by taking actual time users spent viewing each page. Let user session s_j spent $t(p_k, s_j)$ on viewing page p_k. In this case similarity is known as viewing time based(VTB) which can be expressed by:

$$sim(s_i, s_j) = \frac{\sum_k (t(p_k, s_i) * t(p_k, s_j))}{\sqrt{\sum_k (t(p_k, s_i)^2 * \sum_k (t(p_k, s_j)^2}} \tag{2.3}$$

Accessing order of web pages is important in some applications. Let Q^i and Q^j are the navigation paths accessed by users in session s_i and s_j, respectively. Similarity between sessions can be defined as the natural angle between paths Q^i and Q^j.

$$sim(s_i, s_j) = \frac{< Q^i, Q^j >}{< Q^i, Q^i >_l . < Q^j, Q^j >_l} \tag{2.4}$$

where $l = min(length(Q^i), length(Q^j))$. This is known as visiting order based(VOB) measure.

All these similarity measures describe the way to calculate pairwise similarity. Next they have described the concept of similarity matrix which shows the similarity between all user sessions. Further matrix based clustering algorithm is described and they have used all four similarity measures to cluster sessions. Results shows that the number of clusters produced for VOB-based measure is always greater than that of others and UB-based measure is always the smallest among the four measures.

A new similarity measure is proposed by Velásquez et al. [37] considering access sequences, time spent on each page and the type of web page. They have derived the measure in the following way:

Let α and β be two visitor behaviour vectors of cardinality C^{α} and C^{β} respectively and $\Gamma(.)$ is a function that, applied over α or β, returns the navigation sequence corresponding to a visitor vector.

$$sm(\alpha,\beta) = dG(\Gamma(\alpha),\Gamma(\beta))\frac{1}{\eta}\sum_{k=1}^{\eta}\tau_k * dp(p_{\alpha,k},p_{\beta,k}) \tag{2.5}$$

where dG is the similarity between sequences of pages visited. $\eta = min\{C^{\alpha},C^{\beta}\}$ and τ_k is an indicator of visitor's interest of the web page visited. The term τ_k is defined as $\tau_k = min\{\frac{t_{\alpha,k}}{t_{\beta,k}}\frac{t_{\beta,k}}{t_{\alpha,k}}\}$. The term $dp(p_{\alpha,k},p_{\beta,k})$ is the similarity of pages visited. This is the angle cosine similarity between two word-page vectors.

For the term τ_k they have considered that the time spent on web page is proportional to the interest the visitor has in its contents. If the times spent by visitor α and β on k^{th} page that they have visited are close to each other the value(τ_k) becomes close to 1 and otherwise it will be close to 0. Further they have considered the content of the web page as the third portion of the similarity measure. Finally they have applied Self-Organizing Feature Map(SOFM) for session clustering. According to the results they have ended up with four useful clusters. Clustering results have been used to do structural changes in the web site.

2.4 Markov Chain Model

A.A. Markov [38] studied how the past outcomes can influence the prediction of outcome of the next experiment when we observe a sequence of chance experiment. Based on his findings in a first order Markov chain, prediction for the outcome of next experiment depends only on the current state. We can define a Markov chain as [39]:

- A set of states Q

- For each pair of states i and j, a transition probability a_{ij} from state i to j

- $\sum a_{ij} = 1$

In a Markov chain model, we transition from one state to another in discrete time steps $n = 1,2,3...$ If we are in state i at time step n, we go to time step j in time step $n+1$ with probability a_{ij} and it is the transition probability from state i to state j. State at time n, x_n depends only on the most recent state which is known as Markov property.

$$p(x_n = j | x_0, x_1, x_2, ..., x_{m-1}, x_m = i) = P(x_n = j | x_m = i), m < n. \tag{2.6}$$

If $m = n-1$ this is a_{ij}.
Now let us look at the probability obtaining a sequence $x_1, ...x_n$ from our Markov chain. This is basically the probability of a path $x_1, ...x_n$ in the chain.

$$p(x_1...x_n) = p(x_1)a_{x_1x_2}...a_{x_{n-1}x_n} = p(x_1)\prod_{i=1}^{L} a_{x_ix_{i+1}} \tag{2.7}$$

$P(x_1)$ is known as the probability of initial state. Let this special start state be denoted by 0. Then $p(x_1) = a_{0x_1}$. Therefore, the probability of the path will be:

$$p(x_1...x_n) = p(x_0 = 0, x_1...x_n) = a_{0x_1}a_{x_1x_2}...a_{x_{n-1}x_n} = \prod_{i=0}^{n-1} a_{x_ix_{i+1}} \tag{2.8}$$

Laplacian Smoothing

Probabilities in the transition matrix sometimes can be very small and hence generate a sparse matrix. In such cases we can apply Laplacian smoothing to avoid zero probability values. Laplacian smoothing engages in adding a pseudo-count for both nominator and denominator using the following Equation 2.9. Here m is the number of possible states and n_{ik} be the number of times that the process moved from state i to k.

$$\hat{\mathbf{P}}_{ij} = \frac{n_{ij} + \alpha}{\sum_{k=1}^{m} n_{ik} + n_{ij}\alpha} \tag{2.9}$$

2.5 Summary

According to the review provided in this chapter, many studies have been done on the pattern analysis using web server access log files. The techniques they have used varies from simple statistical analysis to complex data mining tasks. Among the available literature for web crawler analysis, Pan-Ning and Vipin Kumar [3] provide a comprehensive study on the features that can be applied over web server access log files to identify possible web crawlers. Although some have studied the patterns of malicious web crawlers, unfortunately we didn't find any previous work on web crawler impersonation attacks. For the purpose of web user profiling unsupervised learning is applied in order to generate profiles. Jitian et al. [36] and Velásquez et al. [37] have proposed similarity measures to be used in clustering tasks.

3 METHODOLOGY

This chapter presents our methodology. In the Section 3.1 methodology for web crawler identification and characterization is presented and in Section 3.2 methodology for detecting Googlebot impersonation is discussed. The methodology for human visitor profiling is described in Section 3.3.

3.1 Web Crawler Identification and Characterization

During the first portion of the research, web crawler behaviour patterns were studied. As the first step we developed a methodology to identify web crawlers through analysis of web server log files and to characterize them. We extended the study to identify Googlebot impersonation. Figure 3.1 shows our approach in summary.

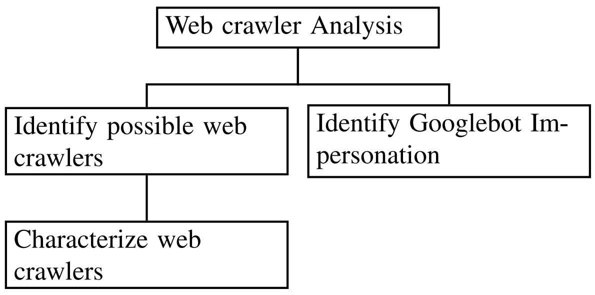

Web crawler Analysis
Identify possible web crawlers
Identify Googlebot Impersonation
Characterize web crawlers

Figure 3.1: Summary of web crawler characterization.

Figure 3.2 shows the overall methodology for identification and characterization of web crawlers. We used two special modules to handle the problem. First one is *“IDENTIFIER“* which was used to identify possible web crawlers from the log files. Secondly *“CLASSIFIER“* module was used to categorize identified web crawlers.

3.1.1 Data Preparation-Web Crawler Detection

3.1.1.1 Data Cleansing

Data cleansing process can be different based on the mining task. For the purpose of web crawler identification we removed the incomplete log lines from the log file. We observed that some log files were not in order of the time-stamp and hence we sorted the log.

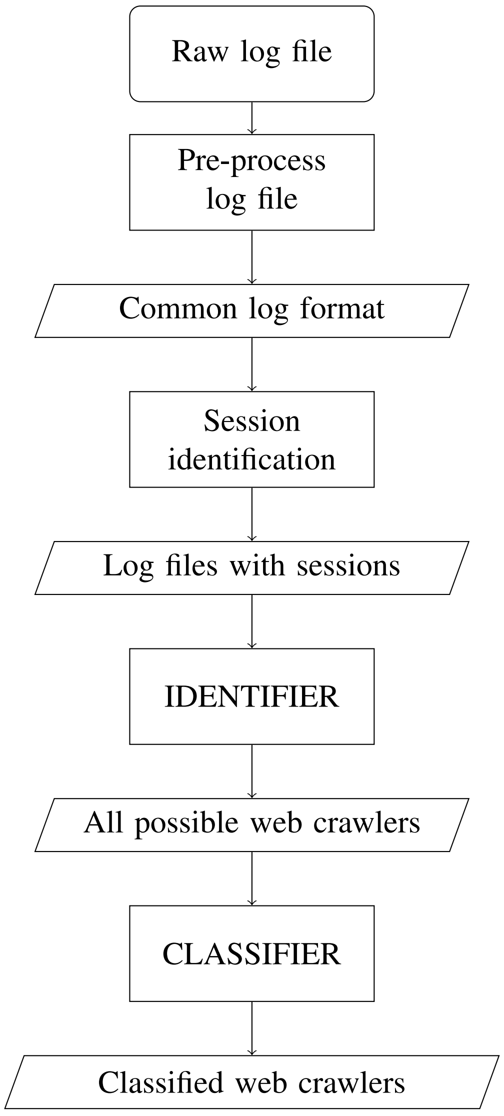


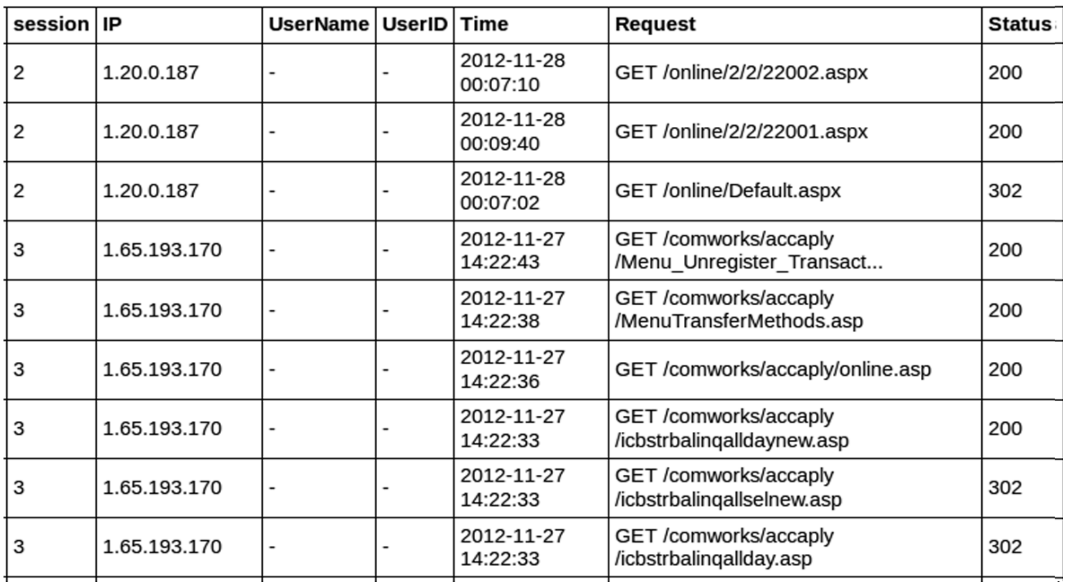
Figure 3.2: Methodology for identification & characterization of web crawlers.

3.1.1.2 Convert Log File

Log files can be in different formats [40]. In this research all log files were converted into Apache combined log file format discussed in Section 2.1.

3.1.1.3 Session Identification

The goal of session identification is to group one single user's activities within a certain time period. The user is identified using IP address, and "user-agent" information. We considered the default time out period 30 minutes as suggested by Cooley et al.[41]. Given n web pages in the web site and m web users visiting the web site during a period of time, collection of pageviews can be expressed as $P = \{p_1, p_2, ...p_n\}$ and session collection can be expressed as $S = \{s_1, s_2, ...s_m\}$. Figure 3.3 is an example for sessionized log file.



session	IP	UserName	UserID	Time	Request	Status
2	1.20.0.187	-	-	2012-11-28 00:07:10	GET /online/2/2/22002.aspx	200
2	1.20.0.187	-	-	2012-11-28 00:09:40	GET /online/2/2/22001.aspx	200
2	1.20.0.187	-	-	2012-11-28 00:07:02	GET /online/Default.aspx	302
3	1.65.193.170	-	-	2012-11-27 14:22:43	GET /comworks/accaply /Menu_Unregister_Transact...	200
3	1.65.193.170	-	-	2012-11-27 14:22:38	GET /comworks/accaply /MenuTransferMethods.asp	200
3	1.65.193.170	-	-	2012-11-27 14:22:36	GET /comworks/accaply/online.asp	200
3	1.65.193.170	-	-	2012-11-27 14:22:33	GET /comworks/accaply /icbstrbalinqalldaynew.asp	200
3	1.65.193.170	-	-	2012-11-27 14:22:33	GET /comworks/accaply /icbstrbalinqallselnew.asp	302
3	1.65.193.170	-	-	2012-11-27 14:22:33	GET /comworks/accaply /icbstrbalinqallday.asp	302

Figure 3.3: Sessionalized log file example.

3.1.2 Identifier

The *“IDENTIFIER“* module was used to identify possible web crawlers from the web log file based on described features. A detailed view of the module is shown in Figure 3.4. Crawler sessions were considered as class=1 and others as class=0.

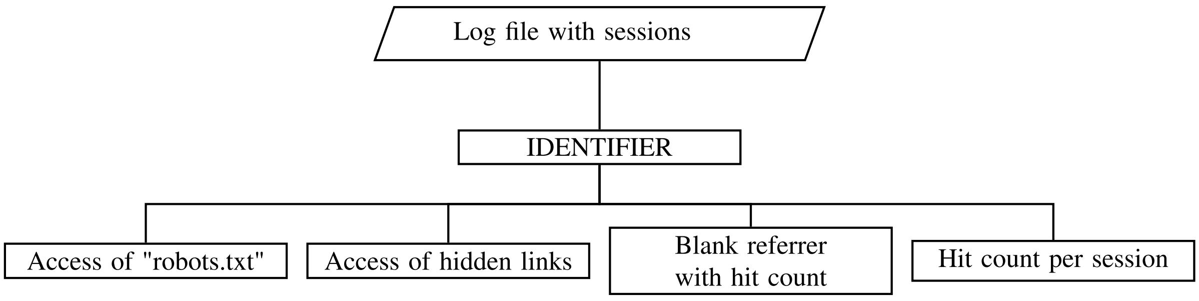


Figure 3.4: Flow chart of “IDENTIFIER“ module.

3.1.2.1 Access of “robots.txt“ File

According to the Robots Exclusion Protocol [20], it is expected that crawlers should first acknowledge “robots.txt“ before downloading any cont from the web site. There for given a user session s_i, the possible crawler (pc) property is characterized as follows.

$$pc(s_i) = \begin{cases} 1 & if \text{ ``}robots.txt\text{``} \; is \; requested \; in \; s_i \\ 0 & Otherwise \end{cases} \tag{3.1}$$

But in practice there are many web crawlers that either do not access "robots.txt" or even if accessed they do not follow the rules. Hence we have used three other checks which detect crawler sessions, as described in the following.

3.1.2.2 Access of Hidden Links

A novel technique that we introduced is to use hidden links in the web page. Hidden links are not visible in browsers. This technique acts as a honeypot in the web page and it is possible to identify someone accessing this as a web crawler with high confidence. We have implemented three types of hidden links.

