PROJECT NAME :

Click Fraud Detection on the Advertiser Site

horizontal line

# 

Guide :

* A.S.V Vandana Ma’am

Team :

1. Surram Manikanta Reddy - 18071A12H2.
2. Vunnam Sai Sandeep - 18071A12H6.
3. Yerraguntla Nitya Chowdary - 18071A12H9.
4. Manthena Sujith Varma - 18071A12J0.

ABSTRACT :

Click fraud—malicious clicks at the expense of pay-per-click advertisers—is posing a serious threat to the Internet economy. Although click fraud has attracted much attention from the security community, as the direct victims of click fraud, advertisers still lack effective defense to detect click fraud independently. In this paper, we propose a novel ap- proach for advertisers to detect click frauds and evaluate the return on investment (ROI) of their ad campaigns without the helps from ad net- works or publishers. Our key idea is to proactively test if visiting clients are full-fledged modern browsers and passively scrutinize user engage- ment. In particular, we introduce a new functionality test and develop an extensive characterization of user engagement. Our detection can sig- nificantly raise the bar for committing click fraud and is transparent to users. Moreover, our approach requires little effort to be deployed at the advertiser side. To validate the effectiveness of our approach, we imple- ment a prototype and deploy it on a large production website; and then we run 10-day ad campaigns for the website on a major ad network. The experimental results show that our proposed defense is effective in identifying both clickbots and human clickers, while incurring negligible overhead at both the server and client sides.

**Keywords :** Click Fraud, Online Advertising, Feature Detection.

Introduction :

In an online advertising market, advertisers pay ad networks for each click on their ads, and ad networks in turn pay publishers a share of the revenue. As online advertising has evolved into a multi-billion dollar business , click fraud has become a serious and pervasive problem. For example, the botnet “Chameleon” infected over 120,000 host machines in the U.S. and siphoned $6 million per month from advertisers .

Click fraud occurs when miscreants make HTTP requests for destination URLs found in deployed ads . Such HTTP requests issued with malicious intent are called fraudulent clicks. The incentive for fraudsters is to increase their own profits at the expense of other parties. Typically a fraudster is a pub- lisher or an advertiser. Publishers may put excessive ad banners on their pages and then forge clicks on ads to receive more revenue. Unscrupulous advertis- ers make extensive clicks on a competitor’s ads with the intention of depleting the victim’s advertising budget. Click fraud is mainly conducted by leveraging clickbots, hiring human clickers, or tricking users into clicking ads [4].

In an act of click fraud, both an ad network and a publisher are beneficiaries while an advertiser is the only victim, under the pay-per-click model. Although the ad network pays out to the publisher for those undetected click fraud ac- tivities, it charges the advertiser more fees. Thus, the ad network still benefits from click fraud. Only the advertiser is victimized by paying for those fraudu- lent clicks. Therefore, advertisers have the strongest incentive to counteract click fraud. In this paper, we focus on click fraud detection from the perspective of advertisers.

Click fraud detection is not trivial. Click fraud schemes have been continuously evolving in recent years . Existing detection solutions attempt to identify click fraud activities from different perspectives, but each has its own limitations. The solutions proposed in perform traffic analysis on an ad network’s traffic logs to detect publisher inflation fraud. However, an advanced clickbot can conduct a low-noise attack, which makes those abnormal-behavior-based detection mechanisms less effective. Haddadi proposed exploiting bait ads to blacklist malicious publishers based on a predefined threshold. Motivated by , Dave et al. proposed an approach for advertisers to measure click- spam ratios on their ads by creating bait ads. However, running bait ads increases advertisers’ budget on advertisements.

In this paper, we propose a novel approach for an advertiser to independently detect click fraud attacks conducted by clickbots and human clickers. Our ap- proach enables advertisers to evaluate the return on investment (ROI) of their ad campaigns by classifying each incoming click traffic as fraudulent, casual, or valid. The rationale behind our design lies in two observed invariants of legiti- mate clicks. The first invariant is that a legitimate click should be initiated by a real human user on a real browser. That is, a client should be a real full-fledged browser rather than a bot, and hence it should support JavaScript, DOM, CSS, and other web standards that are widely followed by modern browsers. The sec- ond invariant is that a legitimate ad clicker interested in advertised products must have some level of user engagement in browsing the advertised website.

