Advertisement Click Fraud Detection

Problem Statement

On-line advertisement has become one of the most important funding models to support Internet sites. Given that large sums of money are involved in on-line advertisement; malicious parties are unfortunately attempting to gain an unfair advantage. Click-fraud attacks are one instance of such malicious behavior, where software imitates a human clicking on an advertisement link.
As an example strategy, we present Bluff ads; a class of ads that join forces in order to increase the effort level for click-fraud spammers. Bluff ads are either targeted ads, with irrelevant display text, or highly relevant display text, with irrelevant targeting information. They act as a litmus test for the legitimacy of the individual clicking on the ads. Together with standard threshold-based methods, fake ads help to decrease click-fraud levels.
Traditionally, online advertisers used the Cost-Per-Impression (CPI) model to charge for advertisements. The CPI is often measured in terms of the cost of one thousand impressions of the ad. These ads can be videos, images and links. Recently, search engines such as Google have given rise to the Pay-Per-Click (PPC) model for online advertising. In this type of arrangement, advertisers pay a certain amount of money to the publisher for every click on their ad (which leads to the advertiser’s website). This model has however led to the rise of click fraud where the publisher, his employees or a leased botnet of computers fraudulently click on the ads in order to generate revenue for the publisher in an undetectable manner. Another forms of click fraud happens when competitors of an advertiser click on his ads in order to deplete his daily budget. In another scenario, some advertisers or publishers may cooperate against a single publisher by clicking on his ads, in order to force the broker to exclude the publisher from the ad campaigns.

Background

There are a number of solutions for avoiding click fraud and performing better advertisement. One suggestion is to charge based on user’s actions, i.e., the publisher gets a premium only after the successful conversion of the ad, meaning the user’s visit to the advertiser website and performing an action such as buying an item or signing up for a service. There are a number of basic attempts at such an approach by means of tracking cookies, however these efforts make up a negligible portion of the current advertising revenue on the Internet.
Cryptographic approach for replacing the pay-per-click model with one in which pay-per action (e.g., shopping) can attract premium rates and unsuccessful clicks are discarded. in this system, the users which make a purchase are identified by the network of advertisers as premium advertisers. The client browsers use a coupon instantiated by third party cookies or issued by the attestor upon redirection. The disadvantage of this method is the ability of malicious attacker, possibly an advertiser, to use a botnet and replay the coupons numerous times, for a large number of cooperating publishers. This will then force the syndicator either discount all those replays, or removing those clients form the system with valid coupons. In both cases, the advertisement income is minimized. It also allows for the syndicator and the attestor (ad broker and middle box) to profile the users accurately including their spending budget.
click fraud learning algorithms to compute the estimated click-through rate. They focus on a situation in which there is just one ad slot, and show that fraudulent clicks can not increase the expected payment per impression by more than o(1) in a click-based algorithm. However the complexity of the inferred algorithm and the need for click-through rate estimation would make it impractical as it also deviates form the pay per click model, to pay per view model, which is the least desired model in the modern advertisement world where bidding for space is of critical importance.

Methodology

There is an ongoing industry-wide effort to develop tools that will effectively detect and block many common click-fraud attacks. Most of the attacks discovered and reported so far have been

malware-based attacks that rely on automated scripts,25 individuals hired by competitors,26 or proxy servers used to generate clicks for paid advertisements. Companies like AdWatcher27 and ClickProtector28 have initiated efforts to counter this. The essence of their approach is to track IP addresses of machines generating the clicks, as well as identifying the domain from which the clicks are registered. By collecting large logs and performing expert analysis, irregularities such as repeated number of clicks for a certain advertisement from a particular IP address, a particular domain, or abnormal spikes in traffic for a specific web site are identified. However, as described previously, the stealthy attack described in this article will go undetected by any of these tools. It is therefore of particular importance to determine other unique mechanisms of detecting and preventing attacks of this nature. These can be divided into two classes: active and passive. Our proposal for the former is intended to detect clickfraud attempts housed on web pages that users intentionally navigate to (whether they wanted to go there or were deceived to think so), whereas the proposal for the latter is suited for detection of email-instigated click-fraud.
An active client-side approach interacts with search engines, performs popular searches, and visits the resulting sites. It also spiders through sites in the same manner as users might. To hide its identity, such an agent would not abide the robots.txt conventions and so would appear as an actual user to the servers it interacts with. The agent would act like a user as closely as possible, including occasionally requesting some advertisements; it would always verify that the number of ad calls that were made correspond to the number of requests that a human user would perceive were made. (The latter is to detect click-fraud attempts in which a large number of ad requests are made after a user initiates a smaller number of actual requests.)
A passive client-side approach observes the actions performed on the machines of the person appearing to perform the click. This may be done by running all JavaScript components in a virtual machine (appearing to be a browser) and trapping the requests for advertisements that are made. Any web page that causes a call of a type that should only be made after a click occurred can be determined to be fraudulent. While this takes care of the type of automated click-fraud described herein, it would not defend against a version that first causes a long (and potentially random) delay, and then commits clickfraud unless the virtual machine allows randomly selected scripts to run for significant amounts of time, hoping to trap a delayed call. We note that excessive delays are not in the best interest of the fraudster, as his target may close the browser window and therefore interrupt the session before a click is made. Passive client-side solutions may be housed in security toolbars or by anti-virus softwares. • Another type of passive solution is an infrastructure component. This would sift traffic, identify candidate traffic, and emulate the client machine that would have received the packets in question, with the intent of identifying click-fraud. A suitable application might be an ISP-level spam filter or an MTA. Before emails are delivered to recipients, they could be delivered in virtual machine versions of the same, residing on the infrastructure component, but mimicking the client machine.
For all of the above solutions, it should be clear that it is not necessary to trap all abuse. Namely, even if a rather small percentage of abuse is detected, this would betray the locations that house click-fraud with a high likelihood that increases with the number of users that are taken to one and the same fraud-inducing domain.

