

On Hit Inflation Techniques and Detection in Streams of Web Advertising Networks *

Ahmed Metwally^{†‡} Divyakant Agrawal[†] Amr El Abbadi[†]
{metwally, agrawal, amr}@cs.ucsb.edu

Qi Zheng[‡]
jerry@fastclick.com

Abstract

Click fraud is jeopardizing the industry of Internet advertising. Internet advertising is crucial for the thriving of the entire Internet, since it allows producers to advertise their products, and hence contributes to the well being of e-commerce. Moreover, advertising supports the intellectual value of the Internet by covering the running expenses of the content publishers' sites. Some publishers are dishonest, and use automation to generate traffic to defraud the advertisers. Similarly, some advertisers automate clicks on the advertisements of their competitors to deplete their competitors' advertising budgets. In this paper, we describe the advertising network model, and discuss the issue of fraud that is an integral problem in such setting. We propose using online algorithms on aggregate data to accurately and proactively detect automated traffic, preserve surfers' privacy, while not altering the industry model. We provide a complete classification of the hit inflation techniques; and devise stream analysis techniques that detect a variety of fraud attacks. We abstract detecting the fraud attacks of some classes as theoretical stream analysis problems that we bring to the data management research community as open problems. A framework is outlined for deploying the proposed detection algorithms on a generic architecture. We conclude by some successful preliminary findings of our attempt to detect fraud on a real network.

1. Introduction

As the Internet thrives, Internet advertising flourishes as the perfect choice for small as well as large businesses [8, 17, 32]. In contrast to conventional TV/Radio advertising that tends to be expensive, broadcast-based, and diluted; Internet advertisements are targeted to the appropriate customer base on the fly. An Internet advertiser provides an advertising commissioner with its advertisements, allocates a budget, and sets a commission for each customer action, such as clicking an advertisement, bidding in an auction, or

making a purchase. The Internet publishers, motivated by the commission paid by the advertisers, contract with the commissioner to display advertisements on the publishers' sites. Clearly, the main orchestrators in this setting are the Internet advertising commissioners, who are positioned as the brokers between publishers and advertisers, and whose servers are the backstage for all the targeting and budgeting.

Commissioners keep track of the advertisements previously shown to each surfer in a cookie. Frequency-capping [26] is a process which limits the number of exposures of one advertisement to a surfer per a specified time period. This avoids showing repetitive advertisements to the same surfer, which is boring, and also gives the advertisers better exposure. Whenever a surfer visits a publisher's site, the surfer is referred to the servers of the commissioner. Among the advertisements not shown recently for that surfer, the commissioner picks the advertisement that maximizes the commissioner's returns to be displayed on the publisher's site, and logs the impression (advertisement rendering) for accounting purposes. Whenever a surfer clicks an advertisement on a publisher's site, the surfer is referred to the commissioner's servers, which log the click, and clicks-through the surfer to the advertiser's site.

Since publishers are paid by the traffic they drive to advertisers, there is an incentive for fraudulent publishers to inflate the number of impressions and clicks their sites generate [2, 4, 14, 16, 20, 22, 23, 25, 27, 29, 34, 35]. In real life, we notice some fraudsters, disguised as publishers, mirror some sites under different domains, sign up online with the commissioners for those sites, provide a PayPal® account for anonymity, and start simulating traffic and earning revenue. In addition, dishonest advertisers tend to simulate clicks on the advertisements of their competitors to deplete the competitors' advertising budgets [20, 29]. Failing to identify fraudulent traffic in real time results in bad reputation for the commissioner, incurs high accounting overhead, and may entail paying forfeitures for advertisers [18, 19].

In this work, we focus primarily on publishers' fraud, since the discussion can be generalized to advertisers' fraud, and we refer specifically to advertisers' fraud whenever necessary. Through the sequel, we concentrate on fraud detec-

*Supported by NSF under grants IIS 02-23022, and CNF 04-23336.

[†]Department of Computer Science, University of California, Santa Barbara, CA 93106.

[‡]FastClick, Inc., a ValueClick company. 360 Olive Street, Santa Barbara, CA 93103.

tion, and thus, assume that the source IP addresses of the traffic are not spoofed. Spoofing is an area of research that is being studied in the network security community [10]. In addition, we assume the internal access restriction policies deter insider attacks from employees of the commissioners.