Based on the design principles above, we develop a click fraud detection sys- tem mainly composed of two components:

* a proactive functionality test and
* a passive examination of browsing behavior.

The functionality test actually challenges a client for its authenticity (a browser or a bot) with the assumption that most clickbots have limited functionality compared to modern browsers and thus would fail this test. Specifically, a client’s functionality is validated against web standards widely supported by modern browsers. Failing the test would in- duce all clicks generated by the client to be labelled as fraudulent. The second component passively examines each user’s browsing behaviors on the advertised website. Its objective is to identify human clickers and those more advanced clickbots that may pass the functionality test. If a client passes the functionalitytest and also shows enough browsing engagement on the advertised website, the corresponding click is labelled as valid. Otherwise, a click is labelled as casual if the corresponding client passes the functionality test but shows insufficient browsing behaviors. A casual click could be generated by a human clicker or by an unintentional user. We have no attempt to distinguish these two since neither of them is a potential customer from the standpoint of advertisers.

To evaluate the effectiveness of the proposed detection system, we build a prototype and deploy it on a large production web server. Then we run ad campaigns at one major ad network for 10 days. The experimental results show that our approach can detect much more fraudulent clicks than the ad network’s in-house detection system and achieve low false positive and negative rates. We also measure the performance overhead of our detection system on the client and server sides.

Note that our detection mechanism can significantly raise the bar for commit- ting click fraud and is potentially effective in the long run after public disclosure. To evade our detection mechanism, clickbots must implement all the main web standards widely supported by modern browsers. And a heavy-weight clickbot will risk itself of being readily noticeable by its host. Likewise, human clickers must behave like real interested users by spending more time, browsing more pages, and clicking more links on the advertised sites, which contradicts their original intentions of earning more money by clicking on ads as quickly as pos- sible. At each point, the net effect is a disincentive to commit click fraud.

The remainder of the paper is organized as follows. We provide background knowledge in Section 2. Then, we detail our approach in Section 3 and validate its efficacy using real-world data in Section 4. We discuss the limitations of our work in Section 5 and survey related work in Section 6. Finally, we conclude the paper in Section 7.

Background :

Based on our understanding of the current state of the art in click fraud, we first characterize clickbots and human clickers, the two main actors leveraged to commit click fraud. We then discuss the advertiser’s role in inhibiting click fraud. Finally, we describe the web standards widely supported by modern browsers, as well as feature detection techniques.

Clickbots :

A clickbot behaves like a browser but usually has relatively limited functionality compared to the latter. For instance, a clickbot may not be able to parse all elements of HTML web pages or execute JavaScript and CSS scripts. Thus, at the present time, a clickbot is instantiated as malware implanted in a victim’s computer. Even assuming a sophisticated clickbot equipped with capabilities close to a real browser, its actual browsing behavior when connected to the advertised website would still be different from that of a real user. This is because clickbots are automated programs and are not sophisticated enough to see and think as human users, and as of yet, do not behave as human users.

A typical clickbot performs some common functions including initiating HTTP requests to a web server, following redirections, and retrieving contents from a web server. However, it does not have the ability to commit click fraud itself but instead acts as a relay based on instructions from a remote bot master to com- plete click fraud. A bot master can orchestrate millions of clickbots to perform automatic and large-scale click fraud attacks.

Illustrates how a victim host conducts click fraud under the com- mand of a botmaster. First, the botmaster distributes malware to the victim host by exploiting the host’s security vulnerabilities, by luring the victim into a drive-by download or running a Trojan horse program. Once compromised, the victim host becomes a bot and receives instructions from a command-and- control (C&C) server controlled by the botmaster. Such instructions may specify the target website, the number of clicks to perform on the website, the referrer to be used in the fabricated HTTP requests, what kind of ads to click on, and when or how often to click.

After receiving instructions, the clickbot begins traversing the designated pub- lisher website. It issues an HTTP request to the website (step 1). The website returns the requested page as well as all embedded ad tags on the page (step 2). An ad tag is a snippet of HTML or JavaScript code representing an ad, usually in an iframe. For each ad tag, the clickbot generates an HTTP request to the ad network to retrieve ad contents just like a real browser (step 3). The ad network returns ads to the clickbot (step 4). From all of the returned ads, the clickbot Clickselects an ad matching the specified search pattern and simulates a click on the ad, which triggers another HTTP request to the ad network (step 5). The ad network logs the click traffic for the purpose of billing the advertiser and paying the publisher a share, and then returns an HTTP 302 redirect response (step 6). The clickbot follows the redirection path (possibly involving multiple parties) and finally loads the advertised website (step 7). The advertiser returns back the landing page1 to the clickbot (step 8). At this point, the clickbot completes a single act of click fraud. Every time an ad is “clicked” by a clickbot, the adver- tiser pays the ad network and the involved publisher receives remuneration from the ad network. Note that a clickbot often works in the background to avoid raising suspicion, thus all HTTP requests in Figure 1 are generated without the victim’s awareness.