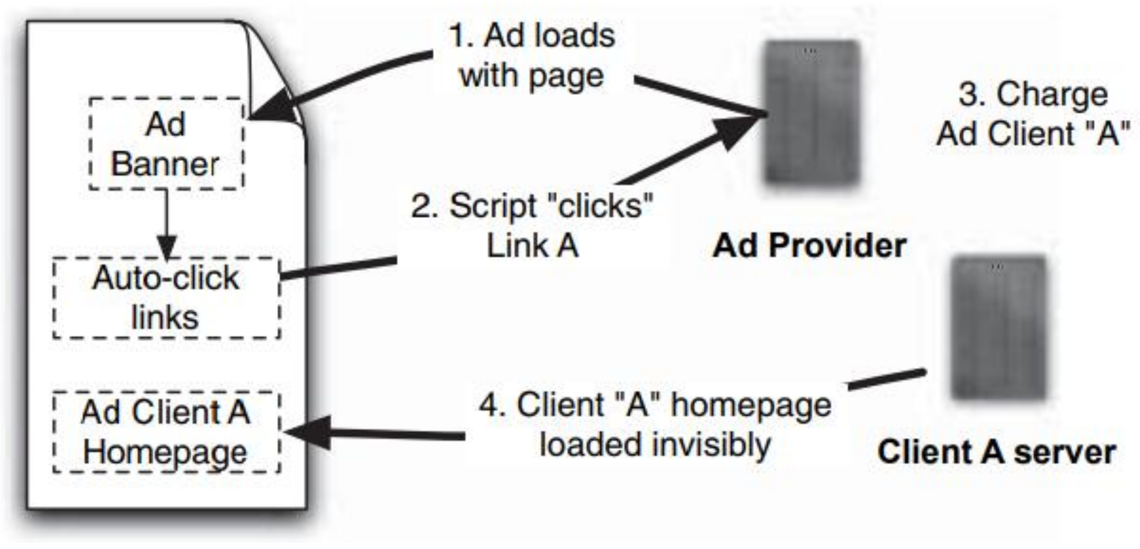

Fig: 2 Auto-clicking in a hidden badadvertisement. Compared to Figure 3; the ad banner presented here is hidden. JavaScript code extracts links from the hidden ad banner and causes them to be displayed in an another hidden iframe, creating the impression that the user has clicked these links.

Experimental Design

Suppose that the variables here combine to yield p ∈ {1/1000,1/10,000,1/100,000}. The analysis for given hypothetical detection probabilities in shown in Figure 7. Note that even a modest detection probability such as p=1/10,000 limits potential profits considerably

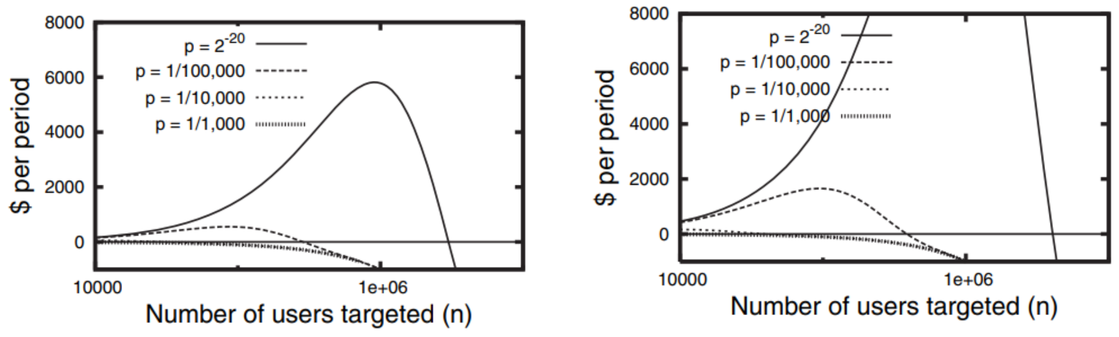

Fig: 1 Benefit for fraudster, in dollars earned, given several probabilities p of detection by ad providers. Note that when p=1/ 1000, the profit hardly goes above zero; when p=1/10,000, profit tops out at around $500 per domain; after that, the risks quickly outweights the rewards. Once p shrinks below 1/10,000, however, the fraudster fares much better — this is the current threat situation as far as we can tell. (a) Benefit when the reward per click is $0.25; (b) considers the reward per click to be $1.00.