The complementary problem of *hit shaving* has been studied by Reiter, Anupam, and Mayer [30]. *Hit shaving* is the fraud performed by an advertiser, who does not pay commission on some of the traffic received from publishers, by claiming that it received less traffic than the true numbers. In this paper, we focus primarily on *hit inflation* attacks, which are the more prevalent form of fraud [34].

This paper lays the ground for making a direct utilization of statistical data analysis approaches in comprehensively detecting advertising networks fraud. Discovering patterns in traffic streams reveals malicious intentions. We develop a compendious framework for deploying data analysis in fraud detection. We classify the publishers' *hit inflation* fraud attacks into *non-coalition* and *coalition* attacks. The former class comprises attacks that are performed by a single publisher. Whereas *coalition* attacks involve a coalition among fraudsters and can be subtler. We devise detection algorithms for several fraud attacks, and abstract detecting the fraud attacks of some classes as open theoretical stream analysis problems. Finally, we describe how to deploy the detection techniques on a generic architecture, and report our successful preliminary findings on a real network.

2. State of the Art

Falsely increasing the number of impressions and clicks has been a concern to advertising commissioners since their conception [35], especially since real life fraud techniques are rarely published [2, 16]. Classical fraud detection depends on detecting uncommon trends and spikes in traffic. However, fraudulent publishers have become smarter, have developed traffic that resembles real traffic, and the fight has grown into an arms race. Meanwhile, several cryptographic techniques have been proposed in the literature to combat this problem. Yet, these techniques are not used in practice since they compromise surfers' privacy. This represents the conflict between security and the individuals' privacy.

One of the main challenges involved in detecting such fraudulent behavior is putting an end to the arms race between the fraudulent publishers and the commissioners, while maintaining the privacy of the web users. In this paper, we depend on data analysis techniques that can be executed on an aggregate level such that the individuals' identities are not revealed, but still enabling the data analysis algorithms to achieve satisfactory levels.

2.1. The Classical Approach

Classical fraud detection judges publishers based on the quality of the traffic visiting their sites. The quality of a click or an impression is an estimate of the probability that

it leads to a sale [24]. The quality of traffic is estimated by its conformity with the network norms. The most effective classical metric is monitoring the *Click Through Rate (CTR)*¹. The *CTR* of an advertisement is consistent on sites of the same nature [16]. Conversely, advertisements of different nature have different *CTRs* on the same site. Automating site visits and clicks on advertisements not only deviates the *CTRs* of the displayed advertisements from the norms, but also yields similar *CTRs* for the advertisements on the site. Monitoring the behavior of advertisements can be developed further. Commissioners proactively load empty advertisements (blank images) on the publishers' sites, and check for clicks on those fake advertisements.

Classical fraud detection depends on the commissioners' edge in knowing the network-wide behavior of advertisements. Supposedly, this knowledge is only available to commissioners, since the traffic model drives HTTP requests of impressions and clicks from publishers' pages directly to commissioners. Hence, it conceals from publishers the advertisements loaded and clicked on their own sites.

The classical approach, however, lacks effectiveness. First, fraudsters can deceive the classical tools. For various reasons, publishers build their sites in a way that reveals the advertisements loaded and clicked. For instance, all impressions and clicks on CNN.com are redirected to a pool of servers operated by CNN before reaching the commissioner. This architecture allows fraudsters to sample the *CTRs* of the advertisements, including any empty fake advertisements, whose *CTRs* are $\approx 0\%$. Hence, fraudsters can automate traffic whose metrics comply with real traffic.

Second, the classical approach had been historically developed before Internet advertising reached maturity. Consequently, it overlooks the conflict of interest between commissioners, and advertisers. Traffic that does not comply with the network-wide metrics, can be legitimate, though of poor quality. For instance, advertisements on kids and games sites have similar and high *CTRs*, though clicks on kids' sites rarely lead to sales. Aggressively discounting low-quality traffic conserves the advertisers' budgets, though "underpays" both honest publishers and commissioners for traffic their servers delivers. The commissioner deserves to be paid for serving non-inflated traffic. However, the classical approach discards legitimate low-quality traffic, since it cannot detect malicious intentions.

2.2. The Cryptographic Approach

The main challenge is to distinguish fraudulent traffic from normal traffic. Asking for the cooperation of surfers, as proposed by the cryptographic approaches [4, 27], is neither scalable nor practical and has had no viable impact on the industry. Such approaches entail changing the industry model, and making the advertising network non-transparent

¹The *CTR* of a publisher, advertiser, or advertisement is the number of clicks it receives as a percentage of impressions.

to the surfer. Even more, to be effective, they require the commissioners to uniquely identify surfers. This can only be done by having the surfers reveal personal information, which compromises surfers' privacy. Detecting fraudulent behavior while maintaining the privacy of the web surfers is an intriguing problem. This represents the standard conflict between security and the citizens' privacy.