- Hidden link type I

$$< a \; href = \text{''}link1.html\text{''} \; style = \text{''}visibility : hidden\text{''} > link1.html < /a >$$

- Hidden link type II

$$< a \; href = \text{''}link2.html\text{''} \; style = \text{''}color : white\text{''} > link2.html < /a >$$

- Hidden link type III

$$<!-- < a \; href = \text{''}link3.html\text{''} \; style = \text{''}color : white\text{''} > link3.html < /a > -- >$$

The three types of hidden links differ from each other. The first type of hidden link is not visible in the browser at all but the second type is visible in the mouse hover event. The third type of link is put inside an html comment (comments are used by developers as a best practice to provide information about the code). As in the previous one access of hidden link property can be expressed as:

$$pc(s_i) = \begin{cases} 1 & if \; hiddenlink \; is \; accessed \; in \; s_i \\ 0 & Otherwise \end{cases} \tag{3.2}$$

3.1.2.3 Hitcounts

Hit count in a particular session is a numerical attribute calculated by counting the HTTP requests during each session. A session entry $s_i \in S$ includes IP address $s_i.ip$, the session id $s_i.id$, accessed page $s_i.p$, time of access $s_i.time$, referrer $s_i.referrer$ and “user-agent“ string $s_i.ua$. Let T be the time of first page request in each session. We have considered only pages and access time for construction of the model.

$$s_i =< s_i.id, (s_i.p_k, s_i.time_t), ..., (s_i.p_k, s_i.time_t) > \tag{3.3}$$

Where for $1 < k < n$, $T < t < T + 30$

Hit count (H_i) in a particular session can be expressed as in

$$H_i = \sum(s_i.ip)\ Where\ T < t < T + 30 \tag{3.4}$$

If the hit count in a given session is greater than our threshold value (X) we considered session to be by a possible crawler, because we observed the hit count within a session is high for web crawlers.

$$pc(s_i) = \begin{cases} 1 & if\ H_i > X \\ 0 & Otherwise \end{cases} \tag{3.5}$$

3.1.2.4 Blank Referrer Hit Count

Most of the crawlers initiate HTTP requests with an unassigned referrer. Pan-Ning and Vipin Kumar [3] also have used this as a feature to identify web crawler requests. But in our approach we have used this as another numerical attribute by calculating the hit count with blank referrer.

$$H_i = \sum(s_i.ip)\ Where\ T < t < T + 30\ and\ s_i.referrer = `-` \tag{3.6}$$

If the hit count in a given session is greater than our threshold value (Y) we considered session to be by a possible crawler.

$$pc(s_i) = \begin{cases} 1 & if\ H_i > Y \\ 0 & Otherwise \end{cases} \tag{3.7}$$

Ultimately we get possible crawler list marked as class=1 and the rest as class = 0. We have considered only web crawler sessions under this part of the research and ignored the sessions with class=0.

3.1.3 Classifier

The next step of our approach is the usage of the “Classifier“ module to classify possible crawlers. The idea is to verify the “user-agent“ string. We created a white list of verified “user-agent“ strings of “known“ crawlers. Our possible crawler list was checked against this white list and we filtered out the “known“ crawler sessions. For the remaining entries in our possible crawler list we conducted several tests to assess suspicious behavior of crawlers which are described below. This process is depicted in Figure 3.5.

3.1.3.1 Anomalies in “user-agent“ Field

As the first step “user-agent“ field of crawlers was checked to identify suspicious behavior patterns. The “user-agent“ is defined by RFC 2616 under section 14.43 states “User agents SHOULD include this field with HTTP requests“ [42]. However some crawling tools allow users to omit the “user-agent“ field. We classify requests coming

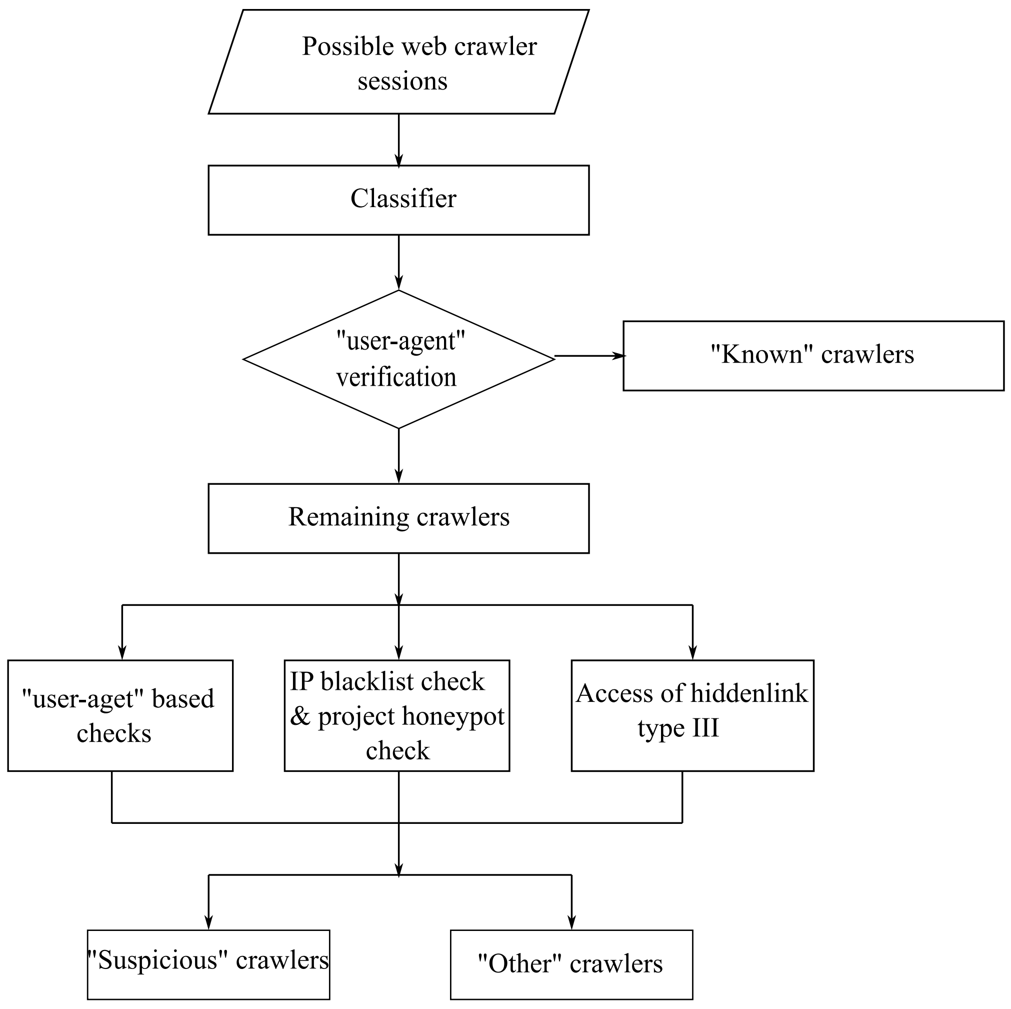

Possible web crawler sessions
Classifier
"user-agent" verification
"Known" crawlers
Remaining crawlers
"user-aget" based checks
IP blacklist check & project honeypot check
Access of hiddenlink type III
"Suspicious" crawlers
"Other" crawlers

Figure 3.5: Flow chart of the “CLASSIFIER“ module.

with un-assigned “user-agent“s as suspicious since it violates RFC2616.

User agents rely a great deal upon trust. RFC2616 does not state that the user cannot alter default value of the “user-agent“. The servers generally trust that the user will not modify the “user-agent“ field. Since the trust cannot be verified, a user can manipulate the “user-agent“ field to pretend to be any “user-agent“. This is not addressed in RFC2616 and some use crawlers for malicious purposes masking their true identity. We classified the identified crawlers based on the “user-agent“ into two categories. They are:

- “user-agent“ has been masked into a known web crawler (e.g., google, bing, baidu, msn): “user-agent“ verification confirmed that site was crawled using a fake known crawler user agent. Such cases were classified as “suspicious“.

- “user-agent“ appeared to be of a web browser (Firefox, Internet Explorer, Opera): This type of entries can be of either browser-based crawler or the user has altered their identity (“user-agent“) pretending to be a known browser. Such behavior was classified as “suspicious“ crawlers if they violate RFC2616 and Robot exclusion protocol or caught by other suspicious crawler classification checks described below.

3.1.3.2 Access Hidden Link Inside HTML Comments

We identified that some crawlers have accessed the hidden link inside html comments (hidden link type III described previously). Since âĂIJknownâĂİ crawlers have never accessed this type of hidden link we found the behavior of accessing hidden links inside html comments as suspicious and categorized them âĂIJsuspiciousâĂİ crawlers.

3.1.3.3 IP Blacklist Checks

In addition to the above checks we conducted an IP blacklist check [43] [44] and project honeypot database [45] check to identify blacklisted crawler IP addresses. Such crawlers were classified as “suspicious“. We also observed that some of the “known“ crawler IP addresses are also blacklisted. How ever we ignored them since we have verified and identified “known“ crawlers. The rest of the crawlers in our identified list were classified as “other“ crawlers since they have shown some characteristics of crawlers but cannot be classified as either “known“ or “suspicious“.

3.1.4 Summary

Our characterization of web crawlers include three types.

- Known web crawlers : These are the crawlers from well-known search engines, which help to drive more users to the web site and hence increase the reputation of the web site. On the other hand for e-commerce web sites these will help to generate more revenue.

- Suspicious web crawlers : These are the crawlers trying to exploit vulnerabilities and threat for site integrity and availability.

- Other crawlers : The intent of these crawlers is hard to define. They may or may not be harmful.

Details of the experimental set up and results are discussed in Chapter 4.

3.2 Detection of Googlebot Impersonation

Over the past decade Internet traffic generated from web crawlers has increased drastically [1]. Presently web crawler programs can be operated by any one and can be used for a variety of purposes.

Nong Ye et al. [27] has used a Markov chain model to detect intrusions in a computer and network system by learning temporal behavior of the normal profile from historic data and new observed behavior is analyzed to derive probabilities. We propose a similar approach using Markov chain models to learn profiles of real and fake Googlebots based on their patterns of web resource access sequences.

In our problem the transition probabilities between states have to be differ for fake Googlebot and real Googlebot. Therefore, we need to build two Markov chains one for each. Then given a sequence, we compute the probability p for obtaining the sequence in the real Googlebot, and the probability q of obtaining the sequence in the fake Googlebot Markov chain. The log-odds ratio[46] of these two can be used to determine whether the sequence is coming from real Googlebot or fake. Following are the steps:

- Have the labeled sequences from real Googlebot and fake Googlebot

- Calculate the transition probabilities between states for the real Googlebot and fake

We have calculated log-odds ratios for a given set of crawler sessions and our results show that the higher the log-odds score, the higher the probability that a given sequence comes from the real Googlebot. Experimental results show, at a threshold log-odds score we can distinguish the real Googlebot from the fake.

Now, given a sequence x, compute p(x) for each Markov chain, denote these by $p(x|real)$ and $p(x|fake)$. Then we use the log-odds ratio to determine if x is coming from real Googlebot or not.

$$S(x) = log\frac{p(x|real)}{p(x|fake)} = log\frac{\prod_{i=0}^{n} a^{real}_{xix_{i+1}}}{\prod_{i=0}^{n} a^{fake}_{xix_{i+1}}} = \sum_{i=1}^{n-1} log\frac{a^{real}_{xix_{i+1}}}{a^{fake}_{xix_{i+1}}} \tag{3.8}$$

Lengths of the sequences are different and to normalize this ratio by length, we used length-normalized log-odds ratio of $S(x)/|x|$.

The overall methodology for fake-Googlebot detection is depicted in Figure 3.6.

3.2.1 Data Preparation-Googlebot Impersonation

We collected web server access log files from an e-commerce web site for a period of about 10 weeks. Size of the data set was 409.2MB and contained 1,633,913 log entries before pre-processing. We considered 75% of the data set as training data and 25% as test data.

Table 3.1: Summary of the data set.

Period of the log file	69 Days
Size of the log file	409.2 MB
Number of HTTP requests before pre-processing	1,633,913
Number of HTTP requests after pre-processing	1,633,578

Log data has to be pre-processed before applying our techniques as shown in Figure 3.6. Data cleansing and formatting are the two major pre-processing operations performed. Details of the data preparation steps are described in the following section.

3.2.1.1 Data Cleansing

During the process of data cleansing we have removed lines of the log file which were not fully recorded and did not satisfy the sufficient length of a log line. We observed that some records of the log file were not in sorted order. We sorted the log file in the ascending order of the recorded time stamp.

3.2.1.2 Data Formatting

Web servers record requests processed by the server in different formats. We have observed that Apache common log format is very comprehensive. Original log files were converted into Apache combine log format [11].

3.2.1.3 Identification of Requests from Google Crawlers

For our experiments we needed only the HTTP requests from Google crawler. We removed all the requests not containing the string "Googlebot" in “user-agent“ field from our access logs.

After the identification of requests from Googlebot, a forward and backward DNS lookup was performed to verify requests from real Googlebot [10].

- Identification of real Googlebot :
  Forward and backward DNS lookup verified the Google host name. Table 3.2 lists the “user-agent“s used by real Googlebot while browsing our web site.

- Identification of fake Googlebot :
  Although the “user-agent“ indicated from Google, the DNS verification resulted the host name not from Google. Table 3.3 lists the “user-agent“s used by fake Googlebot in our data set.

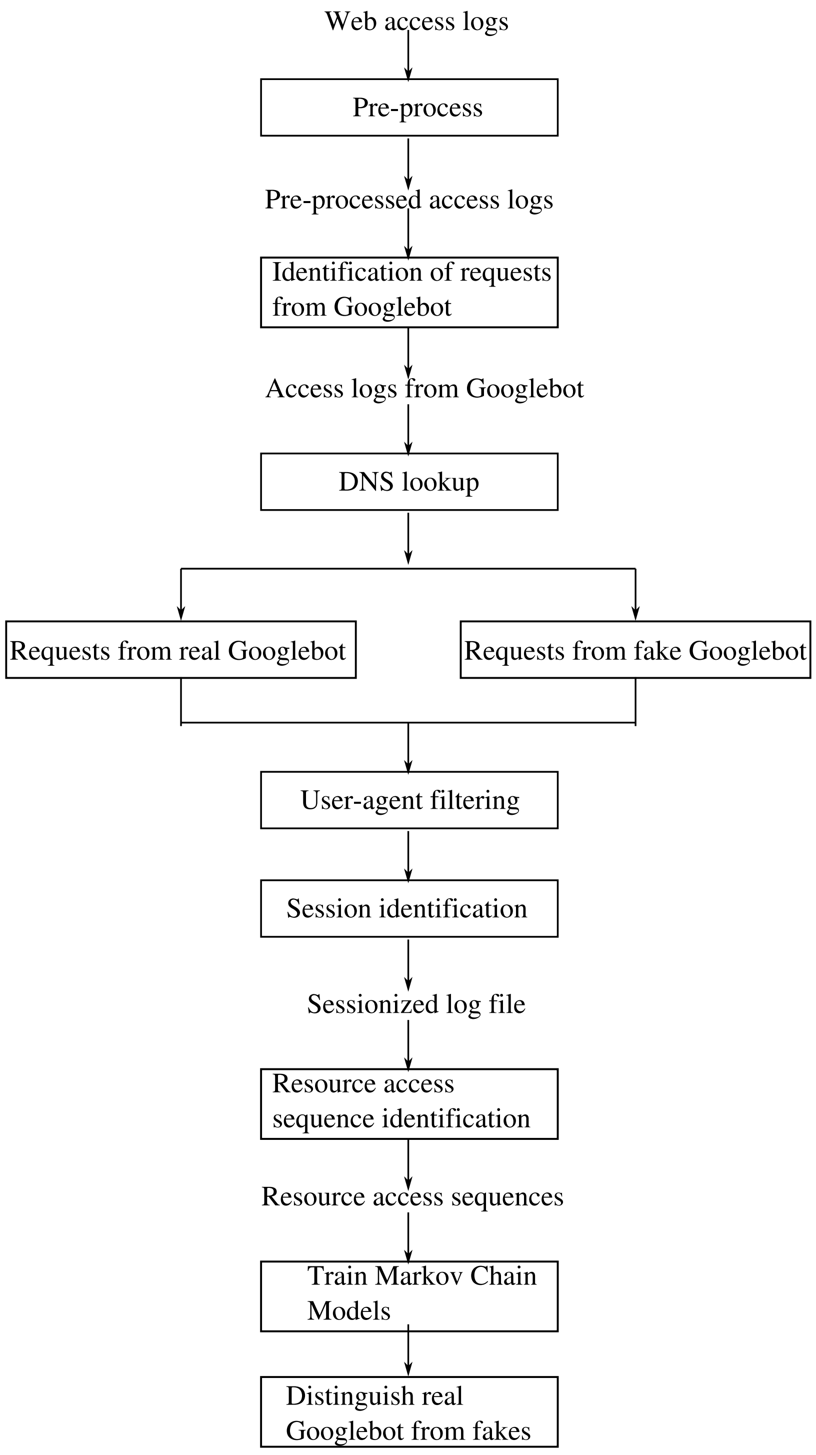

Web access logs
Pre-process
Pre-processed access logs
Identification of requests from Googlebot
Access logs from Googlebot
DNS lookup
Requests from real Googlebot
Requests from fake Googlebot
User-agent filtering
Session identification
Sessionized log file
Resource access sequence identification
Resource access sequences
Train Markov Chain Models
Distinguish real Googlebot from fakes

Figure 3.6: Methodology for fake Googlebot detection.

