Cracking the Smart ClickBot

Conference Paper · September 2011

DOI: 10.1109/WSE.2011.6081830 · Source: DBLP

2 authors:

Chamila Walgampaya
University of Peradeniya
19 PUBLICATIONS **46** CITATIONS
SEE PROFILE

Mehmed Kantardzic
University of Louisville
120 PUBLICATIONS **581** CITATIONS
SEE PROFILE

Cracking The Smart ClickBot

Chamila Walgampaya
Computer Engineering and Computer Science Department
University of Louisville
Louisville, KY 40292, USA
Email: ckwalg01@louisville.edu

Mehmed Kantardzic
Computer Engineering and Computer Science Department
University of Louisville
Louisville, KY 40292, USA
Email: mmkant01@louisville.edu

***Abstract*—Nowadays, almost every task involving Web traversing and information retrieval recurs to Web robots. Web robots are software programs that automatically traverse the Web's hypertext structure. They proliferate rapidly aside with the growth of the Web and are extremely valuable and important means not only for the large search engines, but also for many specialized services such as investment portals, competitive intelligence tools, etc. While many web robots serve useful purposes, recently, there have been cases linked to fraudulent activities committed by these Web robots. *Click fraud*, which is the act of generating illegitimate clicks, is one of them. This paper details the architecture and functionality of the *Smart ClickBot*, a sophisticated software bot that is designed to commit click fraud. It was first detected and reported by NetMosaics Inc. in March, 2010, a real time click fraud detection and prevention solution provider. We discuss the machine learning algorithms used, to identify all clicks exhibiting Smart ClickBot like patterns. We constructed a Bayesian classifier that automatically classifies server log data as being Smart ClickBot or not. We also introduce a Benchmark data set for Smart ClickBot. We disclose the results of our investigation of this bot to educate the security research community and provide information regarding the novelties of the attack.**

***Keywords*-Click Fraud; Smart ClickBot; Bayesian Classifier;**

I. Introduction

A *web robot* is a software program that traverses the hypertext structure of the Web, starting from a 'seed' list of documents and recursively retrieving links accessible from that list [2], [3]. Web robots are also referred to as crawlers, wanders, or spiders. All major search engines including Google, Yahoo!, and Bing employ powerful crawlers that traverse the Internet continuously, trying to discover and retrieve as many Web pages as possible[6]. More recently, web robots have been used also as tools for focused crawling, *shopbot* implementation, and for supporting added-value services on the Web(portals, personalized and mobile services, etc.) [4]. As a consequence, the number and the variety of active robots operating on the Internet increases continuously, resulting to a noticeable impact on WWW traffic and Web-server activity. New robots are being launched all the time and it is not inconceivable that in the near future a large number of users will have their own personal spiders, raising even big questions as regarding how robot use is identified and treated [8].

There are a number of situations in which it is desirable to identify Web robots and distinguish them from other accesses. In such cases, it will be desirable to disable accesses by non-authorized robots [18]. First of all, e-commerce Web sites may be concerned about unauthorized deployment of shopbots for gathering business intelligence at their Web sites. Secondly, visits by Web robots can distort the access patterns that are mined by Web Usage Mining systems. Thirdly, the deployment of Web robots usually comes at the expense of other users,as they may consume considerable network bandwidth and server resources. Fourthly, Web robot accesses could be indicative of fraudulent behavior. For example, there are many click-through payment programs, such as Pay-Per-Click [10], established on the Web, in which an advertiser (i.e. the target site) would reward the referring Web site for every visitor who reach the target site by clicking on the referrer's advertisement or banner. Unscrupulous referrer site owners can easily abuse such a payment scheme by inflating the click-through rate using Web robots. hence, detection of Web robot sessions is absolutely necessary to protect the target site owner from such malpractices [19].

Despite a few early efforts towards identifying special classes of click-bot attacks such as clickbot.A and Bahama bot[5], [14], [13], there has been a little work on studying bot-generated click traffic. A number of challenges make this task difficult. First, the amount of data to process is often huge, on the order of terabytes per day. Thus any method that mines the data for identifying bot traffic has to be both efficient and scalable. Secondly, most of these data are not disclosed due to privacy, security and business policy issues. Furthermore, with many bot-net hosts available, attacks are getting increasingly stealthy with each host submitting only a few clicks to evade detection. Therefore, click bot detection methods cannot just focus on aggressive patterns, such as in Bahama bot, but also need to examine the low rate patterns that are mixed with normal traffic. Third, attackers can constantly craft new attacks to make them appear different and legitimate; thus we cannot use the training-based approaches that derive patterns from historical attacks [21]. Finally, with the lack of ground truth, evaluating detection results is non

trivial and requires different methodology and metrics than the detection methods.

The main contributions of this paper are as follows:

1) Specifying characteristics of SmartClick Bot, an advanced and intelligent click bot.
2) Introducing the methodologies that we have developed to detect click bots not based on characteristic of individual clicks but based on statistics of the context of the click.
3) Introducing a Benchmark data set generated by smart click bot. Various click fraud solution providers and practitioners can use this data set for their own experimentations.

This paper is organized as follows. In Section II an overview of the Smart ClickBot is given. Section III introduces the proposed methodologies for Smart ClickBot detection. Experimental results and discussion are provided in Section IV. Conclusions are given in Section V.