Human Clickers :

Human clickers are the people who are hired to click on the designated ads and get paid in return. Human clickers have financial incentives to click on ads as quickly as possible, which distinguishes them from real users who are truly interested in the advertised products. For instance, a real user tends to read, consider, think, and surf the website in order to learn more about a product before purchase. A paid clicker has few such interests, and hence tends to get bored quickly and spends little time on the site [12].

Advertisers :

Advertisers are in a vantage point to observe and further detect all fraudulent activities committed by clickbots and human clickers. To complete click fraud, all fraudulent HTTP requests must be finally redirected to the advertised website, no matter how many intermediate redirections and parties are involved along the way. This fact indicates that both clickbots and human clickers must finally communicate with the victim advertiser. Thus, advertisers have the advantage of detecting clickbots and human clickers in the course of communication. In ad- dition, as the revenue source of online advertising, advertisers have the strongest motivation to counteract click fraud.

Web Standards and Feature Detection Techniques :

The main functionality of a browser is to retrieve remote resources (HTML, style, and media) from web servers and present those resources back to a user. To correctly parse and render the retrieved HTML document, a browser should be compliant with HTML, CSS, DOM, and JavaScript standards which are rep- resented by scriptable objects. Each object is attached with features including properties, methods, and events. For instance, the features attached to the DOM object include createAttribute, getElementsByTagName, title, domain, url, and

Landing page is a single web page that appears in response to clicking on an ad.Outline of click fraud detection mechanism many others. Every modern browser supports those features. However, different browser vendors (and different versions) vary in support levels for those web standards, or they implement proprietary extensions all their own. To ensure that websites are displayed properly in all mainstream browsers, web develop- ers usually use a common technique called feature detection to help produce JavaScript code with cross-browser compatibility.

Feature detection is a technique that identifies whether a feature or capa- bility is supported by a browser’s particular environment. One of the common techniques used is reflection. If the browser does not support a particular fea- ture, JavaScript engines return null when referencing the feature; otherwise, JavaScript returns a non-null string. For instance, if the JavaScript statement “document.createElement” returns null in a specific browser, it indicates that the browser does not support the method createElement attached to the document object. Likewise, by testing a browser against a large number of fundamental features specified in web standards for modern browsers, we can estimate the browser’s support level for those web standards, which helps validate the au- thenticity of the execution environment as a real browser.

Feature detection techniques have three primary advantages. First, feature detection can be an effective mechanism to detect clickbots. A clickbot cannot “pass” the feature detection unless it has implemented the main functionality of a real browser. Second, feature detection stresses the client’s functionality thoroughly, and even a large pool of features can be used for feature detection in a fast and efficient manner. Lastly, the methods used for feature detection are designed to work across different browsers and will continue to work over time as new browsers appear, because new browsers fundamentally support reflection— even before implementing other features—and should also extend, rather than replace, existing web standards.

Methodology :

Our approach mainly challenges a visiting client and its user engagement on the advertised site to determine whether the corresponding ad click is valid or not. To maximize detection accuracy, we also check the legitimacy of the origin (client’s IP address) and the intermediate path (i.e., the publisher) of a click.

JavaScript Support & Mouse Event Test

Functionality Test

HTML DOM CSS JavaScript Standard Standard Standard Standard

Supported by

Fail Pass Fraudulent

Browser Behavior Examination

Behavioral classification

Valid/Casual

Click Fraud Detection Methodology

# total clicks

# total mouse moves

# pages viewed

visit duration ...