Table 3.2: Real Googlebot “user-agent“ and %hitcount

“user-agent“ string	%hitcount
DoCoMo/2.0 N905i(c100;TB;W24H16) (compatible; Googlebot-Mobile/2.1; +http://www.google.com/bot.html)	4.15%
Googlebot-Image/1.0	6.61%
Googlebot/2.1 (+http://www.google.com/bot.html)	0.19%
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	58.91%
Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25 (compatible; Googlebot-Mobile/2.1; +http://www.google.com/bot.html)	24.19%
SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0 Configuration/CLDC-1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0 (compatible; Googlebot-Mobile/2.1; +http://www.google.com/bot.html)	1.92%
Mozilla/5.0 (iPhone; CPU iPhone OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5376e Safari/8536.25 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	3.99%

3.2.1.4 “user-agent“ Analysis

Googlebot visits web sites with multiple “user-agent“ strings [47]. These crawlers are designed for different purposes. For e.g., "Googlebot-Image/1.0" for image file indexing and "Googlebot-Video/1.0" for video file indexing etc. and hence the resource request patterns are unique based on the purpose of the crawler. In our experiment to remove unnecessary confusions we wanted to limit to one specific “user-agent“. According to the Tables 3.2 and 3.3 high proportion of the hit counts resulting 58.91% and 87.81% were from “user-agent“ "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)". We considered only access patterns of this “user-agent“ string.

3.2.1.5 Session Identification

We have used IP address, “user-agent“ string and session timeout period in order to identify user sessions following the methodology proposed by Mobasher et al. [41]. In his work, this time period has been considered as 1800 seconds. But in our study we used 3600 seconds (60mins) and if the interval is more than the defined period, existing session was closed and new one was initiated. Following are the details of the sessions identified. We doubled the session time out period, since the length of sequences of some fake crawlers were very short during 30mins. Following are the details of the sessions identified.

- Total number of real Googlebot sessions : 1148

Table 3.3: Fake Googlebot “user-agent“ and %hitcount

“user-agent“ string	%hitcount
Googlebot (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	0.58%
Googlebot-Mobile/2.1	0.19%
Googlebot/2.1 (+http://www.google.com/bot.html)	0.38%
Googlebot/2.1 (+http://www.googlebot.com/bot.html)	10.25%
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	87.81%
Mozilla/5.0 (compatible; Googlebot/2.1;+http://www.google.com/bot.html)	0.77%

- Total number of fake Googlebot sessions : 365

3.2.1.6 Resource Type Identification

We have grouped the resource extensions available in a web site into eight classes. For example ’.png’,’.gif’,’.jpeg’,’.jpg’ extensions are grouped into *img* class and ’.pdf’,’.ppt’, ’.doc’ extensions are into *doc* class. Details of extension categorization is depicted in Table 3.4. These are the eight states of the Markov chain model.

Table 3.4: Resource Classes

Class	Extension
web	html,htm,php,jsp,cgi,js,css
img	gif,png,jpg,jpeg
doc	doc, ppt, pdf, ps,xls,odp
comp	zip, rar, gzip, tar, gz, 7z
robot	Requests for âĂIJrobots.txtâĂİ
favicon	Requests for âĂIJfavicon.icoâĂİ
root	Requests for root directory
no	Requests for directories other than root

These resource classes can be shown as a resource request pattern diagram (Fig 3.7).

3.2.1.7 Resource Access Sequence Identification

We have identified resource access sequence of real Googlebot and fake Googlebot period of over two moths in our training data set. This data is used in two Markov chain models for the real and fake one. A session-resource access sequence matrix for

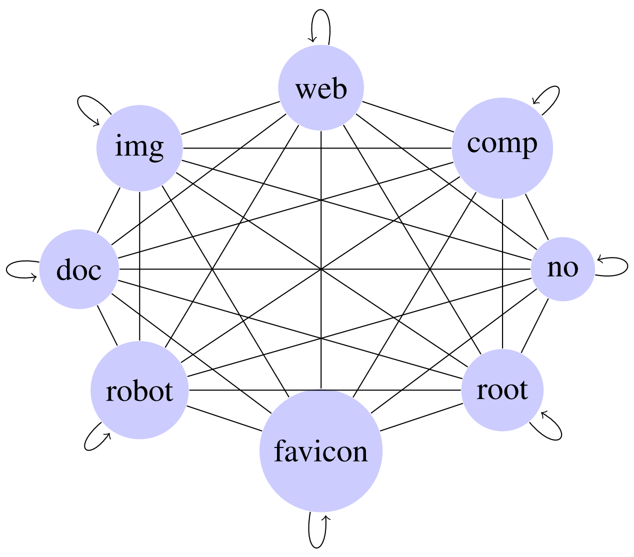


Figure 3.7: Resource request pattern diagram

both real and fake Googlebot sessions as in Table 3.5 is created to calculate log-odds ratios.

Table 3.5: Resource access sequence matrix

session	Resource access sequence
1	robots,root,web,web,web,web,web
2	web,robots,web,web,web,web,web
3	robots,root,web,web,doc,doc,doc
..	root,root,web,web,web,web,web

3.2.2 Summary

Web crawlers are programs or automated scripts that scan web pages methodically to create indexes. Search engines such as Google, Bing use crawlers in order to provide web surfers with relevant information. Today there are also many crawlers that impersonate well-known web crawlers. For example, it has been observed that Google's Googlebot crawler is impersonated to a high degree. This raises ethical and security concerns as they can potentially be used for malicious purposes. In this research, we present an effective methodology to detect fake Googlebot crawlers by analyzing web access logs.

3.3 Human Visitor Profiling

Visitors of web site (web clients) can be either software programs or human users. There is a huge difference between the patterns of web usage by these two groups. In Section 3.1 we discussed the methodology for detecting patterns of web crawlers which are automated scripts. In this section we discuss the proposed methodology for human visitor profiling.

Similarity measures have been proposed by many researchers in order to calculate similarity between the browsing patterns. For example Jitian et al. [36] talks about four different measures to calculate pairwise similarity between two sessions. Velásquez et al. [37] have proposed a new similarity measure based on the content of the respective web pages and the similarity between different page sequences. They have applied it in a Self Organizing Feature Maps(SOFM) clustering for user session clustering. How ever we see some drawbacks in the proposed similarity measure.

They have derived the measurement in the following way. Let α and β be two visitor behaviour vectors of cardinality C^{α} and C^{β} respectively and $\Gamma(.)$ is a function that applied over α or β that returns the navigation sequence corresponding to a visitor vector.

$$sm(\alpha,\beta) = dG(\Gamma(\alpha),\Gamma(\beta))\frac{1}{\eta}\sum_{k=1}^{\eta}\tau_k * dp(p_{\alpha,k},p_{\beta,k}) \tag{3.9}$$

where dG is the similarity between sequences of pages visited. $\eta = min\{C^{\alpha},C^{\beta}\}$ and τ_k is an indicator of visitor's interest of the web page visited. The term τ_k is defined as $\tau_k = min\{\frac{t_{\alpha,k}}{t_{\beta,k}}\frac{t_{\beta,k}}{t_{\alpha,k}}\}$. The term $dp(p_{\alpha,k},p_{\beta,k})$ is the similarity of pages visited. This is the angle cosine similarity between two word-page vectors.

For the term τ_k they have considered that the time spent on web page is proportional to the interest the visitor has in its contents. If the times spent by visitor α and β on k^{th} page that they have visited are close to each other the value(τ_k) becomes close to 1 and otherwise it will be close to 0.

The problem of this approach for τ_k is that time is not normalized. For example the value will be 0.5 for following two cases. But there is a huge difference of interest between the times.

$$t_k^{\alpha} = 2$$
$$t_k^{\beta} = 4$$
$$T_k = 2/4 = 0.5$$

$$t_k^{\alpha} = 50$$
$$t_k^{\beta} = 100$$
$$T_k = 50/100 = 0.5$$

We propose a better improved similarity measure to eliminate the problems in Velásquez et al. [37] approach.

For the purpose of human visitor profiling multiple steps have to be taken before applying web mining techniques. These techniques are discussed in the following subsections. The overall methodology of human visitor profiling is shown in Figure 3.8.

3.3.1 Data Preparation

Data preparation for the human visitor profiling also requires the steps of data cleansing, data formatting and user session identification. These steps are discussed in detail previously. (Section 3.1.1)

3.3.1.1 Remove web-crawler sessions

For the purpose of user-profiling we are interested only on requests from human users. Hence we have followed the methodology proposed by Algiriyage N. et al.[26] to detect web crawler sessions and then remove them.

3.3.1.2 Filter Low Click Pages Sessions

The number of web pages browsed during a user-session had a large variation. For the purpose of profiling we considered only the sessions having browsed more than three pages.

3.3.1.3 Filter Image and Styling Files

For the purpose of human visitor profiling we were not interested in image files. Hence we removed following extensions (GIF,cache,css,png,gif,jpeg,.js,jpg,ico,axd,ashx,xml) and considered only web page requests (asp,aspx,htm,html,php,pdf,doc).

3.3.1.4 Navigation Sequence Identification

A web site with 10 web pages is shown in the Fig 3.9.
Suppose in two visitor sessions users visit the web site in the following way.

$$S1 = 1 \rightarrow 2, 2 \rightarrow 7, 2 \rightarrow 5, 5 \rightarrow 6$$

$$S2 = 1 \rightarrow 3, 3 \rightarrow 7, 7 \rightarrow 9$$

Based on the traversal patterns in two sessions we can derive the navigation sequences as:

$$Seq(S1) = [1, 2, 7, 5, 6]$$
$$Seq(S2) = [1, 3, 7, 9]$$

Cardinality(C) of the two sequences can be given as:

$$C(S1) = 5$$

$$C(S2) = 4$$

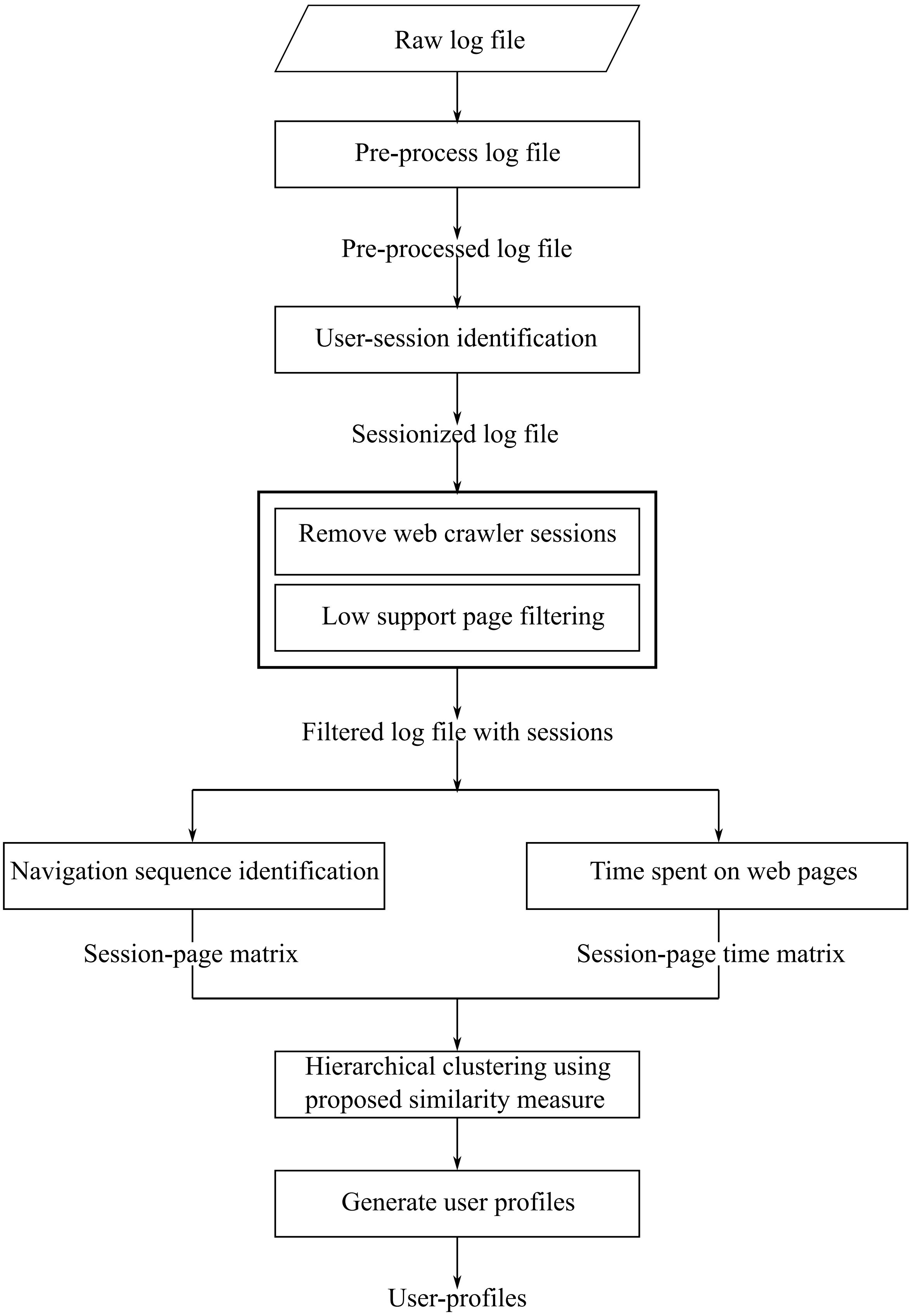

Raw log file
Pre-process log file
Pre-processed log file
User-session identification
Sessionized log file
Remove web crawler sessions
Low support page filtering
Filtered log file with sessions
Navigation sequence identification
Time spent on web pages
Session-page matrix
Session-page time matrix
Hierarchical clustering using proposed similarity measure
Generate user profiles
User-profiles

Figure 3.8: Methodology for human visitor profiling.

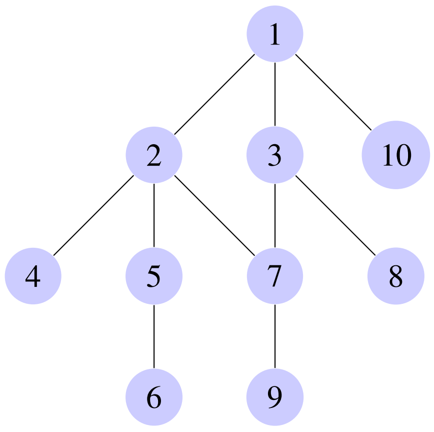


Figure 3.9: Web site with navigational paths

3.3.1.5 Prepare Session-page Matrix

Once the navigation sequence is identified we prepared the session-page matrix to show the web page sequences visited in each session. (Table 3.6). Our methodology for session identification is grouping IP address, “user-agent“ within 30 minute time period. Due to this methodology there can be some sessions from the middle of their activities. To avoid the confusions of such sessions we considered only the sessions starting with “sign-in“ and “home pages“ in the test data set. Further in the session-page matrix, we did Scipy encoding [48] to map string page names into integer ones.