II. An Overview of the Smart ClickBot

The Smart ClickBot is a software Web robot that clicks on ads (by issuing HTTP requests for advertiser web pages) to help an attacker conduct click fraud[1]. It was first detected and reported by the NetMosaics click fraud detection system in 2010 [10], [9], [11]. Smart ClickBot is a for-sale click bot and it can be purchased[20]. Once installed and configured the Smart ClickBot is able to act by itself. It uses anonymous proxies to generate traffic with IP diversity. It also has a random user-agent generator that generates user-agents registered to famous HTTP browsers. By doing so, it can mimic a request originated from a valid browser because click fraud detection solutions usually suspect clicks without a valid user-agent field [19]. To make it look more realistic it can even attach a referrer field. Referrer in pay-per-click system is a website that helps a web user to reach another website. Therefore if the referrer field carries values correspond to famous search engines or other popular websites it will be least suspicious to anybody observing the server logs.

Operator of the Smart ClickBot can set the time interval between successive clicks, known as the *Click-Through-rate*(CTR)[2] and configure to run multiple click campaigns simultaneously. The Smart ClickBot has 3 distinct campaign modes. They are: single hit, list-like, and banner-like, which are especially designed to suit different webpage structures. Once the bot loads the webpage that has the advertisements the user can specify where to click.

[1]the act of generating illegitimate clicks to make profit or deplete competitor advertisement budget.

[2]CTR is a way of measuring the success of an online advertising campaign. A CTR is obtained by dividing the "number of users who clicked on an ad" on a web page by the "number of times the ad was delivered" (impressions). For example, if a banner ad was delivered 100 times (impressions delivered) and 1 person clicked on it (clicks recorded), then the resulting CTR would be 1 percent.

In the next section, we discuss in detail the systematic approach taken by the NetMosaics system to detect click patterns generated by the Smart ClickBot.

III. Methodology for Smart Clickbot detection

A. Data Collection, Pre-processing and Session identification

Interactions with a Web server, either by humans or software, are recorded in the server access logs. To characterize the behavior of bots statistically, we need to be able to isolate the behavior of robots from that of the general population of (human) Web users.

Most of the existing bot detection systems uses only the server side data in their analysis and therefore entirely depends on these server log data to identify robot *sessions*. A session is the duration that a user (either human or software bot) maintains an active HTTP connection with the server. But, because of the stateless nature of HTTP traffic, incoming requests are considered and logged as independent events. Therefore, access logs do not contain any information that could relate together requests issued during a single "visit" of one user to the Web-pages of a Web server [17].

Furthermore, in [19] Tan and Kumar stated that "Without client-side tracking, cookies or embedded session identifiers, it is extremely difficult to identify the individual sessions in the Web server logs reliably". Even though there were some alternative attempts[16] to group server logs into sessions, none of them showed promising results.

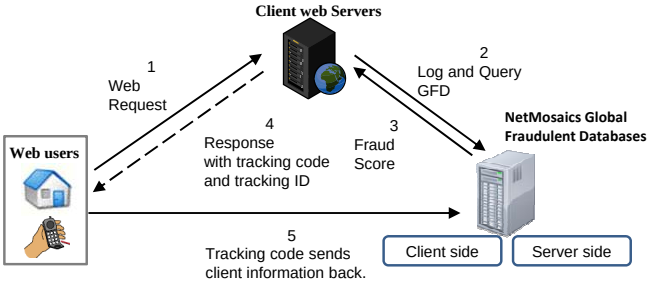

Figure 1. The NetMosaics data collection process.

Therefore, we developed a click fraud detection system, NetMosaics, that uses both server side data and client side data to better understand the context of the click, while providing an easy platform to generate user sessions. Figure 1 shows the high-level processing flow of the NetMosaics system. In this system user sessions are easily matched with a unique tracking number that is shared by both server side and client side data. Robot generated traffic usually do not have client side entries. Therefore, none of the bot traffic will be merged, and they will be left in the server side log. The matched traffic is further analyzed by the NetMosaics system for more suspicious activities to improve the quality of the incoming traffic. Only the improved matched traffic

will be delivered to the NetMosaics clients. What is left in the server side is separately analyzed, which is the scope of this paper, for potential bot networks. We have discussed the functionality of our system in detail in [10], [9], and [11].

Our system has been collecting and analyzing click data continuously since it was launched in 2004. While delivering higher quality traffic to our clients we periodically analyze what is left in server side, which are mostly bot data, collectively to identify unknown bots and their behavioral patterns. Discovery of Smart ClickBot is a result of such an attempt. One of our honeypot[3] servers have infected with the Smart ClickBot. After the detection of the bot we were able to reverse engineer it to obtain a copy.

Multiple copies of the isolated click bot is then installed and used to carry out attacks in a controlled environment. Bot clicks are collected for a period of 7 days from 3/3/2011 to 3/9/2011. 1000s of different IP addresses are generated through proxy servers. Also, another 100s of referrer sites are used. Time between clicks is varied randomly between 0 - 1000 seconds. They are configured to issue HTTP clicks at www.thebestmusicsites.org, a Web site that we have designed, that has both text and banner advertisement links. A bot is set to issues clicks between 0-5 in a given session. During the experiment, bot configurations are randomly changed to maximize the diversity. This data set will be available publicly to researchers who wants to test their click fraud detection systems against this new type of click bot. Since Smart ClickBot has been developed with utmost care to not to be detected, the techniques that we have developed may help to detect even other types of click bots.