3. The Data Analysis Approach

Data analysis tools can potentially alleviate the drawbacks of the cryptographic approach. Since advertising commissioners face the dilemma of preserving surfers' privacy versus fraud detection, they can only perform statistical analysis on aggregate data using temporary surfers' identification. Commissioners may track individual surfers by their IP addresses and by distinct customer IDs they assign in cookies. Using cookies and IPs is the least intrusive method for the current Internet architecture compared to the techniques that require user login. Cookies store no personal information, and they can be accepted, blocked, or periodically cleared [21]. An IP is usually assigned to a surfer temporarily, and could be shared by several surfers. Hence, employing data analysis techniques on cookie IDs and IPs does not alter the industry model; and can be executed on an aggregate level to further obfuscate identities, but still detect fraud satisfactorily.

Statistical data analysis can be done by storing the entire traffic data in a database and periodically executing aggregate SQL queries to identify outliers that would be candidates for fraudulent behavior. Nevertheless, there is a challenging scale problem here. Since an average-sized commissioner receives around 70M records per hour, storing the traffic, and periodically running queries on the stored traffic to detect malicious patterns is very expensive. To cope with real-time traffic, a fraud detection system should allocate no more than $50\mu\text{s}$ to process each traffic entry, which allows only fast in-memory operations. In addition, debiting advertisers' accounts, and then crediting them if fraud is detected, is wasteful and makes fraud detection non-transparent to advertisers. Therefore, *hit inflation* detection is a quintessential application for streaming and sampling algorithms.

Recently, we have developed several online algorithms for detecting specific attacks. In [22], we proposed a simple Bloom-based [3] algorithm to detect duplicates in a stream of impressions or clicks. The algorithm uncovers a primitive *hit inflation* attack where a script continuously simulates clicks on advertisements. Experiments on real data were revealing. More than 27% of the clicks were suspicious. Interestingly, one of the advertisements was clicked 10,781 times by the same cookie ID in one day. In [23], a detection algorithm was proposed for the sophisticated *hit inflation* attack identified in [2], which involves a coalition of dishonest publishers. The detection algorithm requires the collaboration with Internet Service Providers (ISPs) and

entails identifying associations in an HTTP stream. This work was generalized in [25] to detect all attacks involving publishers' coalitions, as discussed in Section 5.

Interestingly, statistical techniques [22, 23, 25] identifies specific patterns and correlations that characterizes fraudulent traffic [20]. Hence, this approach reveals malicious intentions behind inflated traffic. Monitoring traffic quantity complements the classical tools that monitors traffic quality.

To circumvent the traffic data analysis techniques, publishers need to dilute the strong correlation between their sites and the machines from which the attacks are launched. This is done either on the side of the attacking machines or on the side of the publishers' sites. To dilute the correlations on the machines' side, fraudsters exploit the limitations in tracking customers' machines by obliterating or frequently changing the identification of the machines. Or else, fraudsters have to use many attacking machines to launch the attack. It will become clear in the sequel that the more IPs and cookie IDs the fraudster controls, the more difficult it is to detect the fraud. On the other hand, to dilute the correlations on the sites' side, fraudsters use the same machines to generate fraudulent traffic for several sites.

This naturally categorizes any *hit inflation* attack as a *non-coalition* attack, a *coalition* attack, or a blend of both. A *non-coalition* attack is performed by one fraudster who generates traffic from some machine(s), possibly while obliterating or frequently changing the identification of the machine(s). On the other hand, a *coalition* attack is performed by many fraudsters sharing the identification of their machines. In order to stop the arms race between fraudsters and commissioners for *non-coalition* attacks, we have to guarantee inclusiveness of all attacks in our classification, and researching effective solutions for all the classes. If both categories of attacks can be detected, then we have solved the *hit inflation* problem.

4. A Classification of Non-Coalition Attacks

We develop our classification of *non-coalition* attacks based on the way surfers' machines are identified, i.e., on the basis of the IP and the cookie ID of the machine. The normal surfers' traffic that visits a publisher's site encodes statistical relationship between the site, the IPs and cookie IDs. Since fraudsters do not control a large portion of surfers' machines, any attack leaves a fingerprint on the relationship between the IPs and cookie IDs. The imbalance in the relationship between the IPs and the cookie IDs is what we, as fraud detectives, look for to select a subset of the publishers for further manual investigation.