Table 3.6: Session-page matrix

Session	Encoded page access sequence
1	6,61,70,76,101,110,112,134,161
2	6,61,70,72,76,81,112,136,143
3	6,61,101,102,112,161,162
...	6,61,70,73,76,101,110,112,161,168
n	8,28,29,30,35,43,44,45,46,49,59,61,114,121

3.3.1.6 Prepare session-time matrix

Session-page matrix was prepared to show the web pages browsed in each session. Another matrix is prepared to show the time spent on each page within the session. (Table 3.7).

3.3.1.7 Calculate Lavenshtein Distance

To compare the similarity between two sequences we need to use a dissimilarity measure. Lavenshtein distance [37] which is also referred as edit distance measures the similarity between two strings.

Table 3.7: Session-time Matrix

Session	p1,p2,p3,p4,p5,p6,p7
1	0.5,1.0,2.1,1.2,2.0,1.5,0.7
2	1.0,1.2,1.5,0.0,0.0,0.0,0.0
3	2.3,1.0,1.6,2.3,0.0,0.0,0.0
..	0.9,2.0,2.4,2.5,2.0,1.5,0.0
n	0.0,0.0,0.9,1.4,0.0,0.0,0.0

For two sequences $a = (a_1, ..., a_x)$ *and* $b = (b_1, ..., b_y)$ Lavenshtein distance is defined as:

$$LD(a_x, b_y) = \begin{cases} max(x,y) & if\ min(x,y) = 0 \\ min \begin{cases} L(a,b)(x-1,y)+1 \\ L(a,b)(x,y-1,j)+1 \\ L(a,b)(x-1,y-1)+1(a_x \neq b_y) \end{cases} & Otherwise \end{cases} \tag{3.10}$$

For example for the two sequences that we discussed earlier Seq(S1) = [1,2,7,5,6] and Seq(S2) = [1,3,7,9] the Lavenshtein distance is 3. That is 3 insert/update/delete operations are required to transform [1,2,7,5,6] to [1,3,7,9]. We calculated the similarity between the two visiting sequences using Equation 3.11:

$$dLD = sim(S1,S2) = 1 - \{(LD(seq(S1), seq(S2)))/max(C(S1), C(S2))\} \tag{3.11}$$

For the discussed example, similarity of the sequences is 0.4 based on the Equation 3.11. We refer this similarity which is based on Lavenshtein distance as *dLD*.

3.3.2 Comparison of Visitor Sessions

To compare the visitor sessions a similarity measure is required. The following subsections describe our approach in formulating the measure based on navigational sequence and time spent on web pages.

3.3.2.1 Similarity Between Navigational Sequences

Let S_x and S_y be two visitor sessions and F is a function applied over S_x and S_y which returns navigation sequences. Cardinality of the two sequences are C_x and C_y.The distance between two sequences is calculated using Lavenstine distance based measure described in Equation 3.11.

3.3.2.2 Similarity Based on Time Spent

Web visitors spend time on web pages based on the interest and relative importance of the content to them. To get a proper understanding we normalized the time value using standard score (Z-score). Preparation of session-time matrix was described in an earlier section (Table 3.7).

Suppose time spent on page n is t_n and the mean of time spent on page n by all visitors is μ and standard deviation is σ.

$$z = \frac{t_n - \mu}{\sigma} \tag{3.12}$$

Then we calculate the euclidian distance between the Z-values.

$$ED(z_x, z_y) = \sum_{i}^{n} (z_{xi} - z_{yi})^2 \tag{3.13}$$

We refer this similarity which is based on Euclidian distance as *dED*.

3.3.2.3 Proposed Similarity Measure

We propose a similarity measure between two visitor sessions based on the web page access sequence and time spent on each page. Let s_α and s_β are two visitor sessions and S_α and S_β are the related resource navigational sequences.

$$sim(s_\alpha, s_\beta) = dLD(S_\alpha, S_\beta) * dED(z_x, z_y) \tag{3.14}$$

Formulation of *dLD* and *dED* is discussed in Equations 3.11 and 3.13 respectively.

3.3.2.4 Hierarchical Clustering

Hierarchical clustering [49] builds a hierarchy of clusters decomposing a given set of data objects. Generally there are two types of hierarchical clustering based on the way of hierarchy is formed.

- Agglomerative : Agglomerative which is also known as “bottom-up“ merges the objects or groups that are close to one another, until all of the groups are merged into one or until the termination condition is met.

- Divisive : Which is also known as “top-down“ approach starts with all of the objects in the same cluster and in each iteration a cluster is split up to smaller clusters until there is one object in each cluster or termination condition is met.

We used the proposed similarity measure in a agglomerative hierarchical clustering [50] visualized it in a dendrogram. Dendrogram[49] is a tree structure which is used to represent the results of a hierarchical clustering. It shows how objects are grouped in step by step.

3.3.2.5 Visitor Profiling

Frequent pattern mining was used to profile the behaviour in each identified cluster. Frequently accessed web pages were considered in the process of profiling. Details of the application is discussed in Experimental Evaluation and Discussion in Chapter 4.

3.3.3 Summary

We have improved the proposed similarity measure by Velásquez et al. [37] based on the navigation sequence and time spent on each page. We have used *dLD*, a Lavenshtein distance based measure and *dED*, an Euclidian distance based measure. Time spent on each web page is normalized using Z-score normalization. This similarity measure is used in an unsupervised hierarchical clustering in order to identify visitor groups with similar interests. Finally profiles are generated using frequent item set mining.

4 Experimental Evaluation and Discussion

This chapter presents the experimental setups for methodology evaluation and discuss the results in detail. We first discuss the experimental details of the web crawler characterization in Section 4.1. In Section 4.2, experimental results of Googlebot impersonation and in Section 4.3 results of web visitor profiling are discussed.

4.1 Web Crawler Identification and Characterization

4.1.1 Data Set

Data set for our study was obtained by collecting web server access log files from a commercial web site in Sri Lanka. First we analysed a data set of more than two months from 04/Dec/2013 to 11/Feb/2014. The size of the pre-processed log file is 405.8 MB and it contained 1,633,578 HTTP requests. Log file contained 79,621 visitor sessions and among them we identified 16,198 sessions (20.43% of all visitor sessions) as possible crawler sessions. Table 4.8 shows the summary of the log-file.

Table 4.1: Summary of the log file.

Period of the log file	69 Days
Size of the log file	405.8 MB
Number of HTTP requests before pre-processing	1,633,913
Number of HTTP requests after pre-processing	1,633,578
Total number of visitor sessions	79,621
Number of possible web crawler sessions	16,198

Figure 4.1 shows web traffic in our log file using HTTP requests per day from 04/Dec/2013 to 18/Jan/2014. In the following sections we discuss some analysis of the identified possible web crawlers.

As we have described in Chapter 2 crawlers use “user-agent“ to put their identifying information. But there are some crawlers that use browser “user-agent“s. We used a “user-agent“ parser to identify the web crawlers who visited our web site. Table 4.2 lists the identified web crawler names.

Further we analysed countries generating web crawlers. The bar graph in Figure 4.2 shows the topmost countries generating these crawlers. According to the results United States and China have generated most of the web crawlers who visited our web site.

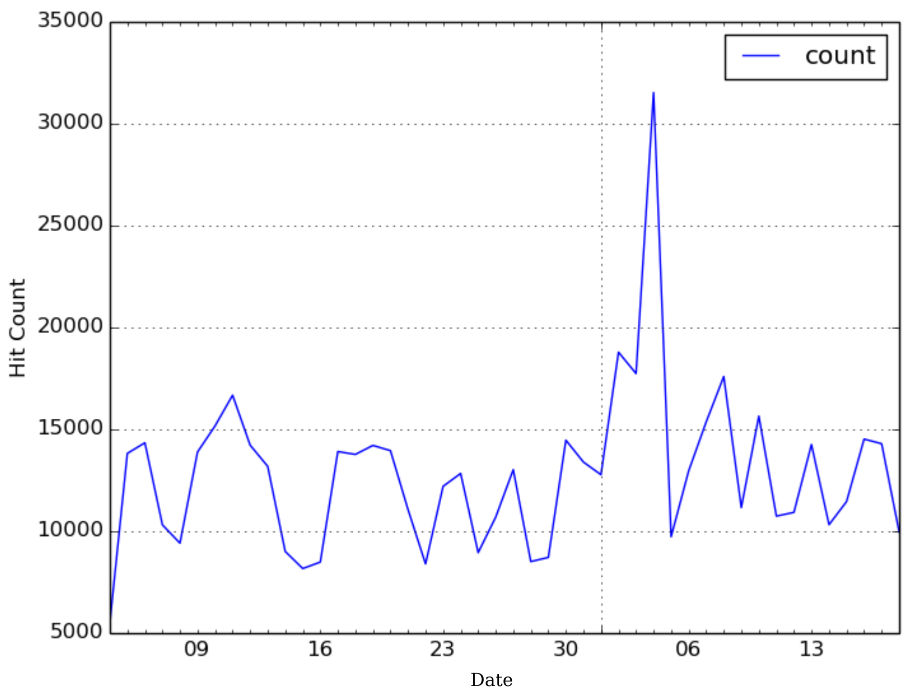


Figure 4.1: Number of HTTP requests per day.

In our test data set there were altogether 16,198 web crawler sessions. This comprises of multiple sessions from the same crawler. We analysed the total number of sessions generated by a particular crawler. According to the results in Figure 4.3 Baiduspider/2.0 has generated the most sessions.

4.1.2 Methodology Evaluation

We conducted several experiments to check the accuracy of our crawler detection methodology on the same web site. In this attempt we didn't remove the traffic generated by other sources from the data set for testing. Several common web crawling tools were used with and without modifications to conduct experiments which are listed below. We have crawled the site using seven different methods including simple crawling patterns to more advanced crawling. These individual tests are listed below.

- Crawler Scenario 01: Use the crawling tool without any modifications

- Crawler Scenario 02: Use crawling tool with a modified "user-agent" of a common browser

Table 4.2: Web crawlers found in the dataset.

bingbot/2.0	YandexBot/3.0	YandexImages/3.0
linkdexbot/2.0	Googlebot/2.1	archive.org_bot
proximic	AhrefsBot/5.0	SiteExplorer/1.0b
Baiduspider/2.0	Yeti/1.0	SISTRIX Crawler
Butterfly/1.0	BLEXBot/1.0	elefent/Elefent 1.2
Abonti/0.91	yacybot	GrapeshotCrawler/2.0
Twitterbot/1.0	GeliyooBot/1.0	Mail.RU_Bot/2.0
Exabot/3.0	coccoc/1.0	CompSpyBot/1.0
PiplBot	Ezooms/1.0	Googlebot-Mobile/2.1
MJ12bot/v1.4.4	TweetmemeBot/3.0	TweetedTimes Bot/1.0
PaperLiBot/2.1	SemrushBot/0.97	KomodiaBot/1.0
SeznamBot/3.2	BLEXBot/1.0	CloudServerMarketSpider/1.0
MojeekBot/0.6	CompSpyBot/1.0	BIXOCRAWLER
Blekkobot	aiHitBot/2.8	emefgebot/beta
uMBot-FC/1.0	oBot/2.3.1	PHPDevelBot/1.0
Prlog/1.0	MJ12bot/v1.4.4	URLAppendBot/1.0
BeetleBot	LoadTimeBot/0.9	socialbm_bot/1.0
200PleaseBot/1.0	news bot /2.1	TweetedTimes Bot/1.0
WASALive-Bot	Vagabondo/4.0	LoadTimeBot/0.81
CrawlBot/1.0.0	SEOkicks-Robot	BuiltWith.org/0.1
YRSpider	Webidex Bot/1.2	SearchmetricsBot

- Crawler Scenario 03: Use crawling tool with a fixed time delay of 20 seconds

- Crawler Scenario 04: Use crawling tool with a modified “user-agent“ of a common browser, assigning a fake referrer for all HTTP requests, with a random time delay

- Crawler Scenario 05: Use crawling tool ignoring “robots.txt“, with a modified “user-agent“ of a common browser, assigning a fake referrer for all HTTP requests, with a random time delay

- Crawler Scenario 06: Use crawling tool ignoring “robots.txt“, with a modified “user-agent“ of a known crawler, assigning a fake referrer for all HTTP requests, with a random time delay

- Crawler Scenario 07: Use crawling tool ignoring “robots.txt“, with modified random “user-agent“s, assigning a referrer for all HTTP requests with random fake referrers, and also random time delay between each of requests.

These crawler scenarios are summarized in Table 4.3.

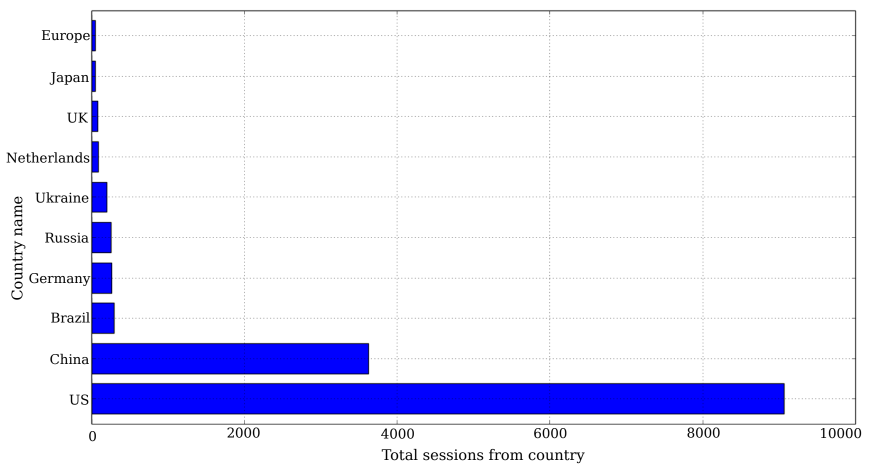


Figure 4.2: Top most countries generating web crawlers.

4.1.3 Experimental Results

Our results show that all the seven crawler scenarios have been identified as possible crawlers and out of them three crawler scenarios (5, 6 and 7) were categorized into “suspicious“ crawler category and four crawler scenarios (1, 2, 3 and 4) were classified into the “other“ crawler category. Since we didn’t remove the traffic generated by other sources in our test data set, it consisted of both experimental crawling results(crawler scenarios) and other crawlers’ data. We analyzed these crawler attempts in web server log to determine the behavior patterns of various crawlers. 6.23% of the total sessions identified as web crawler sessions in the web server log file, which also includes our test scenarios. Among them 53.25% of the crawler sessions were “known“ and 34.16% were categorized as “suspicious“ while 12.49% as “other“ crawler sessions (Table 4.4).

Hidden links, the novel technique that we have introduced in this paper can be considered as a better way trap web crawlers since 8.25% of the web crawlers have ac-

Table 4.3: Summary of crawler scenarios.