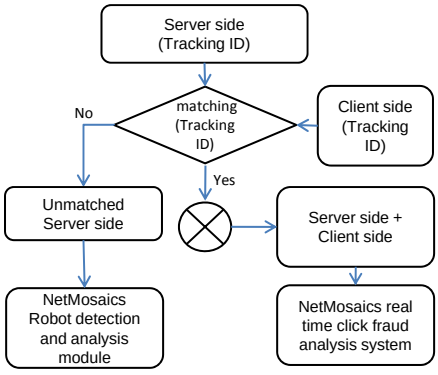

Figure 2. Robot data collection process.

Figure 2 shows the high level view of the flow diagram for bot traffic isolation. These isolated potential bot traffic is pre-processed to remove known bots. These include search engine crawlers such as Google's googlebot, Yahoo!'s Yahoo slurp, and Bing's bingbot, known click bots such as Clickbot.A and Bahamabot, link crawlers and news bots etc. For this purpose, we have used the data available at [1]. Top 10 of those filtered bots are shown in Table I with their user agent and frequencies. These filtered bot data is then divided into sessions based on the techniques explained in [19].

Once the bot data is grouped into sessions, our main idea in the experiment is to classify server side data into two classes: Class 1, and Class 2, where Class 1 will have "clicks" originated from Smart ClickBot and everything else will belong to the Class 2. For example, Class 2 may contain human clicks that do not accept cookies or javascript in their devices or they may be clicks that has only server side information due to an error in HTTP communication between server and client or it may be even new software bots that are not discovered yet. Therefore the next step is to experimentally derive the properties of each session that will distinguish clicks in Class 1 from that of Class 2. Table II presents a summary of attributes that can be derived from the sever sessions. Some of these features are temporal, while some are binary. Extraction process of these features is discussed in the following section.

B. Context Feature Extraction

NetMosaics explores the distributed nature of stealthy attacks. Since click bots are pre-configured, the generated traffic by them is usually similar in nature. NetMosaics leverages this property and aims to identify groups with similar activities.

1) Periodicity of Smart ClickBot: Web robots, especially click bots, usually exhibit periodic behavior because they are preset to activate after a certain time interval. At this point they randomly select a (IP, referrer, user agent) combination from the predefined lists and issue clicks in the form of HTTP requests. By observing the collected data we have seen these lists are updated daily. Therefore we can assume the values in the lists to be the same for at least 24 hour period. Even previous studies of Web robots support the 24 hours threshold time[19]. For each IP we extracted all requests originated within the past 24 hours. By plotting the time activity (i.e. the active and inactive periods of time) of bot processes issuing requests, we observed that several of them seem to exhibit, at least partially, a periodic pattern. We investigated further this observation and verified the periodicity for several IP addresses used by the Smart ClickBot and estimated their time cycles.

For this task, we used the Fast Fourier Transform (FFT). The FFT maps a function in the time field to a, complex in general, function in the frequency field [6]. The idea is that by observing peaks of magnitude in the frequency field we can easily conclude that time activity has periodicity. The frequency coordinate of each possible peak is inversely proportional to the time cycle of the periodicity. Since we are not interested in the phase of the frequency plot, we

[3]honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems. Generally it consists of a computer, data, or a network site that appears to be part of a network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers.

Table I
FILTERED USERAGENTS

User Agent	Requests
google.com	244
search.msn.com	149
Mozilla/5.0 (compatible; YandexBot/3.0; http://yandex.com/bots)	120
yahoo.com	105
Sogou web spider/4.0(http://www.sogou.com/docs/help/webmasters.htm#07)	79
Mozilla/5.0 (Windows; U; Windows NT 5.1; en; rv:1.9.0.13) Gecko/2009073022 Firefox/3.5.2 (.NET CLR 3.5.30729) SurveyBot/2.3 (DomainTools)	55
Mozilla/5.0 (compatible; bingbot/2.0; http://www.bing.com/bingbot.htm)	42
Netcraft	32
whois.sc	21

Table II
SUMMARY OF ATTRIBUTES DERIVED FROM THE SERVER SESSIONS.

Id	Attribute Name	Remark	Purpose
1	Periodicity	Periodicity of the attack	Feature
2	trackingIDs	Tracking IDs per IP	Feature
3	multiAgents	Unique user agents recorded per single IP	Feature
4	referrerPattern	Referrer and IP distribution	Feature
5	HEAD	Page requests made with HEAD method	Classify
6	GET	Page requests made with GET method	Feature
7	POST	Page requests made with POST method	Feature
8	clickRate	Maximum clicks per session	Feature
9	Duration	Duration of session	Feature
10	imageRequests	Percentage of image requests	Feature
11	pdf/ps	Percentage of pdf/ps requests	Feature
12	4xx	Percentage of 4xx error responses	Feature
13	Robot.txt	Whether Robot.txt file accessed during the session	Classify
14	% proxy	Percentage of proxy servers used	Feature

illustrate the spectral density function, which is the square of the magnitude of the FFT.

Before implementing the FFT, time is assumed to be sliced; we used a 30 second time interval (granularity). Ideally, the granularity should be as small as possible, but we tried to keep the number of resulting points relatively small for a faster FFT computation.

We count the requests issued from an IP address of interest in each time interval. Because our focus of interest at this stage is on the presence of some periodic action, we assign the value of one to the intervals that have at least one hit and the value of zero to the ones with zero hits. Consequently, we produce an ON-OFF signal that represents the Smart ClickBot's time activity for the selected granularity. This signal is passed as input to the FFT function. The resulting diagrams reveal a periodicity in the requests issued by IP addresses belonging to the click bot; in some cases this phenomenon is rather intense.