The attack either comes from one or several IPs. The number of cookies used on the machines used in the attack can be none, one or more. Therefore, there are six possible combinations of IPs and cookie IDs used in the attacks. Table 1 summarizes the combinations, and the fingerprint to look for in order to discover each class of attacks.

IP-Cookie Relationship		Cookie ID		
		0	1	Many
IP	1	High Percentage of Cookie-less Traffic	Duplicates and u -Duplicates	Unbalanced activities of IPs and Publishers
	Many		A cookie occurring in many IPs	Random: A cookie occurring in many IPs Pre-assigned: Unbalanced activities of IPs and Publishers

Table 1. Non-Coalition Attacks Classification.

4.1. Cookie-Less Attacks

Cookie-less attacks fit into the first column of the table. There are two main techniques to launch cookie-less attacks. The fraudster can either disable the cookies on the machine(s) used for the attacks, or use network anonymization. Network anonymization services are commercially available through www.anonymizer.com, www.tor.eff.org, and others. Those services were designed to protect surfers' privacy. Hence, they block third party cookies [5]. Such publishers would have a high percentage of their traffic being cookie-less. This can be easily detected by monitoring the percentage of cookie-less traffic for each publisher and investigating publishers that clearly deviate from the norm.

4.2. Single Cookie and Single IP Attacks

To disguise the high percentage of cookie-less traffic, a fraudster can run a simple script, with no cookie manipulation, on one machine connected to the Internet through one fixed IP. An example of such a script is given in [16]. Naïvely, this is a problem of duplicate detection of impressions and clicks [22]. Unfortunately, this is not necessarily the case. Since advertisement networks display different advertisements for consecutive visits by the surfer [28], the script can visit the site multiple times and take advantage of this *dynamic advertisements placement*. Hence, the commissioner cannot define two consecutive clicks, with the same IP and cookie ID, as duplicates if they are on different advertisements on the same page, since it could be a real surfer who is interested in both advertisements. The commissioner needs to set a limit on the acceptable rate of impressions and clicks from one pair of cookie ID and IP on the same site even if several advertisements are involved.

We propose using an extension of our Bloom based solution [22] to report a click or an impression on a site only after being repeated several times, u , within a specific period, t . The *u -Duplicate-Detector* algorithm employs M integers, instead of M bits, as well as d independent hash functions. This is very close to the counting Bloom filters in [13]. To report a pair of an IP and a cookie ID that has made more than u clicks, each integer should be capable of counting up to $u + 1$. Initially all the integers are initialized to 0. Every time a click is observed, it is hashed using the independent hash functions, and the integers corresponding to the results of the hash functions are checked. The algorithm should alert on the last click only if all the integers have values equal to u . Otherwise, the click has not yet oc-

curred u times. After checking the click, it is inserted into the integer-vector by incrementing all the integers that correspond to the results of the hash functions.

The errors of this scheme are only false positives. That is, it can only err by judging that a click has occurred u times, while the click has really occurred less than u times. However, the scheme does not miss any real u -duplicates. A detailed error analysis is available at [24].

We give a sense of the scale challenge from our experiments using a Pentium IV 2.66GHz PC. Assuming the data structure fits into memory, updating a bloom filter of IPs using 10 simple hashing functions (based on linear permutation [6]) requires around $2.5\mu s$. However, the complete click identification, which is much larger than the 4 Bytes of the IP, comprises the IP (4 Bytes), the cookie ID (64 Bytes), and the site ID (4 Bytes). Instead of IPs only, if the 72 Bytes of the complete click identification is used, updating the Bloom filter requires $187\mu s$ on average per traffic entry, which cannot cope with real-time traffic that entails processing every traffic entry in $50\mu s$.

4.3. Single Cookie and Multi-IPs Attacks

Since it is easier for an attacking publisher to change the cookie than to change the IP of the attacking machines, this class of attack is not widespread among publishers. However, we discuss it here, since it is crucial for discussing attacks with multiple cookies and multiple IPs in Section 4.5.

Formally, we need to discover cookies that appear with more than p IPs in a specified time period, t . Since both the population of the IPs and the cookies are huge, from a stream analysis perspective, it is challenging to identify cookies that appear with an excessive number of IPs. In its full generality, this is a hard problem. However from a pragmatic point of view, this problem can be solved.