Description	Crawler Scenario						
Use default “user-agent“	1		3				
Ignore “robots.txt“					5	6	7
Modified “user-agent“ of a known crawler						6	
Modified “user-agent“ of a browser		2		4	5		
Use random “user-agent“s							7
Assigned referrer field				4	5	6	7
Use random referrer							7
Fixed time delay			3				7
Random time delay				4	5	6	7

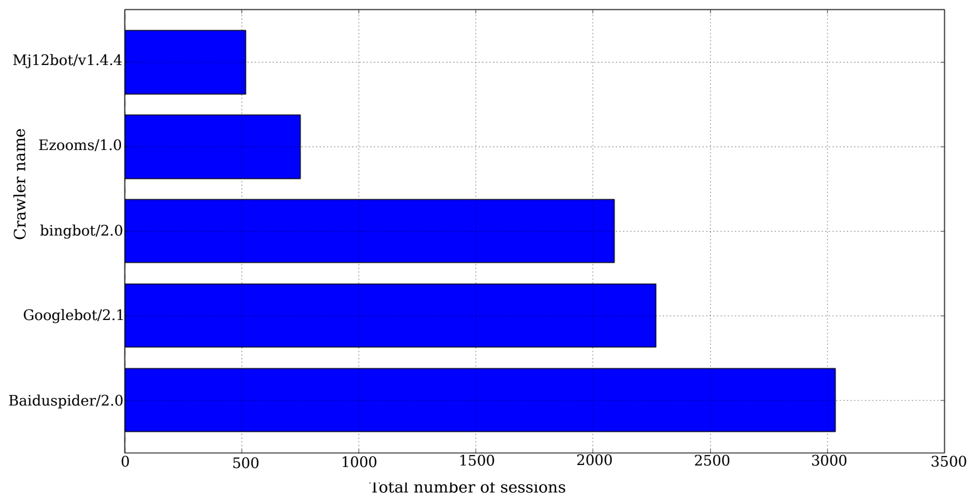


Figure 4.3: Crawlers by total number of sessions generated.

Table 4.4: Summary of crawler sessions.

Crawler Analysis	Percentage (%)
% of âĂIJknownâĂİ crawler sessions	53.25%
% of âĂIJsuspiciousâĂİ crawler sessions	34.16%
% of âĂIJotherâĂİ crawlers sessions	12.49%

cessed it. Classification results of “known“, “suspicious“ and “other“ categories are elaborated in Tables 4.4, 4.5 and 4.6.

As stated in Table 4.5 all “known“ crawlers have accessed “robots.txt“ in 96.71% of the sessions. The reason behind the remaining 03.29% is the usage of default session time out period which is described below.

According to the algorithm used for session generation, a single crawler can be included in multiple sessions (since we considered a 30 minute session timeout period). In such cases the crawler has accessed “robots.txt“ in one session but not in the others. Further, none of “known“ crawler sessions have accessed hidden link type III which was inside an html comment.

Table 4.5: “Known“ crawler patterns.

Crawler Analysis	Percentage (%)
% of crawler sessions accessed âĂIJrobots.txtâĂİ	96.71%
% of crawler sessions accessed hidden link type I & II	03.29%
% of crawler sessions accessed hidden link type III	00.00%

Table 4.6: “Suspicious“ crawler patterns.

Crawler Analysis	Percentage (%)
% of crawler sessions avoided “robots.txt“	20.24%
% of crawler sessions with forged user agent of a known crawler	00.91%
% of a crawler sessions without a user agent in the HTTP request	03.02%
% of crawler sessions accessed hidden link type III	11.48%
% of blacklisted crawler sessions	75.53%

Table 4.7: “Other“ crawler patterns.

Crawler Analysis	Percentage (%)
% of crawler sessions accessed âĂIJrobots.txtâĂİ	66.12%
% of crawler sessions accessed hidden link type I & II	09.92%

Table 4.7 lists “other“ crawler behavior patterns. We did a manual verification for the other crawler list and filtered out all the sessions that didn’t access “robots.txt“ since it violates robots exclusion protocol.

4.1.4 Crawler-Trap Tool

We developed a web based tool using the methodology proposed for web crawler detection and characterization. This tool enables web users to upload log files and check the crawlers visited. Details of the Crawler-Trap tool is included in the Appendix B.

Tables A.2, A.3 and A.4 in Appendix A show few examples for identified “known“, “suspicious“ and “other“ crawlers.

4.2 Googlebot Impersonation

As described in the methodology section, we collected web server access log files from an e-commerce web site for a period of about 10 weeks. Size of the data set was 409.2MB and contained 1,633,913 log entries before pre-processing. We considered 75% of the data set as training data and 25% as test data. Further our methodology was applied to another dataset from an academic web site.

Table 4.8: Summary of the data set.

Period of the log file	69 Days
Size of the log file	409.2 MB
Number of HTTP requests before pre-processing	1,633,913
Number of HTTP requests after pre-processing	1,633,578

4.2.1 Patterns of the Dataset

We found that 5.32% of all Googlebot sessions on that site were fake. We observed multiple instances of php remote code execution vulnerability [?] scans by these fake Googlebots in our data set. Table 4.9 shows the statistics of countries that generate fake Googlebot traffic on our site. According to the analysis, the legitimate real Googlebot always visited only from the United States and most fake Googlebots were visited from Brazil.

Table 4.9: Countries originating fake-Googlebot Academic web site

Country	%Requests
Brazil	48.22
United States	37.54
China	1.95
Japan	1.77
not found	1.42
Mexico	1.06
Turkey	1.06
Saudi Arabia	0.88
Portugal	0.88
Vietnam	0.88
United Kingdom	0.88
Thailand	0.71
Sweden	0.53
Singapore	0.35
India	0.35
Germany	0.35
All Others	1.06

Table 4.9 shows the statistics of countries generating fake Googlebot. In our log files real Googlebot visited only from United States and most of the times fake Googlebots were visited from Brazil.

In the Figures 4.4 and 4.5 show the HTTP requests per day by both real and fake Googlebots in our test data set.

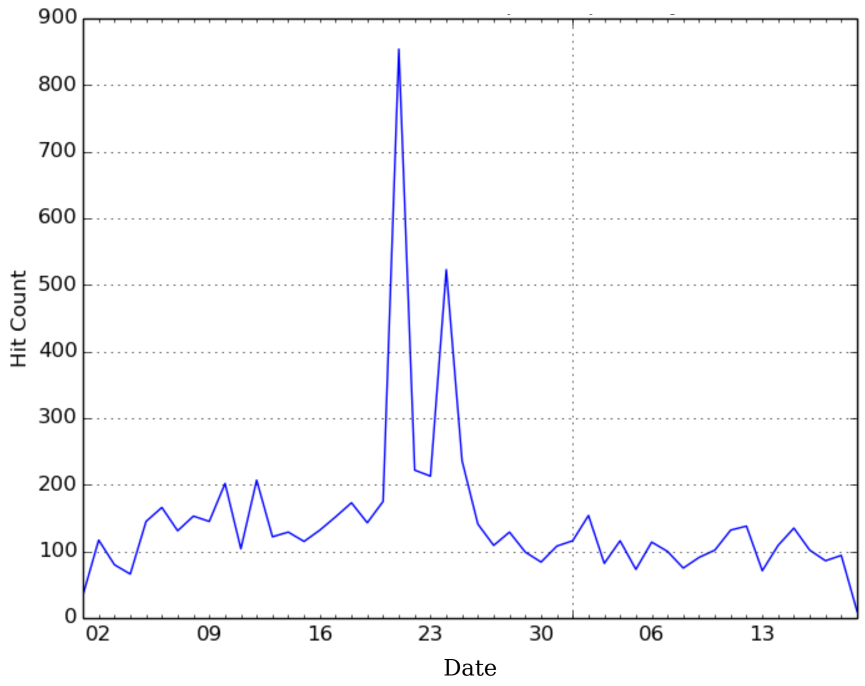


Figure 4.4: HTTP requests per day for real-Googlebot

4.2.2 Markov Chain Models

To detect the resource request patterns of fake Googlebot we needed to build long term profiles of the two types Googlebots in order to differentiate them. Our data set contained more than two months old access log data from an e-commerce web site. We considered 75% of the data as training set and remaining portion for testing. As explained in the section 3 we identified resource request patterns for the fake and real Googlebot and trained two Markov chain models for the long term access patterns of the real Googlebot and fake Googlebot. We have applied Laplacian smoothing for transition matrix to avoid zero probabilities. Further we have assigned a small probability value of 0.00000000001 (IE-10) to state transitions which did not appear in the training data.

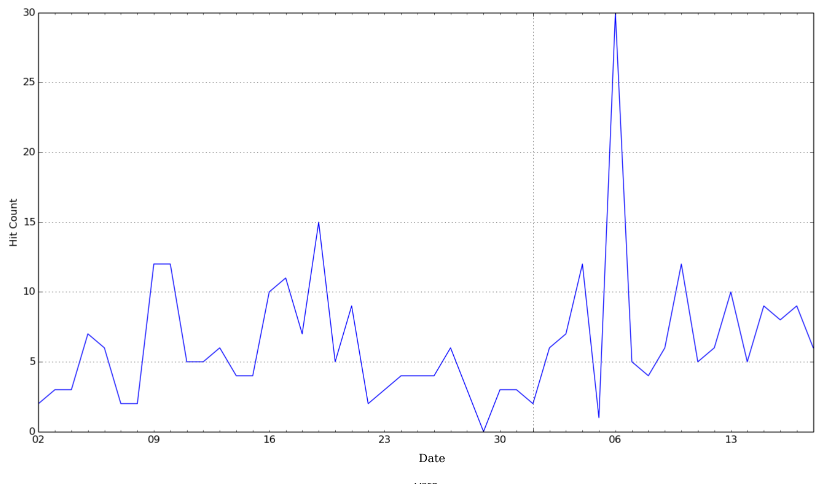


Figure 4.5: HTTP requests per day for fake-Googlebot

The following is the generated transition matrix with probabilities.

```
5.7142857e-01 6.4516129e-03 7.5187969e-03 4.4843049e-03 4.1093075e-04 1.5384615e-02
7.1428571e-02 2.7096774e-01 2.2556391e-02 2.6905829e-02 2.0752003e-02 3.0769230e-02
7.1428571e-02 3.2258064e-02 6.7669172e-02 1.1210762e-01 1.8902814e-02 1.5384615e-02
7.1428571e-02 3.8709677e-02 6.0150375e-02 1.3004484e-01 3.6572837e-02 1.5384615e-02
1.4285714e-01 6.3870967e-01 8.0451127e-01 7.0403587e-01 9.1514279e-01 7.3846153e-01
7.1428571e-02 1.2903225e-02 3.7593985e-02 2.2421524e-02 8.2186151e-03 1.8461538e-01
```

Then the session-resource access sequence matrix was used to calculate log-odds ratios for both types bots in the training data set. Figures 4.6 and 4.7 show the histogram of distribution patterns of log-odds ratios. According to the results a log-odds ratios are high for real Googlebot and low for fake Googlebot in both of web sites.

4.2.3 Accuracy Evaluation

We applied the trained Markov chain models to our test data to find the threshold score of log-odds ratio which differentiate fake Google bot from the real one. Before applying the Markov models we labeled our test data set using the forward and backward DNS lookup. To evaluate the accuracy of our methodology we used accuracy score value which can be defined as follows.

$$accuracy = \frac{number\ of\ true\ positives + number\ of\ true\ negatives}{number\ of\ true\ positives + false\ positives + false\ negatives + true\ negatives} \quad (4.1)$$

Table 4.10 shows the accuracy values with the log-odds ratio for our test data set. The

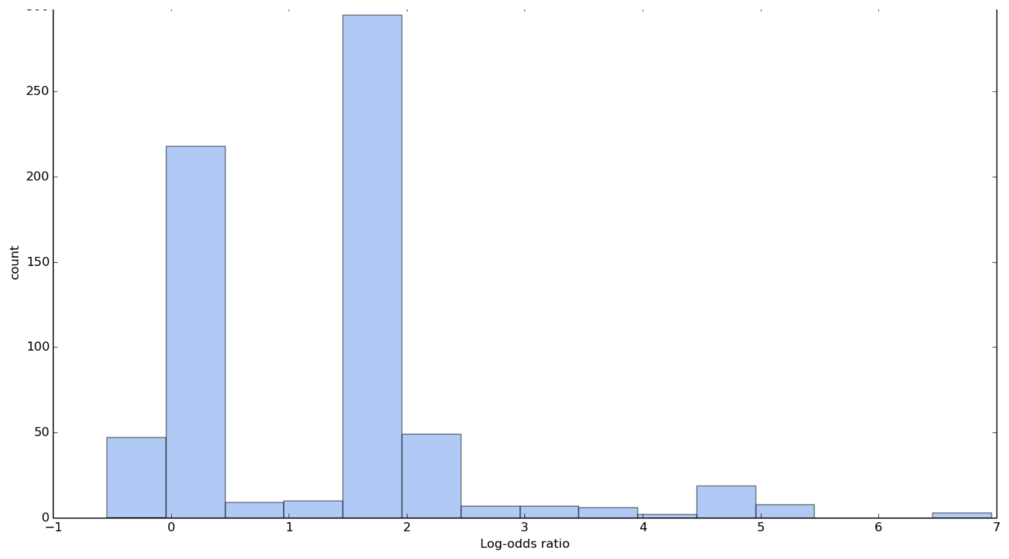


Figure 4.6: Log-odds ratio real Googlebot

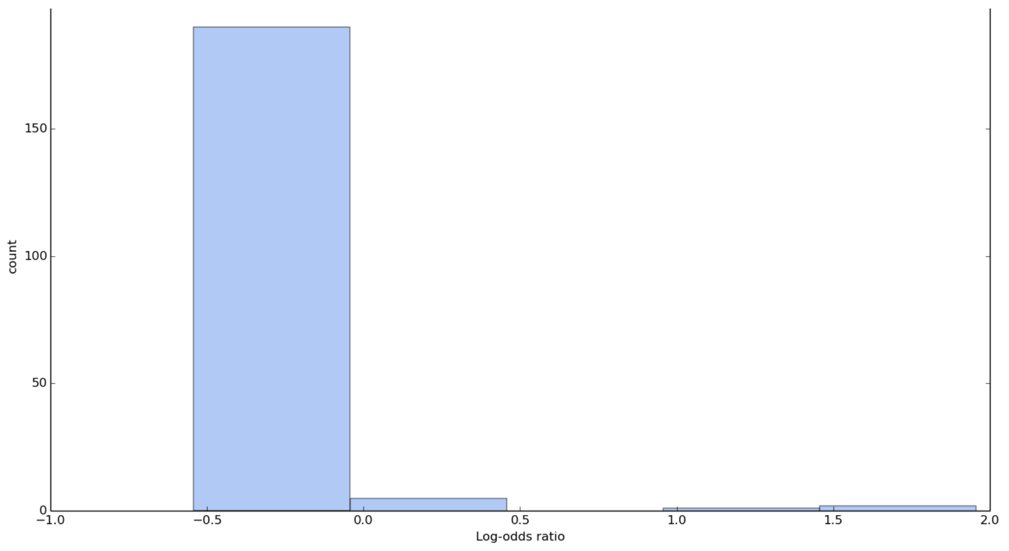


Figure 4.7: Log-odds ratio fake Googlebot

highest score for all the measures have occurred at the log-odds ratio of -0.4. Hence we can differentiate a given sequence of resource access patterns as from fake if the log-odds ratio is less than the threshold value of -0.4.

Table 4.10: Accuracy scores for different log-odds ratios (e-commerce web log)

Log-odds ratio	1.5	1.0	0.5	0.0	-0.1	-0.2	-0.3	-0.4	-0.5
Accuracy	0.673	0.738	0.690	0.936	0.940	0.946	0.951	0.956	0.739

We applied the same methodology for the data set from access logs taken from an academic web site. To train Markov chain models, a log file sized 252.9 MB containing 1,000,000 HTTP requests after pre-processing was considered. For the testing purposes, another log file taken from the same web site with 324,693 HTTP requests was considered. According to the results, the threshold score was 1.0 for the academic web site, with 0.96 accuracy.

4.3 Human Visitor Profiling

4.3.1 Data Preperation

4.3.1.1 Dataset

We obtained web server access log files of a e-commerce service provider in Sri Lanka for a period of two weeks. Size of the log file is 243.6MB and it contained 841,196 HTTP requests before the preprocessing stages. But for the hierarchical clustering we considered a portion of the data set containing 71,256 HTTP requests before and 71,238 requests after pre-processing. Summary of the data set is presented in Table 4.11.

Table 4.11: Summary of the log file.