Figure 3 presents the ON-OFF signal of an IP address. In Figure 4, we plot the power spectral density function. FFT specifies the main periods observed on that signal. For example, in Figure 3, we observe a periodic behavior between 05:15 - 08:15, which corresponds to the peak of around 0.15 in the Figure 4.

Similar results are observed for couple of other IP addresses but due to space limitation we will not produce the results here. We have not seen similar patterns from IP addresses that do not belong to Smart ClickBot, which were left in the server logs. We can therefore conclude that periodic activity can be expected from the Smart ClickBot. Hence this feature will be binary and for IPs that shows periodicity we assign a value 1, while the rest, including some of Smart ClickBot IP addresses that does not show the same behavior, are assigned 0.

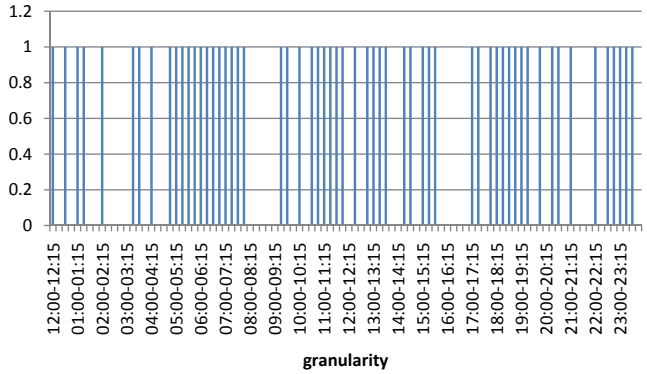

Figure 3. ON-OFF Signal for an IP used by Smart ClickBot.

2) Tracking ID per IP: The NetMosaics system generates a 128bit unique tracking id (in the form of a cookie) for every server side request. This tracking id is installed in

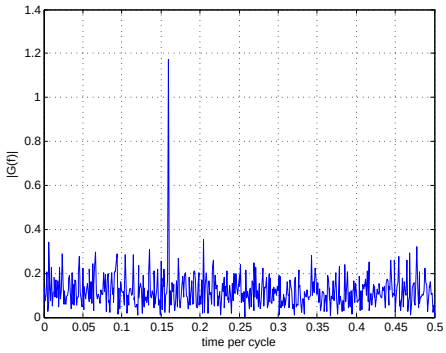

Figure 4. Power spectral density for an IP used by Smart ClickBot.

the client's computer and will be incorporated with every revisit request to the same website by the client in the next 24 hours. Usually software bots do not accept these cookies. Therefore, they generate new tracking id for every visit to the same website. Smart ClickBot is no different. It generated multiple tracking IDs for the same IP and almost all of these IP addresses belong to free proxy servers. Table III shows the summary of average number of tracking IDs generated for a 24 hour period by the top 10 IP addresses used by the Smart ClickBot. If the IP address does not belong to a proxy server we usually treat it as a non software bot. For example it may be a human clicker, which does not allow cookies to be installed or who deletes cookies every time the session is over.

Table III
TRACKING ID PER IP

IP Address	day 1	IP Address	day 2
200.29.216.146	88	115.248.202.21	49
187.4.128.12	85	208.253.158.6	40
190.203.69.69	81	187.11.201.164	39
221.7.145.42	72	200.29.216.146	36
203.153.25.218	59	222.255.28.33	34
115.78.227.155	46	190.203.69.69	33
115.78.224.215	45	203.153.25.218	31
222.255.28.33	43	187.4.128.12	28
125.162.92.233	42	221.7.145.42	26
190.128.218.90	40	89.187.142.113	21

We also discovered a secondary property of these IP addresses, which has large number of tracking IDs. We have seen that almost 90% of the time the same set of IPs contain in all the lists of 24 hours. Even though we assumed they are randomly picked, it seems that the bot has some preference to certain proxy servers may be based on the level of animosity of the proxy.

We have also observed a unique pattern in the wagon wheel for visitor distribution. For example, in Table III, day 1 corresponds to 24 hours while day 2 corresponds to visitors for 12 hour period. But the contribution from each IP looks alike, immaterial of the day or the number of hours observed. Figure 5 shows the wagon wheel distribution of visitors for those two days (day 1(left), day 2(right)).

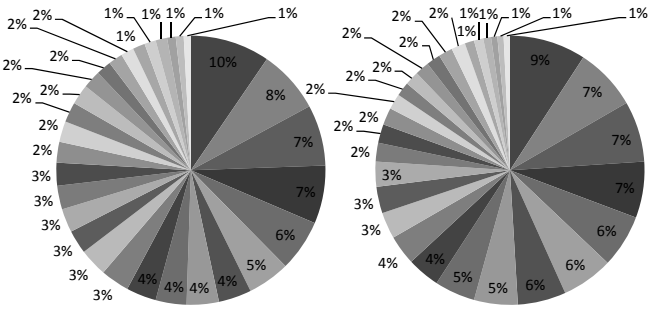

Figure 5. Frequency of TrackingID generation.