Solutions to this problem can be developed along the following lines. The IPs are stored on the cookies with the timestamps of the impressions. Every time an impression request arrives, the commissioner replaces the cookie reported with a new cookie. The new cookie is the same as the old one, with the IP and the timestamp of the request appended. Hence, the commissioner can check the number of IPs associated with the cookie making the request, and alerts if more than p IPs exist in the last specified period, t .

The problem with this solution is that the commissioner has to trust the data stored in the cookies. Bearing in mind that the cookies are stored on machines that could belong to a dishonest publisher, this assumption is unrealistic since the fraudster can manipulate the cookies stored on the attacking machines. One way around this problem is to store a hash-based checksum on the cookie that would be violated if the cookie was modified to delete some IPs. However, this does not safeguard against *cookie-replay* attacks [15].

Overcoming Cookie-Replay Attacks. A *cookie-replay* is where the same unmodified cookie is sent repeatedly to

the server. In our context, the fraudster stores a copy, *old-cookie*, of the cookie, before sending it to the commissioner. When the commissioner appends the IP to the cookie and sends it back to the machine, the fraudster replaces the modified copy by *old-cookie*. Next time, the fraudster sends *old-cookie* again to the commissioner and the same scenario is repeated. Effectively, the fraudster gets around the commissioner's technique of appending IPs to the cookie.

Cookie-replay was discussed in [15] in the context of identity theft by eavesdropping the authentication traffic. Several solutions for the *intercept-replay* scenario were provided in [15] to prevent eavesdropping, and thus, do not apply in our case. Since the cookies are stored on the attacking machines, the fraudster needs no eavesdropping.

We now propose a simple solution for the *cookie-replay* attack in advertising networks, which can also be used in other contexts. The solution lies in storing, in addition to the set of IPs and the checksum, a counter on the cookie. The counter is incremented every time the commissioner receives the cookie, and altering the counter violates the checksum stored on the cookie. A duplicate detection technique is employed to detect duplicates in the stream of cookies making the requests. However, the d hash functions of the duplicate detection mechanism are seeded with the counter of the hashed cookie. Thus, a cookie is detected as a duplicate and the impression/click will be identified as fraudulent, if and only if a fraudster sends the same cookie twice with the same counter value. This technique errs only by discovering false duplicates. Hence, its error analysis is identical to that of the Bloom based solution [22].

Advertisers' Attacks. Detecting *cookie-replay* attacks is crucial. Not only is it a major component in detecting attacks coming from a single cookie and multiple IPs, but also critical for detecting all advertisers' attacks. *Cookie-replay* is the only difference between publishers' and advertisers' fraud, and is the only way attacking advertisers make the commissioners show their competitors' advertisements.

As described earlier, for each surfer, the commissioner displays the most profitable advertisements that were not shown in the recent history. Sending the same cookie repeatedly results in displaying roughly the same advertisements. An attacking advertiser can make several visits to a publisher's site until the advertisements of its competitors are displayed. Then, the attacker saves the cookie, and repeatedly uses it to make the commissioner display the advertisements of his competitors, and simulates clicks on them. We report instances of this attack in Section 7.

4.4. Multi-Cookies and Single IP Attacks

This class of attacks could be performed in several ways. The most straightforward, though not the most economic, way is to employ several scripts running on several machines connected to the Internet through a single router. Hence, several requests will come from the same IP with

several cookie IDs. The problem with this attack is that it resembles normal Internet traffic, where several surfers with different cookie IDs connect to the Internet using one IP via a Network Address Translation (NAT) box or an ISP. A more sophisticated version of this attack is possible by connecting to the Internet through an ISP, so that the same IP used in the attack is also shared with legitimate traffic. By mixing the fraudulent traffic with normal traffic, the attacking publisher can dilute the effect of the attack, and confuse the commissioner's fraud detection mechanisms.

The commissioner needs to detect imbalances in the activities of IPs and publishers' sites, i.e., anomalies that do not reflect the average activity of the IP and the site. Even formalizing the problem is a challenge. One way is fitting a 2-dimensional distribution for the traffic, and picking pairs that exceed the expectation with some threshold. Another way of formalizing the problem is to normalize the traffic of the publishers' sites, normalize the traffic of the IPs, and search for pairs that deviate from the product of the normalized traffics. The large cardinalities of both IPs and publishers' sites (4G and 50,000, respectively) make any problem definition very challenging to solve.