Mouse Events

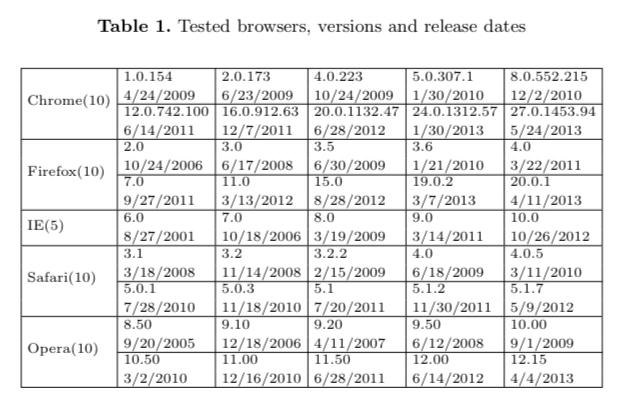
click, double click, mouse up, mouse down, mouse move, mouse over, provides an outline of our approach. Our detection system consists of three components: (1) JavaScript support and mouse event test, (2) browser functionality test, and (3) browsing behavior examination.

For each incoming user, on the landing page, we test if the client supports JavaScript and if any mouse events are triggered. No JavaScript support or no mouse event indicates that the client may not be a real browser but a click- bot. Otherwise, we further challenge the client’s functionality against the web standards widely supported by mainstream browsers. The client failed the func- tionality test is labelled as a clickbot. Otherwise, we further examine the client’s browsing behavior on the advertiser’s website and train a behavior-based classi- fier to distinguish a really interested user from a casual one.

JavaScript Support and Mouse Event Test :

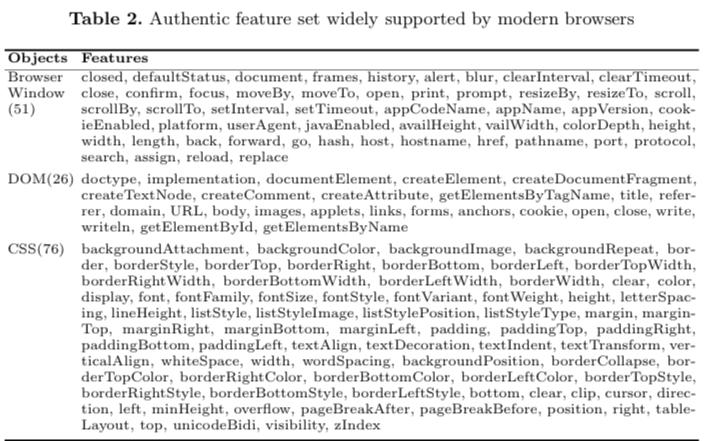
One simple way to detect clickbots is to test whether a client supports JavaScript or not. This is due to the fact that at least 98% of web browsers have JavaScript enabled [14] and online advertising services usually count on JavaScript support.

Monitoring mouse events is another effective way to detect clickbots. In gen- eral, a human user with a non-mobile platform (laptop/desktop) must generate at least one mouse event when browsing a website. A lack of mouse events flags the visiting client as a clickbot. However, this may not be true for users from mobile platforms (smartphones/pads). Thus, we only apply the mouse event test to users from non-mobile platforms.



Functionality Test :

A client passing the JavaScript and mouse event test is required to further un- dergo a feature-detection based functionality test.



To avoid false positives and ensure that each modern browser can pass the functionality test, we perform an extensive feature support measurement on the top 5 mainstream browsers [15]: Chrome, Firefox, IE, Safari, and Opera. To discern the consistently supported features, we uniformly select 10 versions for each browser vendor with the exception of 5 versions for IE. Table 1 lists the browsers we tested. As a result, we obtain a set of 153 features associated with web standards, including browser window, DOM, and CSS (see Table 2). All those features are supported by both desktop browsers and their mobile versions. These features are commonly and consistently supported by the 45 versions of browsers in the past ten years. We call this set the authentic-feature set. We also create a bogus-feature set, which has the same size as the authentic-feature set but is obtained by appending “123” to each feature in the authentic-feature set. Thus, every feature in the bogus-feature set should not be supported by any real browser. Note that we just use the string “123” as an example. When implementing our detection, the advertiser should periodically change the string to make the bogus-feature set hard to evade.