Number of HTTP requests before pre-processing	71,256
Number of HTTP requests after pre-processing	71,238
Total number of visitor sessions	2,033
Number of possible web crawler sessions	185
Number of possible Human Visitor sessions	1,848
Number of visitor sessions with initiating sessions	1,212
Number of visitor sessions after low click page filtering	1,169

4.3.2 Hierarchical Clustering

We applied the proposed similarity measure in an agglomerative hierarchical clustering. Figure 4.8 show the dendrogram generated. Dendrogram is a tree-structured graph used to visualize the results of a hierarchical clustering. A dendrogram can be pruned at any level to generate clusters. In this experiment, we pruned the dendrogram at level 10 to obtain 10 visitor clusters.

4.3.3 User Profiling

Generating profiles, or gaining more insight to the clusters generated, require some additional work. We performed a frequent item set mining task in order to understand the behaviour patterns in each cluster. With the highest confidence and support values, following are the clusters generated in Table 4.12.

Table 4.12: Cluster results

Cluster	Pages Visited	Description
1	6,61,112	successfully logged-in, but nothing was done after the loging
1	8,16,24,34,...	successfully logged-in as corporate users, and performed transactions
3	6,61,70,76,...	successfully logged-in as personal users, and performed transactions
4	6,61,101,112,161,...	successfully logged-in as personal users, and performed transactions
5	6,61,70,72,...	Some strange behaviour
6	6,61,70,76,...	successfully logged-in as personal users, and performed transactions
7	8,28,43,44,...	successfully logged-in as corporate users, and performed transactions
8	6,61,70,76,...	Some strange and random behaviour
9	5,15,17,...	New users, who do not log-in to the web site
10	6,61,8,...	Some random browsing behaviour

To understand and describe the clusters, we logged-in to the e-commerce web site as different types of users and performed multiple transactions. Based on our browsing patterns in the access log file, generated clusters were described in the Table 4.12.

According to the clustering results, some users logged-in to the system and has perform nothing afterward and the others performed different types of transactions. In some clusters, there were random browsing behaviour which was hard to understand. In one cluster, which contained a single user session, the behavior was very strange that the visitor traversed a long path without successfully doing any transaction.

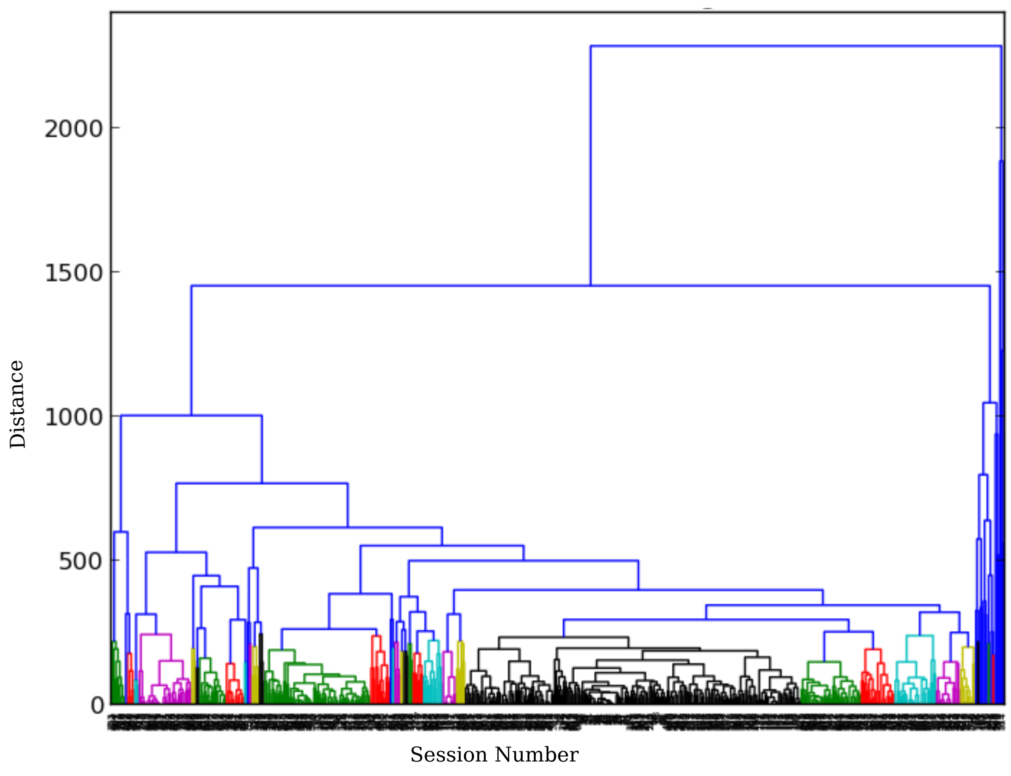


Figure 4.8: Results of hierarchical clustering.

5 CONCLUSIONS

This research focused on the patterns in web server access log files and the important knowledge that can be gained. We considered the access pattern of web clients web crawlers and human visitors.

5.1 Characterization of Web Crawlers

During the first phase our project we detected web crawler patterns from web server access log files. Our categorization of web crawlers included three types “Known“ crawlers, “Suspicious“ crawlers and “Other“ crawlers.

- Known web crawlers : These are the crawlers from well-known search engines, which help to drive more users to the web site and hence increase the reputation of the web site. On the other hand for e-commerce web sites these will help to generate more revenue.

- Suspicious web crawlers : These are the crawlers that may exploit vulnerabilities and potential threat for site integrity and availability.

- Other crawlers : The intent of these crawlers is hard to define. They may or may not be harmful.

Some additional behaviour of these crawlers are : “Known“ crawlers have obeyed RFC2616 and robots exclusion protocol while “suspicious“ and “other“ crawlers have violated these standards most of the times. “Suspicious“ crawlers have accessed Hidden link type III which was inside the html comment but no “Known“ crawler has accessed it.

Python script which implements our approach for web crawler detection and categorization is available as a googlecode project at `http://code.google.com/p/crawler-detection/`.

5.2 Detection of Googlebot Impersonation

Fake Googlebot (or Googlebot impersonation) is a new kind of a threat observed by many web site owners. In this work we evaluated the robustness of using a Markov

Chain model for the problem of differentiating fake and real Googlebot. Our approach was to learn the resource request patterns of web crawlers. As shown in Figures 4.6 and 4.7, there exists a clear gap between the log-odds ratios for the real and fake Googlebots. The higher the log-odds score, the higher the probability that a given sequence is from a real Googlebot. We have calculated a threshold score for our test data set to conclude the type of a given sequence. Further we have applied the methodology another new data set. Threshold score to differentiate fake Googlebot from real one varies from site to site. Our methodology can be easily applied for any web site since we are based on resource access sequences.

5.3 Human User Profiling

We improved the similarity measure proposed by Velásquez et al. [37] to group human users based on their browsing behavior. The similarity measure was included in a agglomerative hierarchical clustering algorithm to identify user clusters. Our results show that there are clear clusters among the visitors/users of the web site. Derived user profiles can be used by the e-commerce organization to get better knowledge about their customer base. And also this can be used as an intrusion detection tool, where we found some clusters having strange browsing patterns. Any way further research has to be carried out to understand how well this clustering helps in intrusion detection.

5.4 Future Improvements

5.4.1 Characterization of Web Crawlers

For the future works, web crawler detection methodology can be improved by using more features like HTTP errors, amount of bytes downloaded. The accuracy of the methodology has to be increased and minimize the rate of false positives. The approach of using hidden links can be further improved and use hidden images and text files.

Our categorization of web crawlers contained three classes "known","suspicious" and "other". The clusters are overlapping is some cases, where some IP address of the “known“ crawlers behave like “suspicious“ or “other“ vise versa. Hence further research has to be carried out, in order to clearly define boundaries between these categorization.

Finally our off-line web crawler detection and characterization methodology can be extended to real-time detection with some more research.

5.4.2 Detection of Googlebot Impersonation

As for the future works we can consider more advanced request patterns of Googlebots for improvements. We have applied the methodology on an e-commerce web server log. To get a better idea this can be applied to multiple web log files. Further research has to be carried out to implement our methodology in web servers to detect fake Googlebot requests real-time.

5.4.3 Human User Profiling

As future works we can consider the content of web pages and more advanced features for the proposed similarity measure. We have applied the methodology on online e-commerce web server logs. To get a better idea this can be applied in multiple web log files. In our approach the similarity measure is used in a agglomerative hierarchical clustering algorithm. This can be used other clustering algorithms and can test the generated user profiles for further improvements.

BIBLIOGRAPHY

[1] D. Doran, "Detection, classification, and workload analysis of web robots," 2014.

[2] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and pattern discovery on the world wide web," in *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, pp. 558–567, IEEE, 1997.

[3] P.-N. Tan and V. Kumar, "Discovery of web robot sessions based on their navigational patterns," in *Intelligent Technologies for Information Analysis*, pp. 193–222, Springer, 2004.

[4] "Google search engine." [Online]. Available:`https://www.google.com/`.

[5] "Yahoo search engine." [Online]. Available:`https://www.yahoo.com/`.

[6] "msn search engine." [Online]. Available:`http://www.msn.com/en-in/`.

[7] "bing search engine." [Online]. Available:`http://www.bing.com/`.

[8] "Four fake google haxbots hit your website every day." [Online]. Available:`http://www.theregister.co.uk/2014/07/25/four\_fake\_google\_haxbots\_hit\_your\_website\_every\_day/`.

[9] "Fake googlebot activity up 61%." [Online]. Available:`http://searchenginewatch.com/article/2358345/Fake-Googlebot-Activity-up-61-Report`.

[10] "Verifying googlebot." [Online]. Available:`https://support.google.com/webmasters/answer/80553?hl=en`.

[11] "Apache http server version 2.2." [Online]. Available:`http://httpd.apache.org/docs/2.2/logs.html`.

[12] "Gnu wget." [Online]. Available:`http://www.gnu.org/software/wget/`.

[13] "Httrack website copier." [Online]. Available:`http://www.httrack.com/`.

[14] "Rapidminer." [Online]. Available:`http://rapidminer.com/`.

[15] "Python." [Online]. Available:`https://www.python.org/`.

[16] "The perl programming language." [Online]. Available:https://www.perl.org/.

[17] "Java." [Online]. Available:`https://www.java.com/en/`.

[18] L. Destailleur, "Awstats official web site." `http://www.awstats.org/`, 2010.

[19] P. Huntington, D. Nicholas, and H. R. Jamali, "Web robot detection in the scholarly information environment," *Journal of Information Science*, vol. 34, no. 5, pp. 726–741, 2008.

[20] M. Koster, *A standard for robot exclusion*. NEXOR., 1994.

[21] D. Stevanovic, N. Vlajic, and A. An, "Detection of malicious and non-malicious website visitors using unsupervised neural network learning," *Applied Soft Computing*, vol. 13, no. 1, pp. 698–708, 2013.

[22] M. D. Dikaiakos, A. Stassopoulou, and L. Papageorgiou, "An investigation of web crawler behavior: characterization and metrics," *Computer Communications*, vol. 28, no. 8, pp. 880–897, 2005.

[23] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in CryptologyâĂŤEUROCRYPT 2003*, pp. 294–311, Springer, 2003.

[24] A. Balla, A. Stassopoulou, and M. D. Dikaiakos, "Real-time web crawler detection," in *Telecommunications (ICT), 2011 18th International Conference on*, pp. 428–432, IEEE, 2011.

[25] D. Stevanovic, N. Vlajic, and A. An, "Unsupervised clustering of web sessions to detect malicious and non-malicious website users," *Procedia Computer Science*, vol. 5, pp. 123–131, 2011.

[26] N. Algiriyage, S. Jayasena, G. Dias, A. Perera, and K. Dayananda, "Identification and characterization of crawlers through analysis of web logs," in *Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on*, pp. 150–155, Dec 2013.

[27] N. Ye *et al.*, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166, p. 169, West Point, NY, 2000.

[28] D. Zhang, D. Zhang, and X. Liu, "A novel malicious web crawler detector: Performance and evaluation," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 1, 2013.

[29] G. Stermsek, M. Strembeck, and G. Neumann, "A user profile derivation approach based on log-file analysis.," in *IKE*, vol. 2007, pp. 258–264, Citeseer, 2007.

[30] Y. Xie and V. V. Phoha, "Web user clustering from access log using belief function," in *Proceedings of the 1st international conference on Knowledge capture*, pp. 202–208, ACM, 2001.

[31] J. Xu and H. Liu, “Web user clustering analysis based on kmeans algorithm,” in *2010 International Conference on Information Networking and Automation (ICINA)*, vol. 2, pp. V2–6, IEEE, 2010.

[32] A. Sharma, A. Goel, P. Gulati, *et al.*, “A novel approach for clustering web user sessions using rst,” in *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT’09. International Conference on*, pp. 657–659, IEEE, 2009.

[33] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram, “Mining web access logs using relational competitive fuzzy clustering,” in *Proceedings of the Eight International Fuzzy Systems Association World Congress*, vol. 1, pp. 195–204, Citeseer, 1999.

[34] B. S. Suryavanshi, N. Shiri, and S. P. Mudur, “An efficient technique for mining usage profiles using relational fuzzy subtractive clustering,” in *Web Information Retrieval and Integration, 2005. WIRI’05. Proceedings. International Workshop on Challenges in*, pp. 23–29, IEEE, 2005.

[35] G. Castellano, F. Mesto, M. Minunno, and M. A. Torsello, “Web user profiling using fuzzy clustering,” in *Applications of Fuzzy Sets Theory*, pp. 94–101, Springer, 2007.

[36] J. Xiao, Y. Zhang, X. Jia, and T. Li, “Measuring similarity of interests for clustering web-users,” in *Proceedings of the 12th Australasian database conference*, pp. 107–114, IEEE Computer Society, 2001.

[37] J. D. Velásquez, H. Yasuda, and R. WEBER, “A new similarity measure to understand visitor behavior in a web site,” *IEICE TRANSACTIONS on Information and Systems*, vol. 87, no. 2, pp. 389–396, 2004.

[38] B. De Finetti and B. de Finetti, “Theory of probability, volume i,” *Bull. Amer. Math. Soc. 83 (1977), 94-97 DOI: http://dx. doi. org/10.1090/S0002-9904-1977-14188-8 PII*, pp. 0002–9904, 1977.

[39] “Computational biology.” [Online]. Available:`http://www.cs.hunter.cuny.edu/~saad/courses/compbio/lectures/lecture9.pdf`.

[40] S. Langhnoja, M. Barot, and D. Mehta, “Pre-processing: Procedure on web log file for web usage mining,” *International Journal for Emerging Technology and advanced enfineering*, vol. 2, no. 12, 2012.

[41] R. Cooley, B. Mobasher, and J. Srivastava, “Data preparation for mining world wide web browsing patterns,” *Knowledge and information systems*, vol. 1, no. 1, pp. 5–32, 1999.

[42] “Hypertext transfer protocol – http/1.1.” [Online]. Available:`https://www.ietf.org/rfc/rfc2616.txt`.

[43] “Spamhaus database.” [Online]. Available:http://www.spamhaus.org/zen/.

[44] “Abuseat database.” [Online]. Available:http://cbl.abuseat.org/.

[45] “Project honeypot database.” [Online]. Available:https://www.projecthoneypot.org/index.php.

[46] R. Durbin, *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[47] “Google crawlers.” [Online]. Available:https://support.google.com/webmasters/answer/1061943?hl=en.

[48] “Encoding categorical features.” [Online]. Available:http://scikit-learn.org/stable/modules/preprocessing.html.

[49] H. Jiawei and M. Kamber, “Data mining: concepts and techniques,” *San Francisco, CA, itd: Morgan Kaufmann*, vol. 5, 2001.

[50] “scipy.cluster.hierarchy.dendrogram.” [Online]. Available:http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.cluster.hierarchy.dendrogram.html.