3) User Agent per IP: User Agent identifies the type of browser someone is using to surf the Internet. There is a favorite browser for all of us and at least we stick with it for a while. We mark our favorites, bookmark certain websites, maintain history etc. and it will be inconvenient for us to switch browsers. Therefore for a given non-shared IP address variation of the user agent is minimal. User Agent is not defined only for browsers. Even for all the legitimate software bots there are unique user agents. They identify themselves as bots whenever visiting websites if they are cooperative. Bots that are created to carryout fraudulent activities use these user agents to mimic themselves as valid user agents. Smart ClickBot seems to pick a user agent randomly from a pre defined list and does not care about the history of user agents assigned to a particular IP. Therefore, we have seen a large number of user agent values assigned for any given IP. Sometimes the variation exceeds 50, and it is highly unlikely that somebody who is legitimately browsing Internet has over 50 user agents.

Among this list, we have also found some outdated user agents. With normal traffic these type of user agents are rarely reported but a higher percentage can be seen with the traffic generated from the Smart ClickBot. Table IV lists few of these outdated user agents with its average request per 24 hours.

Table IV
FREQUENCY OF OUTDATED USERAGENTS

User Agent	Frequency
Mozilla/4.0 (compatible; MSIE 4.01; Windows 95)	39
Mozilla/4.0 (compatible; MSIE 5.01; Windows 95)	36
Mozilla/4.0 (compatible; MSIE 5.5; Windows 95)	33
Mozilla/4.0 (compatible; MSIE 5.0; AOL 5.0; Windows 95; DigExt)	30
Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)	27

4) Referrer and IP distribution: The referrer field is provided by HTTP protocol to allow a Web client (particularly,

a Web browser) to specify the address of the Web page that contains the link the client followed in order to reach the current requested page. Unassigned referrer field is often considered one of the most apparent characteristics of Web robots [12]. As typical Web robots parse a page and build up the list of pages to be visited, referrer field is frequently left blank. In interactive Web surfing environment, the field would contain the URL that led to the current request. Our analysis reveals that such observation is generally, but not always, true. Especially with the Smart ClickBot to avert the detection they have randomly assign a referrer value. But since this list is concise we have seen a pattern that pretty much every referring site is recorded the same amount of referrals. This does not happen with human clicks and the variation of referrals is usually high. Table V shows an example for a referral traffic for the www.thebestmusicsites.org website within a 24 hour period.

Table V
FREQUENCY OF TOP REFERRER SITES

Referrer	IP frequency
http://www.referrer1.edu/	30
http://www.referrer2.com/	29
http://www.referrer3.com/	28
http://www.referrer4.com/	27
http://www.cecs.referrer5.edu/	27
http://www.referrer6.com/	26
http://www.referrer7.com/	26
http://www.referrer8.com/	26
http://www.referrer9.org/	23
http://www.referrer10.com/	23

5) Percentage of HEAD and GET Requests: Many suggest sessions containing a large number of HEAD requests as those generated by web robots [19], [6]. For example, if all the requests are made using the HEAD method, then the session is most likely created by a Web robot. Guidelines issued to web robot designers strongly recommend that only HEAD method be used to minimize performance impact on web servers. However, [6] reported that many Web bots used HEAD method in less than half (e.g., 10 to 50%) of their requests. In our data, almost all (e.g., over 99.9%) the requests made by Smart ClickBot used GET method. Therefore when the HEAD/GET request percentage is high, we use this feature as a strong feature to separate non Smart ClickBot requests from the server logs.

6) Percentage of POST Requests: A POST request is used to send data to the server to be processed in some way, like by a CGI script. It is highly unlikely that a Web bot sends POST request to a Web server. Therefore, we can use this feature to isolate human issued requests that are left in the server logs.

7) Maximum clicks in a session: A click is a request for an HTML file in a Web server. The "Maximum clicks in a session" is a feature corresponds to the maximum number of such HTML requests received within a certain time-window inside a session. The intuition behind this feature is two fold:

1) To isolate human clicks from bot clicks: there is an upper bound on the maximum number of clicks that a human can issue within some specific time-frame t, which is dictated by human factors. To capture this feature, we first set the time-frame value of t and then use a sliding window of time t over a given session in order to measure the maximum sustained click rate in that session. For example, if we set t to 10 seconds and find that the maximum number of clicks within some 10-second time-window inside that session is 40, we conclude that the maximum sustained click rate is 4 clicks per second. This indicates a robot-like rather than a human-like behavior. The sliding window approach starts from the first HTML request of a session and keeps a record of the maximum number of clicks within each window, sliding the window by one HTML request until we reach the last one of the given session. The maximum of all the clicks per window gives the value of this feature.
2) To isolate Smart ClickBots from other software bots: Smart ClickBot can issue $0-n$ number of clicks during a session. But we have seen that for every such click Smart ClickBot changes its IP address, tracking id, referrer, and the user agent. Therefore, there is no way to recognize all the clicks belong to a one session. In the server logs all the clicks generated in a session appears as several single click sessions. We can use this implementation drawback to recognize and isolate Smart ClickBot clicks from other type of software bots.

8) Duration of session: Duration of session is the number of seconds that have elapsed between the first and the last request. Crawler-induced sessions tend to have a much longer duration than human sessions. Smart ClickBot can issue any number of clicks during a session. But we have seen that for every such click it changes its IP address, tracking id, referrer, and the user agent values. Therefore, there is no way to recognize all the clicks belong to one session. In the server logs, all clicks belong to one session, appear as multiple single click sessions. Therefore for this feature, which is a binary, sessions having only a single click caries value 0, while the rest carries value 1.