**How to Perform the Functionality Test.**

The first HTTP request issued by a client, the ad- vertiser’s web server challenges the client by responding as usual, but along with a mixed set of authentic and bogus features. While the size of the mixed set is fixed (e.g.,100), the proportion of authentic features in the set is randomly de- cided. Then, those individual authentic and forged features in the set are randomly selected from the authentic and bogus feature sets, respectively. The client is ex- pected to test each feature in its environment and then report to the web server how many authentic features are in the mixed set as the response to the challenge.A real browser should be able to report the correct number of authentic fea- tures to the web server after executing the challenge code, and thus passes the functionality test. However, a clickbot would fail the test because it is unable to test the features contained in the set and return the correct number. Considering some untested browsers may not support some authentic features, we set up a narrow range [x − N, x] to handle this, where x is the expected number and N is a small non-negative integer. A client is believed to pass the test as long as its reported number falls within [x − N, x]. Here we set N to 4 based on our measurement results.

Evasion Analysis. Assume that a client receives a mixed set of 150 features from a web server and the set consists of 29 randomly selected authentic features and 121 randomly selected bogus features. Thus, the expected number should fall into the range [25,29]. Consider a crafty clickbot who knows about our detection mechanism in advance. The clickbot does not need to test the features, but just guesses a number from the possible range [0,150], and returns it to the server. In this case, the probability for the guessed number to successfully fall into [25,29] is only 3%. Thus, the clickbot has little chance (3%) to bypass the functionality test.

JavaScript Support and Mouse Event Test :

Among the 9.9 thousand ad clicks logged by the advertised site, 75.2% of users do not support JavaScript. We labelled those users as clickbots. Note that this percentage may be slightly over- estimated considering that some users (at most 2% [14]) may have JavaScript disabled. In addition, those visits without support for JavaScript do not corre- late with visits from mobile browsers. We have checked that nearly all mobile browsers provide support for JavaScript despite limited computing power. We then focused on the top 10 publisher websites with the most clicks to iden- tify potentially malicious publishers. Figure 8 depicts the percentage of clicks without script support from those top 10 publishers. Among them, the two non- entertainment websites google.com and ask.com have low ratios, 9.4% and 15.2%, respectively. In contrast, the other 8 entertainment websites have quite high click ratios without script support. There are 86 visits from tvmao.com and none of them support JavaScript. We believe that all 86 clicks are fraudulent and gener- ated by bots. Similarly, 99.1% of clicks from weaponsgames.com, 96.1% of clicks from 3dgames.org, and 95.3% from gamesgirl.net are without JavaScript support either. Such high ratios indicate that the invalid click rate in the real-world ad campaigns is much larger than the average invalid rate of 25.15% alleged by the ad network for our ad campaigns, as shown in Table 3.

We observed 506 ad clicks (with JavaScript support) that result in zero mouse events when arriving at our target site. Of those, 96 are initiated from mobile platforms including iPad, iPhone, Android, and Windows Phone.

Discussion and Limitations :

In this paper, we assume that a clickbot typically does not include its own JavaScript engine or access the full software stack of a legitimate web browser re- siding on the infected host. A sophisticated clickbot implementing a full browser agent itself would greatly increase its presence and the likelihood of being de- tected. A clickbot might also utilize a legitimate web browser to generate activi- ties, and can thus pass our browser functionality test. To identify such clickbots, we could further detect whether our ads and the advertised websites are really visible to users by utilizing a new feature provided by some ad networks. The new feature allows advertisers to instrument their ads with JavaScript code for a better understanding of what is happening to their ads on the client side. With this feature, we could detect if our ad iframe is visible at the client’s front-end screen rather than in the background, and if it is really focused and clicked on.

In addition, compared to our user-visit related features (dwelling time, mouse events, scroll events, clicks and etc.), user-conversation related features3 are ex- pected to have better discriminating power between clickbots, human clickers, and real users in browsing behaviors. However, our advertised site is a profes- sional forum rather than an online retailer. If a user registers (creates an account) on the forum, it is analogous to a purchase at an online retailer. However, such conversion from guest to member is an event too rare to rely upon to enhance our classifier.