A Web Crawler Identification & Characterization

Table A.1: Web crawlers with originated country.

coccoc/1.0	Vietnam
bingbot/2.0	United States
Ezooms/1.0	United States
Googlebot/2.1	United States
360Spider	China
Yahoo! Slurp	United States
SeznamBot/3.2	Czech Republic
archive.org_bot	United States
SISTRIX Crawler	Germany
NetSeer crawler/2.0	United States
GrapeshotCrawler/2.0	United Kingdom
BLEXBot/1.0	United States
MojeekBot/0.6	United Kingdom
CompSpyBot/1.0	United States
Butterfly/1.0	United States
Blekkobot	United States
Plukkie/1.5	Netherlands
SeznamBot/3.1-test1	Czech Republic
Baiduspider/2.0	China
proximic	United States
SiteExplorer/1.0b	United States
meanpathbot/1.0	Canada
PHPDevelBot/1.0)	Canada
oBot/2.3.1	Germany
aiHitBot/2.8	United Kingdom
YandexBot/3.0	Russian Federation

Table A.2: Examples for identified “known“ crawlers.

User-agent string	Crawler
msnbot-media/1.1 (+http://search.msn.com/msnbot.htm) Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)	msnbot-media/1.1 bingbot/2.0
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	Googlebot/2.1
msnbot/2.0b (+http://search.msn.com/msnbot.htm)	msnbot/2.0b
Mozilla/5.0 (compatible; YandexImages/3.0; +http://yandex.com/bots)	YandexImages/3.0
Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)	YandexBot/3.0
Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)	Baiduspider/2.0

Table A.3: Examples for identified “suspicious“ crawlers.

User-agent string	Crawler
Mozilla/5.0 (compatible; AhrefsBot/4.0; +http://ahrefs.com/robot/)	AhrefsBot/4.0
Mozilla/5.0 (compatible; Ezooms/1.0; ezooms.bot@gmail.com)	Ezooms/1.0
Mozilla/5.0 (compatible; Exabot/3.0; +http://www.exabot.com/go/robot)	Exabot/3.0
Opera/9.80 (Windows NT 6.1; U; en) Presto/2.10.289 Version/12.02	-
Mozilla/5.0 (Windows NT 6.1; rv:21.0) Gecko/20100101 Firefox/21.0	-
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:21.0) Gecko/20100101 Firefox/21.0	-

Table A.4: Examples for identified “other“ crawlers.

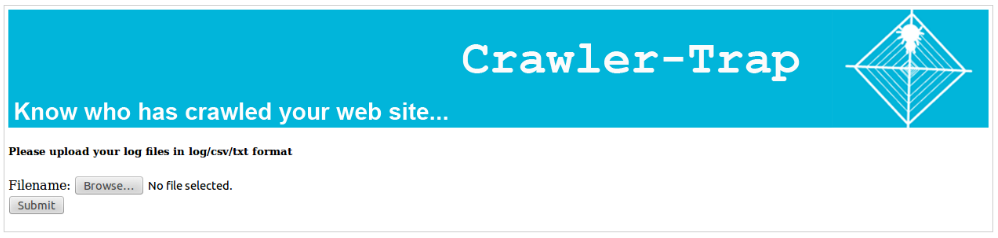
User-agent string	Crawler
Mozilla/5.0+(compatible; UptimeRobot/2.0; http://www.uptimerobot.com/)	UptimeRobot/2.0%
SeznamBot/3.0 (+http://fulltext.sblog.cz/)	SeznamBot/3.0
ShowyouBot (http://showyou.com/crawler)	ShowyouBot
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.110 Safari/537.36	-
Mozilla/5.0 (compatible; coccoc/1.0; +http://help.coccoc.com/)	coccoc/1.0
Mozilla/5.0 (compatible; SISTRIX Crawler; http://crawler.sistrix.net/)	SISTRIX Crawler

B Crawler-Trap Tool

B.1 Introduction

Figure B.1 and Figure B.2 show the upload screen and processing screen. In Figures B.3 and B.4 show the home page. It has the summary of the log file, crawler categorization with percentages, originating countries of web crawlers and amount of crawler visits per day. The information of the crawlers is listed in Figures B.5 and B.6. Basic crawler profile screen and detailed profile screens are shown in Figure B.7, B.8,B.9 and B.10. IP lookup screens are shown in Figures B.11 and B.12. Finally crawler database screens are shown in Figure B.13and Figure B.14.

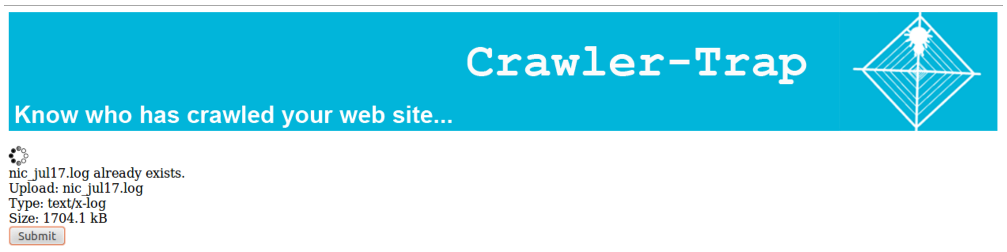
B.2 Screens



Please upload your log files in log/csv/txt format

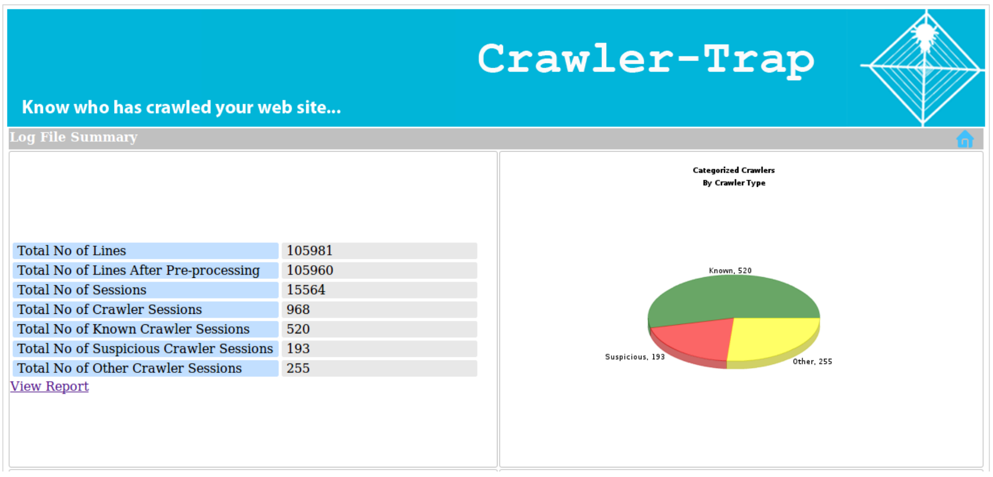
Filename: No file selected.

Figure B.1: Upload log file.



nic_jul17.log already exists.
Upload: nic_jul17.log
Type: text/x-log
Size: 1704.1 kB

Figure B.2: Process log file.



Total No of Lines	105981
Total No of Lines After Pre-processing	105960
Total No of Sessions	15564
Total No of Crawler Sessions	968
Total No of Known Crawler Sessions	520
Total No of Suspicious Crawler Sessions	193
Total No of Other Crawler Sessions	255

View Report

Figure B.3: Home page view I.

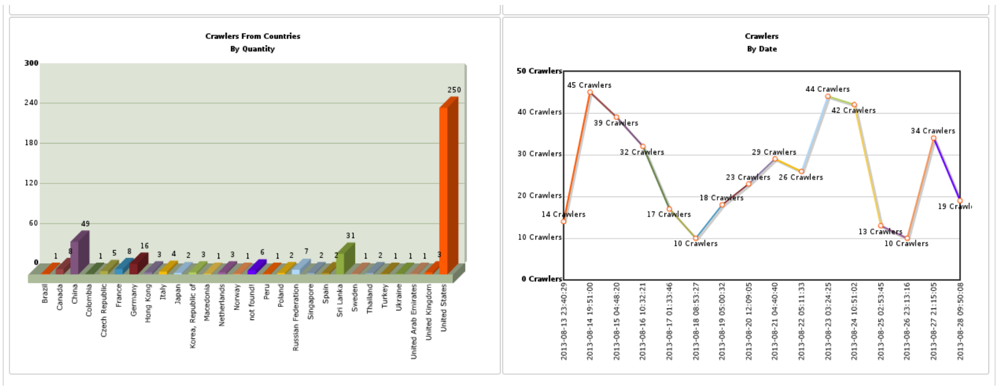
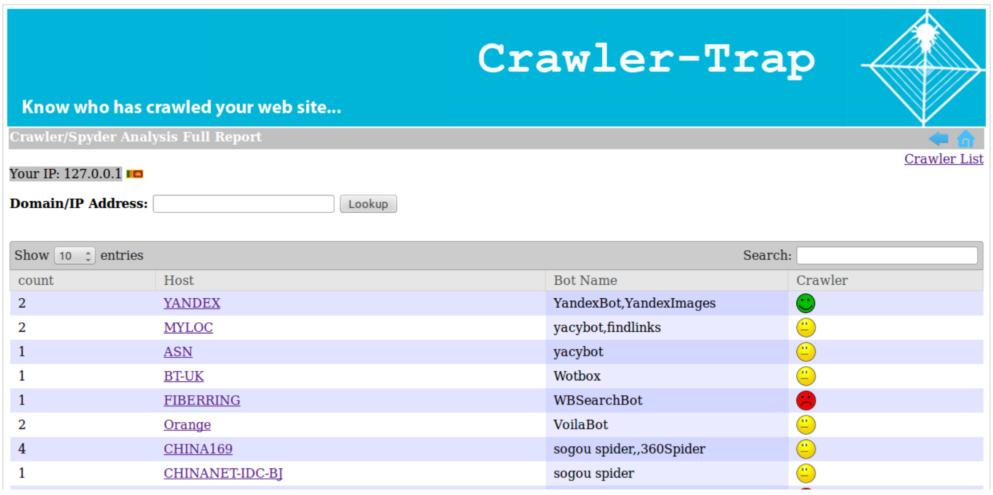


Figure B.4: Home page view II.



Crawler-Trap
Know who has crawled your web site...
Crawler/Spyder Analysis Full Report
Crawler List
Lookup

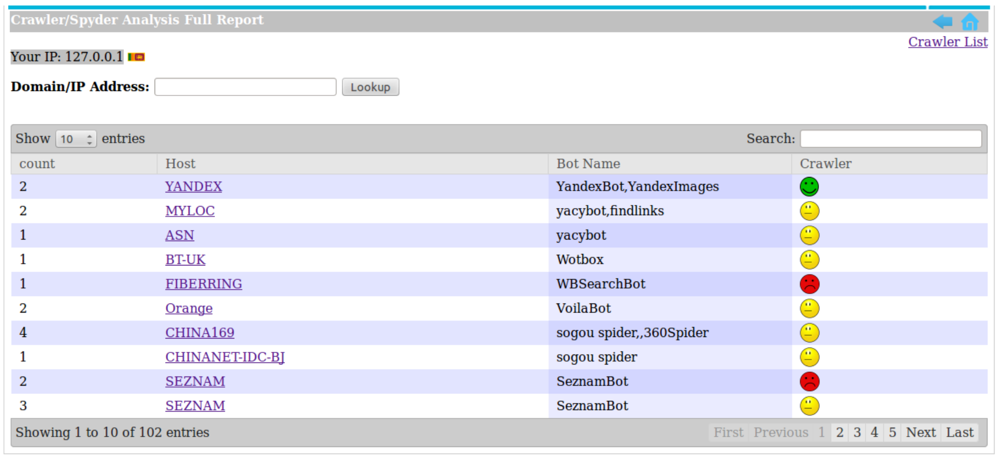
Your IP: 127.0.0.1

Domain/IP Address:

Show 10 entries Search:

count	Host	Bot Name	Crawler
2	YANDEX	YandexBot,YandexImages	🙂
2	MYLOC	yacybot,findlinks	😐
1	ASN	yacybot	😐
1	BT-UK	Wotbox	😐
1	FIBERRING	WBSearchBot	☹
2	Orange	VoilaBot	😐
4	CHINA169	sogou spider,,360Spider	😐
1	CHINANET-IDC-BJ	sogou spider	😐

Figure B.5: Crawler analysis report view I.


Crawler/Spyder Analysis Full Report

Crawler List

Your IP: 127.0.0.1

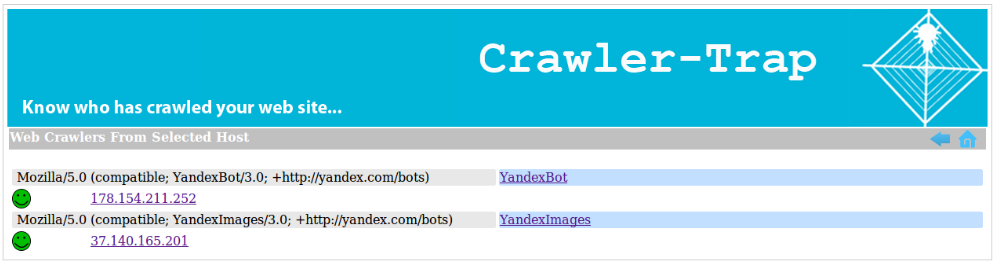
Domain/IP Address: Lookup

Show 10 entries Search:

count	Host	Bot Name	Crawler
2	YANDEX	YandexBot,YandexImages	🟢
2	MYLOC	yacybot,findlinks	🟡
1	ASN	yacybot	🟡
1	BT-UK	Wotbox	🟡
1	FIBERRING	WBSearchBot	🔴
2	Orange	VoilaBot	🟡
4	CHINA169	sogou spider,,360Spider	🟡
1	CHINANET-IDC-BJ	sogou spider	🟡
2	SEZNAM	SeznamBot	🔴
3	SEZNAM	SeznamBot	🟡

Showing 1 to 10 of 102 entries First Previous 1 2 3 4 5 Next Last

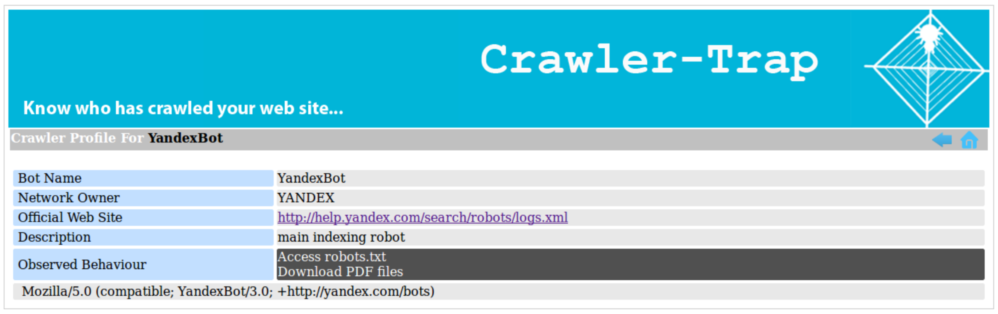
Figure B.6: Crawler analysis report view II.



Web Crawlers From Selected Host

Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)	YandexBot
🟢 178.154.211.252	
Mozilla/5.0 (compatible; YandexImages/3.0; +http://yandex.com/bots)	YandexImages
🟢 37.140.165.201	

Figure B.7: Crawler profile view I.