9) Percentage of image requests: This feature denotes the percentage of requests to image files (e.g. jpg, gif). An earlier study showed that crawler requests for image resources are negligible[17]. In contrast, human generated traffic will have access records to images in a website, because all images are loaded within the user session. Therefore, we can use this feature to differentiate human generated traffic, that for some reason did not have client side activities, from bot generated traffic. The percentage of requests seeking postscript(ps) and pdf files is also a possible

feature to use. But in our experiment we did not consider this feature. Previous studies show that, in contrast to image requests, some crawlers, tend to have a higher percentage of pdf/ps requests than humans[17].

10) HTTP response codes: Web bots such as link validators and email harvestors may have a higher proportion of 4xx error codes in their requests, as they are blindly traversing the web infrastructure. But, clicks from the Smart ClickBot should be very precise because these links are previously checked by humans before launching the attacks, hence we expect fewer 4xx error codes. Human clickers may stand between these two extreme ends because human users are able to recognize, memorize and avoid erroneous links, unavailable resources and servers. Table VI shows the percentages of response codes received from bot and non-bot traffic for a 7 days period. We can clearly see that bot traffic has lesser 404 errors compared to non-bot traffic.

We can also expect lesser 304 response codes with Smart ClickBot traffic. HTTP 304 response code is for "not modified". With this message the web server is basically telling the browser "this file has not changed since the last time you requested it." If a client gets a 304 Not Modified message, then it is the client's responsibility to display the resource in question from its own cache. Since Smart ClickBot does not cache any information it usually gets only HTTP 200 response code. HTTP 200 is telling the browser '"here is a successful response" - which should be returned when it is either the first time your browser is accessing the file or the first time a modified copy is being accessed. Table VI shows the differences in HTTP 200 and HTTP 304 response percentages.

Table VI
PERCENTAGE OF BOT VS. NON-BOT HTTP RESPONSE CODES

Response code	Percentage in Bot traffic	Percentage in non-Bot traffic
200 - OK	97.0%	10.3%
304 - Not Modified	1.7%	80%
307 - Moved Temporarily	0%	2.9%
404 - Not Found	1.2%	6.7%
500 - Internal Server Error	0.1%	0.2%

11) Robots.txt file request: The Robot Exclusion Standard, also known as the Robots Exclusion Protocol or robots.txt protocol, is a convention to prevent cooperating web crawlers and other web robots from accessing all or part of a website which is otherwise publicly viewable. If a site owner wishes to give instructions to web robots they must place a text file called robots.txt in the root of the web site hierarchy such as www.thebestmusicsites.org/robots.txt. Robots that choose to follow the instructions try to fetch this file and read the instructions before fetching any other file from the web site. If this file does not exist web robots assume that the web owner wishes to provide no specific instructions.

We made available the robot.txt file for the experimented website. If a request to the robots.txt file was made during a session, we consider it as a strong evidence to believe that the session belongs to a bot. However because compliance to the Robot Exclusion standard is voluntary, and many robots simply do not follow the proposed standard, we can not totally rely on this criteria to detect Web robots.

12) Distribution of countries: Table VII lists the top 20 countries that Smart ClickBot uses to generate IP diversity. Even though highest number of proxy IPs are from the US, there is a large amount proxy IPs recorded from Indonesia, Brazil, China, and India. Collectively their traffic is larger than the US alone. Since we have used an experimental website to collect data, at this time we do not have enough traffic to find out its actual(non-Robot) country distribution. Therefore, we did not include this as a feature, even though we will use it as soon as the information is available.

C. Classification

After deriving the session features, classification models are built using the Bayesian Networks.

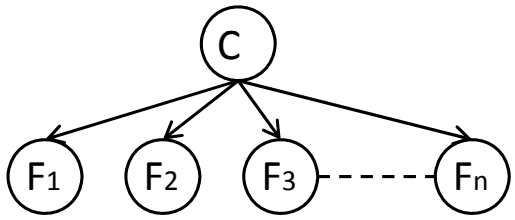

Figure 6. Bayesian Network as a classifier.

Bayesian Networks [7], [15] are directed acyclic graphs in which the nodes represent multi-valued variables, comprising a collection of mutually exclusive and exhaustive hypotheses. The arcs signify direct dependencies between the linked variables and the direction of the arcs is from causes to effects. The strengths of these dependencies are quantified by conditional probabilities. More specifically, each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node, where $Parents(X_i)$ denotes the parent variables of X_i. This conditional probability distribution, which defines the conditional probability table of the variable, describes the probability distribution of the variable for each configuration of its parents. The graph encodes that each node is conditionally independent of its non-descendants, given its parents [7].

Naive Bayes is a special case of a Bayesian network, where a single cause (the class) directly influences a number of effects (the features) and the cause variable has no parents. This network structure is shown in Figure 6. Again, the independence assumption encoded by this model is that each feature is conditionally independent given the class value.