4.5. Multi-Cookies and Multi-IPs Attacks

This is the class of attacks that is most difficult both to perform and to detect. The attacking publisher must have access to several valid cookies and IPs. There are several ways the fraudster can use the cookies and IPs. We mention a few here. The most straightforward, though not the most economic, way is having several machines with several accounts on ISPs at the disposal of the attacking publisher. Another way is to exploit the cookies and IPs of legitimate surfers through spywares and Trojans [31]. Such a *botnet* [1] would simulate impressions and clicks for the attacker's site by issuing the appropriate HTTP requests. The generated traffic resembles real traffic to a high extent.

We can view this class as a more complex case of some of the aforementioned classes. Assume the publisher has access to several valid cookies and IPs, every time an impression/click is made with one of the cookies, it is either pre-assigned to a specific IP, or an IP is picked randomly.

If every time a cookie is used in the attack, a pre-assigned IP is used, then this kind of attacks can be viewed as a more complex instance of the "Multi-Cookies and Single IP" case, but using several IPs. On the other hand, if the IP used with the cookie is randomly selected, then this will lead to the same cookie appearing with several IPs. This can be viewed as a more complex instance of the "Single Cookie and Multi-IPs" case, with several cookies.

5. Publishers' Coalition Attacks

Section 4 emphasizes the understandable correlation between the complexity of launching an attack, and the complexity of detecting its fingerprint. The attacks whose de-

tection mechanisms are complex, and left as open problems are those that involve personifying multiple identities using multiple IPs and cookie IDs. Generating traffic through ISPs that assign virtual IP addresses to surfers, such as AOL®, is not an effective way to increase the number of attacking IPs. The ranges of IPs of those ISPs are well known, and again, the ratio of the traffic of any publisher received from those ISPs is highly stable across all the publishers. Hence, such attacks are easily detected by examining the ratio of the traffic received from ISPs assigning virtual IPs as compared to the entire publisher's traffic. Even when botnets are used, attacks launched from a botnet of tens of thousands of machines infected by top-notch Trojans are more sporadic than attacks from tens of machines infected by out-of-the-box Trojans for two reasons. First, out-of-the-box Trojan code that is more popular is easily detected with anti-virus softwares. Hence, the majority of botnets cannot grow to huge sizes. Second, small botnets can still be powerful if coupled with high bandwidth [12].

Thus, fraudsters are motivated to either own the attacking machines, or to control botnets of real surfers' machines through top-notch Trojans in order to use the identities of real surfers. Launching scalable attacks entails high cost or requires advanced Trojan-writing skills. Therefore, fraudsters are motivated to form coalitions to reduce the cost and the difficulty of launching scalable attacks.

For instance, assume the commissioner pays publishers for up to u impressions per t time units for each pair of IP and cookie. Attacker A can generate u hits from each identity (resource) it controls, without being rejected by the commissioners' radars. Another attacker, B , can simulate another u undetectable hits per controlled resource. Hence, it is more scalable and cost effective for both A and B to time-share the resources, and generate $2u$ hits for each publisher, rather than doubling the amount of controlled resources. To stay under the radars' levels, both A and B use a wider range of resources while generating u visits from each resource for each publisher every t time units.

The phenomenon the commissioner has to look for, to detect *coalition* attacks, is the high similarity of sites' traffic. The set of IPs (or cookies) visiting a site is a representative characteristic of its traffic. Several set similarity metrics were reviewed in [11] that can be used here. From real data analysis, any two independent legitimate sites have negligible similarity metric [25]. Since an average-sized commissioner has around 50,000 publishers' sites, any naïve solution that tests all pairs of sites is unrealistic.

In [25], we devised the *Similarity-Seeker* algorithm based on the algorithm in [7]. *Similarity-Seeker* searches for pairs of sites with high traffic similarity that signals sites forming coalitions. However, fraudsters can still stay under the radar level by giving up some greed, and increasing the coalition size. A group of G attackers can make hard-to-