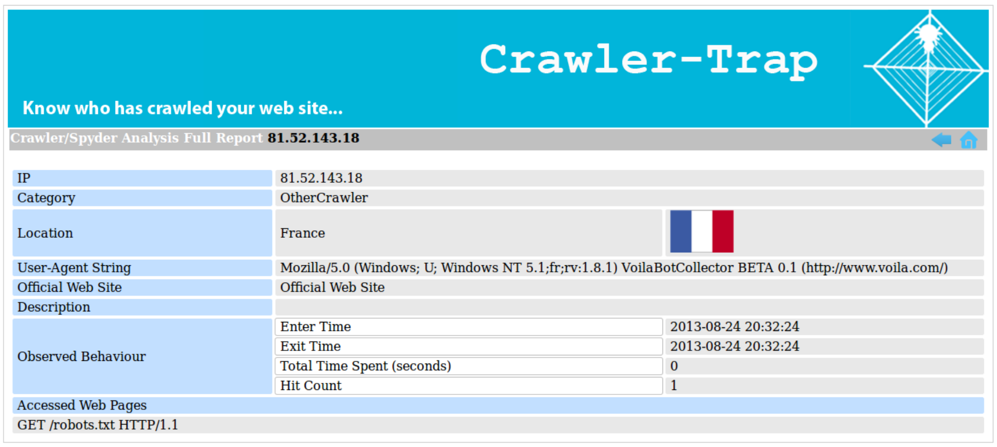


Crawler Profile For **YandexBot**

Bot Name	YandexBot
Network Owner	YANDEX
Official Web Site	http://help.yandex.com/search/robots/logs.xml
Description	main indexing robot
Observed Behaviour	Access robots.txt Download PDF files

Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)

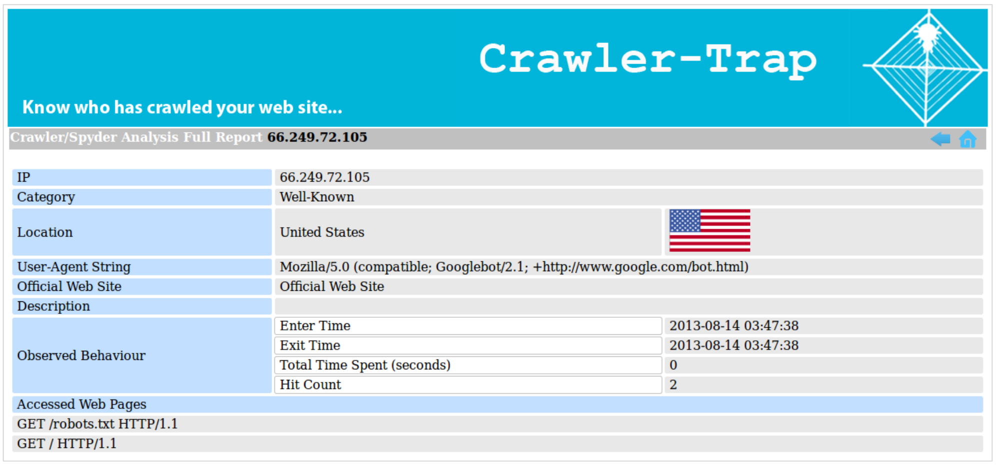
Figure B.8: Crawler profile view II.



Crawler/Spyder Analysis Full Report **81.52.143.18**

IP	81.52.143.18	
Category	OtherCrawler	
Location	France	
User-Agent String	Mozilla/5.0 (Windows; U; Windows NT 5.1;fr;rv:1.8.1) VoilaBotCollector BETA 0.1 (http://www.voila.com/)	
Official Web Site	Official Web Site	
Description		
Observed Behaviour	Enter Time	2013-08-24 20:32:24
	Exit Time	2013-08-24 20:32:24
	Total Time Spent (seconds)	0
	Hit Count	1
Accessed Web Pages		
GET /robots.txt HTTP/1.1		

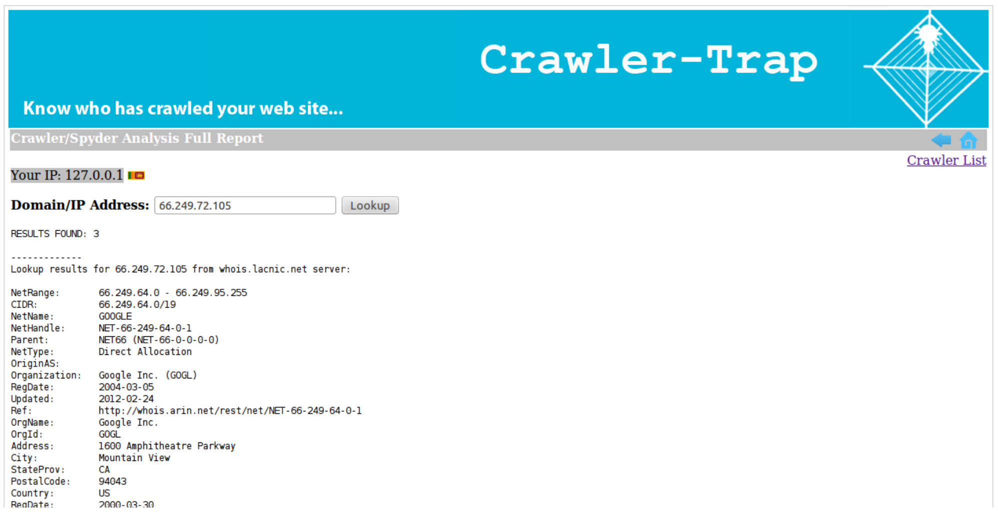
Figure B.9: Crawler profile view III.



Crawler/Spyder Analysis Full Report **66.249.72.105**

IP	66.249.72.105	
Category	Well-Known	
Location	United States	
User-Agent String	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	
Official Web Site	Official Web Site	
Description		
Observed Behaviour	Enter Time	2013-08-14 03:47:38
	Exit Time	2013-08-14 03:47:38
	Total Time Spent (seconds)	0
	Hit Count	2
Accessed Web Pages		
GET /robots.txt HTTP/1.1		
GET / HTTP/1.1		

Figure B.10: Crawler profile view III.

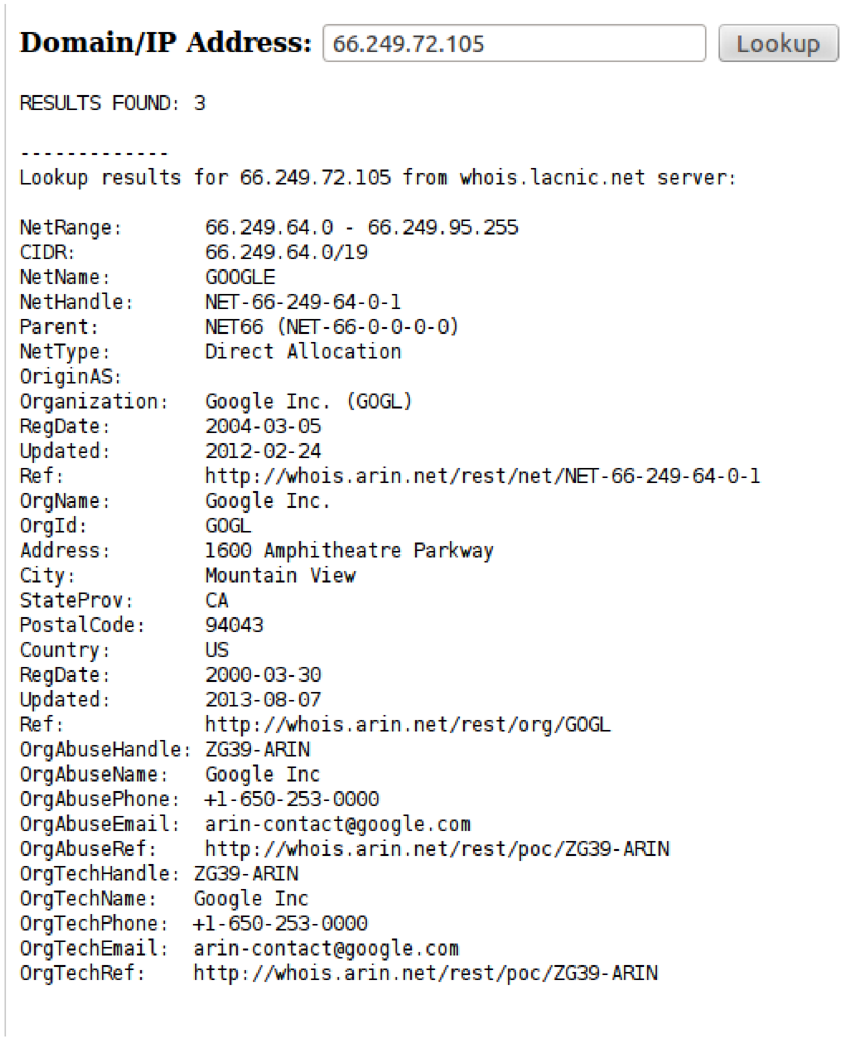


```
RESULTS FOUND: 3

-------------
Lookup results for 66.249.72.105 from whois.lacnic.net server:

NetRange:       66.249.64.0 - 66.249.95.255
CIDR:           66.249.64.0/19
NetName:        GOOGLE
NetHandle:      NET-66-249-64-0-1
Parent:         NET66 (NET-66-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Google Inc. (GOGL)
RegDate:        2004-03-05
Updated:        2012-02-24
Ref:            http://whois.arin.net/rest/net/NET-66-249-64-0-1
OrgName:        Google Inc.
OrgId:          GOGL
Address:        1600 Amphitheatre Parkway
City:           Mountain View
StateProv:      CA
PostalCode:     94043
Country:        US
RegDate:        2000-03-30
```

Figure B.11: IP lookup view I.

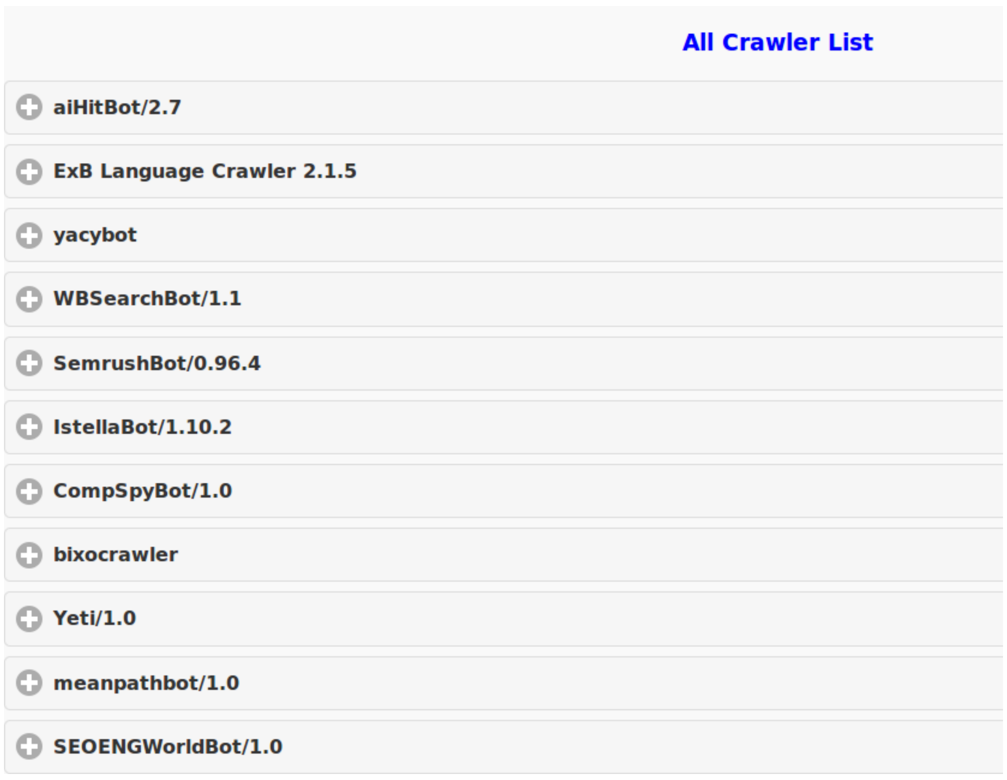


```
RESULTS FOUND: 3

-------------
Lookup results for 66.249.72.105 from whois.lacnic.net server:

NetRange:       66.249.64.0 - 66.249.95.255
CIDR:           66.249.64.0/19
NetName:        GOOGLE
NetHandle:      NET-66-249-64-0-1
Parent:         NET66 (NET-66-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Google Inc. (GOGL)
RegDate:        2004-03-05
Updated:        2012-02-24
Ref:            http://whois.arin.net/rest/net/NET-66-249-64-0-1
OrgName:        Google Inc.
OrgId:          GOGL
Address:        1600 Amphitheatre Parkway
City:           Mountain View
StateProv:      CA
PostalCode:     94043
Country:        US
RegDate:        2000-03-30
Updated:        2013-08-07
Ref:            http://whois.arin.net/rest/org/GOGL
OrgAbuseHandle: ZG39-ARIN
OrgAbuseName:   Google Inc
OrgAbusePhone:  +1-650-253-0000
OrgAbuseEmail:  arin-contact@google.com
OrgAbuseRef:    http://whois.arin.net/rest/poc/ZG39-ARIN
OrgTechHandle: ZG39-ARIN
OrgTechName:   Google Inc
OrgTechPhone:  +1-650-253-0000
OrgTechEmail:  arin-contact@google.com
OrgTechRef:    http://whois.arin.net/rest/poc/ZG39-ARIN
```

Figure B.12: IP lookup view II.


All Crawler List

aiHitBot/2.7

ExB Language Crawler 2.1.5

yacybot

WBSearchBot/1.1

SemrushBot/0.96.4

IstellaBot/1.10.2

CompSpyBot/1.0

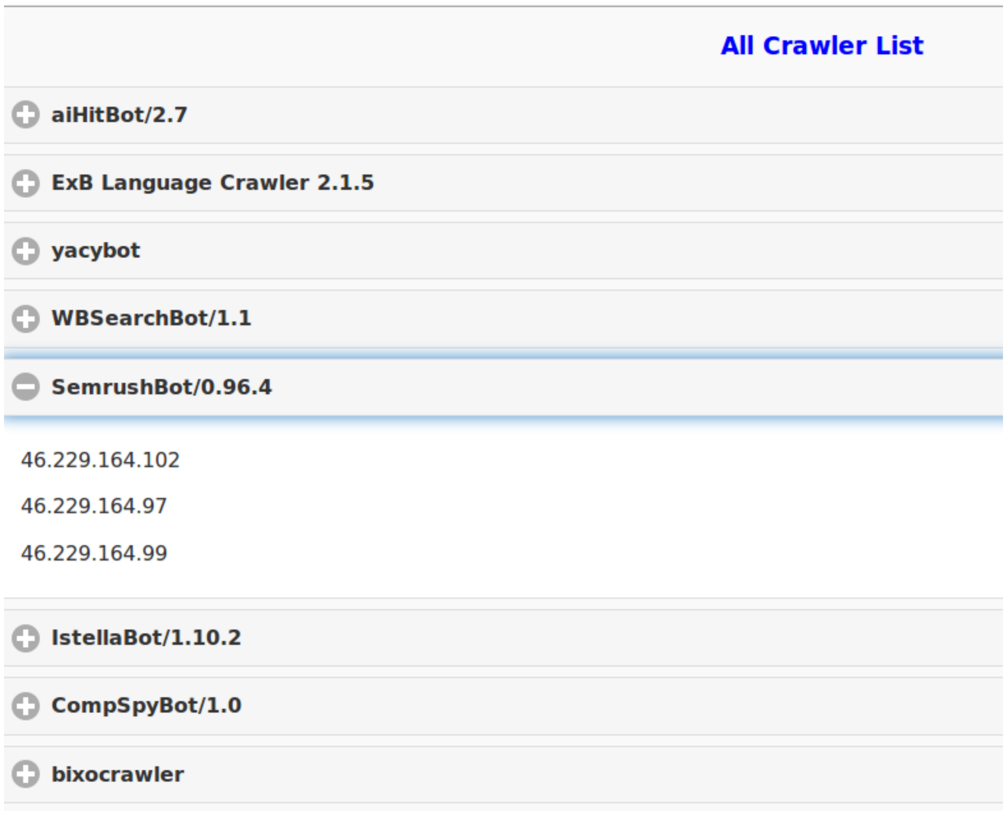
bixocrawler

Yeti/1.0

meanpathbot/1.0

SEOENGWorldBot/1.0

Figure B.13: Crawler list view I.


All Crawler List

⊕ aiHitBot/2.7

⊕ ExB Language Crawler 2.1.5

⊕ yacybot

⊕ WBSearchBot/1.1

⊖ SemrushBot/0.96.4

46.229.164.102

46.229.164.97

46.229.164.99

⊕ IstellaBot/1.10.2

⊕ CompSpyBot/1.0

⊕ bixocrawler

Figure B.14: Crawler list view II.