Related Work :

**Browser Fingerprinting.** Browser fingerprinting allows a website to identify a client browser even though the client disables cookies. Existing browser fin- gerprinting techniques could be mainly classified into two categories, based on the information they need for fingerprinting. The first category fingerprints a browser by collecting application-layer information, including HTTP request header information and system configuration information from the browser [23]. The second category performs browser fingerprinting by examining coarse traffic generated by the browsers [24]. However, both of them have their limitations in detecting clickbots. Nearly all the application-layer information can be spoofed by sophisticated clickbots, and browser fingerprints may change quite rapidly over time [23]. In addition, an advertiser often cannot collect enough traffic information for fingerprinting the client from just one visit to the advertiser. Compared to the existing browser fingerprinting techniques, our feature detec- tion technique has three main advantages. First, clickbots cannot easily pass the functionality test unless they have implemented the main functionality present in modern browsers. Second, the client’s functionality could be tested thoroughly at the advertiser’s side even though the client visits the advertiser’s landing page only once. Lastly, our technique works over time as new browsers appear because new browsers should also conform to the those web standards currently supported by modern browsers.

**Revealed Click Fraud.** Several previous studies investigate known click fraud activities, and clickbots have been found to be continuously evolving and become more sophisticated. As the first study to analyze the functionality of a clickbot, Daswani et al. [3] dissected Clickbot.A and found that the clickbot could carry out a low-noise click fraud attack to avoid detection. Miller et al. [5] exam- ined two other families of clickbots. They found that these two clickbots were more advanced than Clickbot.A in evading click fraud detection. One clickbot introduces indirection between bots and ad networks, while the other simulates human web browsing behaviors. Some other characteristics of clickbots are de- scribed in [4]. Clickbots generate fraudulent clicks periodically and only issue one fraudulent click in the background when a legitimate user clicks on a link, which makes fraudulent traffic hardly distinguishable from legitimate click traf- fic. Normal browsers may also be exploited to generate fraudulent click traffic. The traffic generated by a normal browser could be hijacked by currently visited malicious publishers and be further converted to fraudulent clicks [7]. Ghost click botnet [6] leverages DNS changer malware to convert a victim’s local DNS re- solver into a malicious one and then launches ad replacement and click hijacking attacks. Our detection can identify each of these clickbots by actively performing a functionality test and can detect all other kinds of click fraud by examining their browsing behavior traffic on the server side.

**Click Fraud Detection.** Metwally et al. conducted an analysis on ad networks’ traffic logs to detect publishers’ non-coalition hit inflation fraud [8], coalition fraud [9], and duplicate clicks [10]. The main limitation of these works lies in that adnetworks’ traffic logs are usually not available to advertisers. Haddadi in [11] and Dave et al. in [4] suggested that advertisers use bait ads to detect fraudulent clicks on their ads. While bait ads have been proven effective in detection, advertisers have to spend extra money on those bait ads. Dave et al. [16] presented an ap- proach to detecting fraudulent clicks from an ad network’s perspective rather than an advertiser’s perspective. Li et al. [7] introduced the ad delivery path related fea- tures to detect malicious publishers and ad networks. However, monitoring and reconstructing the ad delivery path is time-consuming and difficult to detect click frauds in real time. Schulte et al. [25] detected client-side malware using so-called program interactive challenge (PIC) mechanism. However, an intermediate proxy has to be introduced to examine all HTTP traffic between a client and a server, which would inevitably incur significant delay. Like [4, 11], our defense works at the server side but does not cause any extra cost for advertisers. Our work is the first to detect clickbots by testing their functionalities against the specifications widely conformed to by modern browsers. Most clickbots can be detected at this step, because they have either no such functionalities or limited functionalities compared to modern browsers. For the advanced clickbots and human clickers, we scrutinize their browsing behaviors on the advertised site, extract effective fea- tures, and train a classifier to identify them.

Conclusion :

In this paper, we have proposed a new approach for advertisers to independently detect click fraud activities issued by clickbots and human clickers. Our proposed detection system performs two main tasks of proactive functionality testing and passive browsing behavior examination. The purpose of the first task is to detect clickbots. It requires a client to actively prove its authenticity of a full-fledged browser by executing a piece of JavaScript code. For more sophisticated click- bots and human clickers, we fulfill the second task by observing what a user does on the advertised site. Moreover, we scrutinize who initiates the click and which publisher website leads the user to the advertiser’s site, by checking the legitimacy of the clients’ IP addresses (source) and the reputation of the re- ferring site (intermediate), respectively. We have implemented a prototype and deployed it on a large production website for performance evaluation. We have then run a real ad campaign for the website on a major ad network, during which we characterized the real click traffic from the ad campaign and provided advertisers a better understanding of ad click traffic, in terms of geographical distribution and publisher website distribution. Using the real ad campaign data, we have demonstrated that our detection system is effective in the detection of click fraud.