Table VII
DISTRIBUTION OF COUNTRIES

Country	Requests	Visitors	Country	Requests	Visitors
United States	4249	824	Korea, Republic of.	423	81
Indonesia	3105	822	Spain	407	94
Brazil	2224	409	Austria	363	36
China	2148	367	South Africa	287	35
India	1121	85	Puerto Rico	271	35
Chile	912	63	Kenya	267	108
Vietnam	865	133	Croatia	262	36
Thailand	841	162	Singapore	256	48
Venezuela	618	106	Turkey	255	69
Russian Federation	429	216	Czech Republic	250	40

Considering Figure 6, assume that $F_1, F_2, .., F_n$ are n features and f_i represents the value of feature F_i. Assume also that C is the class variable and let c represent a possible value (label) of C. Using Bayes rule and the conditional independence assumption, we can derive the posterior probability of each class label $c \in C$, i.e. the probability of the class label given the features observed, to be given by the formula:

$$P(c|f_1, f_2, ..., f_n) = \frac{P(c)\prod_{n=1}^{n} P(f_i|c)}{P(f1, f2, ..., fn)} \quad (1)$$

The class variable C is assigned the label that gives the maximum posterior probability given the features observed. More specifically:

$$class = argmax_{c \in C} P(c)\prod_{i=1}^{n} P(f_i|c) \quad (2)$$

For Smart ClickBot detection we used one, similar to Bayesian Network structure shown in Figure 6. Each child node corresponds to one of the features we presented earlier in section IV.(B). The root node represents the *class* variable.

In the following section we present the experiments performed in order to apply our methodology and evaluate the performance of the Smart ClickBot detection system.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

There are two main objectives in this experiment:(a) to use features described in section IV.(B) to identify as many Smart ClickBot sessions as possible, (b) to find a good model for predicting Smart ClickBot sessions based upon their access features.

We have collected 22991 server side clicks during the period from 3/3/2011 to 3/9/2011. A human expert has labelled the entire data set so that it can be used for model evaluation. To build the classification model we have used the first 5000 samples. In this training data set there were 4501 Smart ClickBot (Class 1) clicks and 499 non-Smart ClickBot (Class 2) clicks. Since class representation is not balanced, training the model was a challenging task. If a model is built using an imbalanced dataset, its characteristics tend to be biased towards the majority class. Especially with the Naive Bayes classifier, the prior probability in the majority class overshadows the differences that exist in the conditional probability entries that quantify the relationship between feature and class variables [17].

There are a few ways to compensate the imbalanced class distribution. We can use techniques such as bagging and boosting or resampling. We used resampling as it was used successfully in a similar study discussed in [17]. Resampling modifies the prior probabilities of the majority and minority class by changing the records on each of the two classes. For this purpose we have used both random over sampling and random under sampling. Over sampling is used with the minority class (Class 2), while Under sampling is used with majority class (Class 1). Table VIII shows the resampled data that we have used to build the classifiers($C_1, C_2, C_3, C_4, and\ C_5$) along with the new prior probability distributions. C_1 is built with the original data set without resampling. Both C_2 and C_3 are built with oversampling Class 2, while C_4 and C_5 are built with under sampling Class 1.

Table VIII
TRAINING DATA SET CONFIGURATION

Data Set	Classifier	Class 1	Class 2	Prior Probabilities
1	C_1	4501	499	(0.90, 0.10)
2	C_2	4501	4501	(0.50, 0.50)
3	C_3	4501	899	(0.83, 0.17)
4	C_4	2436	499	(0.83, 0.17)
5	C_5	499	499	(0.50, 0.50)

We have tested the five Bayesian classifiers with the rest of the 17991 records. For the evaluation purposes "Accuracy" is a reasonable metric but it has the underline assumption that the data set remains evenly distributed i.e. between Class 1(Smart ClickBot), and Class 2(non-Smart ClickBot). When equal class distribution is not present we can use *Precision* and *Recall* to compare the models.

$Precision(p) =$

$$\frac{no.\ of\ SmartClickBot\ sessions\ found\ correctly}{total\ no.\ of\ predicted\ SmartClickBot\ sessions} \quad (3)$$

$$Recall(r) = \frac{no.\ of\ SmartClickBot\ sessions\ found\ correctly}{total\ no.\ of\ actual\ SmartClickBot\ sessions} \quad (4)$$

A classifier that assigns the value 1 to every session will have perfect recall but poor precision. In practice, the two metrics are often summarized into a single value, called the F_1-measure [19].

The F_1 score can be interpreted as a weighted average of the precision and recall. It summarizes the two metrics into a single value, in a way that both metrics are given equal importance. Recall and precision should therefore be close to each other, otherwise the F_1-measure yields a value closer to the smaller of the two. F_1 score reaches its best value at 1 and worst score at 0. Table IX and Figure 7 show the Precision, Recall, and F_1 measure obtained by the five classifiers.

Table IX
TRAINING DATA SET CONFIGURATION

Classifier	Precision	Recall	F_1measure
C_1	0.921	0.824	0.869
C_2	0.889	0.943	0.915
C_3	0.939	0.827	0.879
C_4	0.964	0.834	0.894
C_5	0.834	0.953	0.889

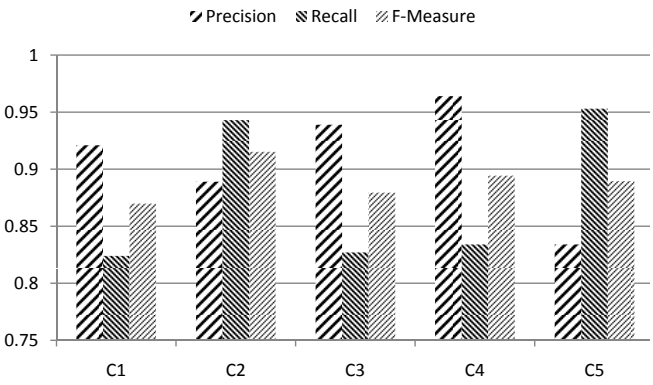

Figure 7. Comparison of Bayesian Classifiers.