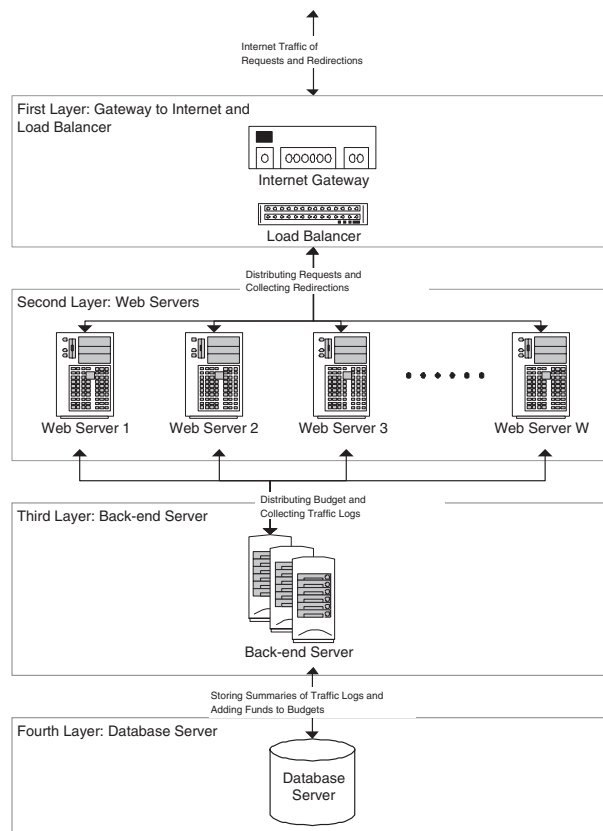


Figure 1. The Commissioners' Architecture.

detect coalitions by every site sharing a resource with only $g < G$ random sites. This reduces the similarity between sites' traffics, and still benefits the colluding sites.

Increasing the size of the coalitions from pairs to large groups shifts our focus from searching for pairs of sites with similar traffic to searching for groups with similar traffic. The relationships among the sites can be modeled as an undirected graph, where nodes represent sites, and an edge between two sites represents relatively high traffic similarity. Hence, instead of just searching for edges, the objective changes to searching for all maximal cliques in a huge sparse graph, which is solved optimally in [33].

6. The Proposed Architectural Model

Having discussed the classes of *non-coalition* and *coalition* fraud and detection techniques, we discuss how those techniques can be deployed on a real clustered Web server architecture borrowed from [9].

6.1. The Generic Web Server Architecture

The architecture in Figure 1 consists of four layers. As far as the fraud detection application is concerned, the first layer comprises the gateway to the Internet, through which the requests come from the Internet Browsers to the commissioner servers; as well as the load balancer, which assigns requests to different Web servers in the second layer of Web servers. We do not make any stickiness assumptions.

That is, two consecutive requests from the same site, and same cookie ID can be routed to two different Web servers according to the load of the servers at the time the requests are made. This assumption increases the portability of our model, though limits the optimizations possible.

The second layer comprises the Web servers. The primary functionality of the Web servers is serving the advertisements to the requests, redirecting clicks, logging traffic, and enforcing constraints like advertisers' geographical targeting, budgeting, etc. We assume the traffic logs collected by the Web servers are sent periodically to the third layer. The third layer is the Back-end that collects the data from the Web servers, does some data manipulation, and re-adjusts advertisements' budgets that are being consumed by the Web servers. This layer is assumed to have several machines with a single file system, or is otherwise running on a single powerful machine with some redundancy for fault-tolerance. The fourth layer is the database layer, which is responsible for accounting, trend analysis, etc.

6.2. Deploying the Techniques on the Architecture

In this section, we discuss how to map the fraud detection techniques of Sections 4 and 5 on this architecture.

Fraud Detection at the Second Layer. The Web servers do not have the complete picture of traffic across all sites or surfers. The fraud filters that can be used at this level are limited. Also, CPU cycles should not be used for heavy fraud detection schemes, since they need to process requests in real-time. Fraud detection in this layer can filter out cookies that appear from a number of IP addresses greater than the threshold, p , in the last specified time period, t .

If stickiness is enforced, further optimizations are foreseeable. If the request is directed from the load balancer to the Web server according to a hash of the cookie ID, then all requests from the same cookie ID would end up on the same server. This is called *cookie-stickiness*. Hence, this layer could detect duplicates, and *cookie-replay* attacks. However, it is generally not recommended to balance the load among the Web servers according to a hash of the IP addresses, the advertisements, or the sites, since the data gets highly skewed when aggregated by these dimensions, which will lead to unbalanced loads.