All the Bayesian classifiers that we have tested achieved a precision of above 83% and a recall of above 82%. Minimum F_1 measure of 87% was reported by C_1, where the training is done with the original data set without resampling. The obvious reason for a low F_1 measure is the class imbalance, where the prior probability of a session to be Smart ClickBot is as high as 0.90. A model trained with such a training data set is biased towards Class 1.

By over sampling Class 2 records for C_3, and by under sampling Class 1 records for C_4 we have achieved a higher F_1-measure than that of C_1. In both of these cases we have tried to increase the class representation of Class 2 records.

The best F_1 measure is achieved by C_2 which was trained using oversampling of Class 2 so that samples reach the number of Class 1 in the original set, hence leaving balanced representation of classes.

In this experiment we have two cases where the class representation is equal. That is with classifiers C_2 and C_5. Training data for C_2 is created by oversampling Class 2 records, while training data for C_5 is created by under sampling Class 1 records. A similar recall values are reported by both C_2 and C_5. However, there is a significant difference between the precision values. Precision of C_5 is 83%, while that of C_2 is 89%. This means we have an increase in the number of false positives in C_5, i.e. Class 1 incorrectly classified as Class 2. The significant decrease in precision of C_5, is not surprising since, with random under sampling there is no control over which examples are eliminated from the original set. Therefore significant information about the decision boundary between the two classes may be lost. This is always a risk with random oversampling where it would do over-fitting due to placing exact duplicates of minority examples from the original set and thus making the classifier biased by "remembering" examples that were seen many times.

V. CONCLUSIONS

In this paper we presented a Bayesian approach for detecting the Smart ClickBot type clicks from server logs. The system combines evidence extracted from web server sessions to determine the final class of each click. Some of these evidences can be used alone, while some can be used in combination with other features for the click bot detection. During training and testing we also addressed the class imbalance problem. Our best classifier shows recall of 94%, and precision of 89%, with F_1 measure calculated as 92%. The high accuracy of our system proves the effectiveness of the proposed methodology. Since the Smart ClickBot is a sophisticated click bot that manipulate every possible parameters to go undetected, the techniques that we discussed here can lead to detection of other types of software bots too.

We have been carrying out similar studies for a while, for the benefit of the Internet community. It is worth mentioning that, we have to accept the fact that well-undercover bots will not be spotted and we have to reach a sustainable trade-off for the detection model. Any attempt to identify deep undercover bots is likely to produce high false positive rates. A high false positive rate will mean that we are labeling many regular traverses as bot activities. Although we wish to detect as many bots as possible, it is far worse to tag regular traffic as bots than to skip some detection.

REFERENCES

[1] Robots database, http://www.robotstxt.org/db.html, accessed on 03/21/2011.

[2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, 2001.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine* 1. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[4] S. Chakrabarti, M. Van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

[5] N. Daswani and M. Stoppelman. The anatomy of Clickbot. A. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 11–11. USENIX Association, 2007.

[6] M. Dikaiakos, A. Stassopoulou, and L. Papageorgiou. An investigation of web crawler behavior: characterization and metrics. *Computer Communications*, 28(8):880–897, 2005.

[7] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

[8] P. Huntington, D. Nicholas, and H. Jamali. Web robot detection in the scholarly information environment. *Journal of Information Science*, 34(5):726, 2008.

[9] M. Kantardzic, C. Walgampaya, and W. Emara. Click fraud prevention in pay-per-click model: Learning through multi-model evidence fusion. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pages 20–27. IEEE.

[10] M. Kantardzic, C. Walgampaya, B. Wenerstrom, O. Lozitskiy, S. Higgins, and D. King. Improving Click Fraud Detection by Real Time Data Fusion. In *Signal Processing and Information Technology, 2008. ISSPIT 2008. IEEE International Symposium on*, pages 69–74. IEEE, 2008.

[11] M. Kantardzic, C. Walgampaya, R. Yampolskiy, and R. Woo. Click Fraud Prevention via multimodal evidence fusion by Dempster-Shafer theory. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 26–31. IEEE.

[12] W. Lu and S. Yu. Web robot detection based on hidden Markov model. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 1806–1810. IEEE, 2006.

[13] A. Metwally, D. Agrawal, A. El Abbad, and Q. Zheng. On hit inflation techniques and detection in streams of web advertising networks. 2007.

[14] A. Metwally, D. Agrawal, and A. El Abbadi. Detectives: detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th international conference on World Wide Web*, pages 241–250. ACM, 2007.

[15] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[16] P. Pirolli, J. Pitkow, and R. Rao. Silk from a sow's ear: extracting usable structures from the Web. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pages 118–125. ACM, 1996.

[17] A. Stassopoulou and M. Dikaiakos. Web robot detection: A probabilistic reasoning approach. *Computer Networks*, 53(3):265–278, 2009.

[18] P. Tan and V. Kumar. Modeling of web robot navigational patterns. In *Second International Workshop on Challenges and Opportunities, WEBKDD 2000, Web Mining for E-Commerce*, 2001.

[19] P. Tan and V. Kumar. Discovery of web robot sessions based on their navigational patterns. *Data Mining and Knowledge Discovery*, 6(1):9–35, 2002.

[20] C. Walgampaya and M. Kantardzic. Netmosaics. In *Internal Documentation*, 2010.

[21] F. Yu, Y. Xie, and Q. Ke. SBotMiner: large scale search bot detection. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 421–430. ACM, 2010.