Fraud Detection at the Third Layer. The Back-end server has a complete picture of the traffic. However, since it interacts with the real-time Web servers, it is not recommended to load this layer with time-consuming queries. Hence, only online stream processing or sampling should be deployed at this layer. Fraud detection filters that can be employed here are those that detect duplicates, *cookie-replay* attacks, sites with high percentage of cookie-less traffic, and unbalanced activities of IP addresses and publishers (if a streaming solution is proposed in the future). This is also the right layer for detecting *coalition* attacks.

Fraud Detection at the Fourth Layer. The database layer has a high level picture of the whole traffic. The support of a DBMS, and powerful machines are assumed. In this layer, highly analytical classical queries should be employed, whose results should compliment those of the streaming algorithms running on higher layers. This layer is the best layer to deploy the classical defense mechanisms suggested in [16, 24]. Those defense mechanisms include heavy SQL statements that would identify the sites with suspicious *Click Through Rates (CTR)*.

7. Real Network Findings

In this section, we report our preliminary findings on a real network. We describe our experience with building a prototypical fraud detection system at Fastclick, Inc., a ValueClick company. We implemented some classical mechanisms including *CTR*. Since those metrics yield many false positives, we only took them into consideration for publishers suggested to be fraudulent by the online algorithms. For some classical techniques, we overcame the scale problem by uniformly sampling the traffic. After implementing, and tuning the defenses, we discovered three major kinds of outbreaks: automated repeated visits with the same identification, a cookie-replay attack, and the subtlest was a coalition attack. Some fraudsters employed simple scripts to generate automated visits to their sites. Some sites received more than 200 visits per hour from one pair of an IP and a cookie ID. The traffic that was generated using these simple attacks was less than 3% of the total traffic.

The cookie-replay attack was more interesting. Commissioners store the most recently displayed advertisements on the cookie to avoid showing repetitive advertisements and to guarantee good exposure for advertisers. As explained earlier, the displayed advertisement depends on the recently shown advertisements, and the returns of all advertisements. Some publishers kept sending the same cookies repeatedly, which violated the *frequency-capping* rules, and was detected through the unchanged cookie counter. In the beginning, we thought these could be false positives of the Bloom Filters, but after investigating the traffic database, it was clear that several publishers were receiving 10–18% of their traffic without altering the cookies, even though most of those publishers have exemplary classical metrics. The attack was not detected as a single IP single cookie attack, since it was generated from various IPs, but with the same cookie IDs. The attacker wanted the commissioner to display roughly the same advertisement every time, and it was clear that the traffic is generated by advertisers to deplete the budgets of their competitors. Although this traffic was directed to a few advertisers, it summed up to 1% of the traffic. Hence, it depleted the advertisers budget rapidly.

The discovered coalition attack was the subtlest. We analyzed the traffic for site pairs of similar traffic using the algorithm in [25]. To reduce the noise, we excluded all the

IPs that visited a large number of sites, because such IPs are probably coming from NAT boxes. The experiments revealed a coalition of 29 sites forming a perfect clique in the sites' similarity graph. This clique was sharing 15 sites with another clique of size 22. Several smaller cliques were also discovered.

The only abnormality with the suspected sites was that their traffic was of moderate size, yet coming from IPs all over the world. After further investigations, we found that the `Referer` fields in the HTTP requests were coming from pages that do not have the commissioner's advertisements; and sometimes no advertisements at all. We suspect the attackers had some form of readily available traffic through a botnet, and that they do not work for the domains they signed up for. When the commissioner sent the account activation e-mails², the publishers, somehow, acquired the attached activation secrets, and activated the accounts. Since the activation secrets are stored in a hashed form, the attackers must have compromised some machines on those domains, and acquired the attached secrets.

The preliminary implementation of our techniques serves as a successful proof of concept. Although there are more detection techniques to be devised and integrated into our current system, the system discovered several fraud attacks that sum up to 5.6% of the traffic. Interestingly, the discarded traffic is definitely fraudulent, which is fair for both advertisers and the commissioners.

8. Conclusion

In this paper, we provided a classification of the publishers' *non-coalition* and *coalition* fraud techniques. In addition, we proposed several techniques for detecting automated traffic. We developed stream analysis techniques for most of the problems, and abstracted some theoretical stream analysis problems for the research community. Finally, we communicated our experience in building a fraud detection system on a real network.

References

- [1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of the 6th ACM SIGCOMM IAC Internet Measurement Conference*, pages 41–52, 2006.
- [2] V. Anupam, A. Mayer, K. Nissim, B. Pinkas, and M. Reiter. On the Security of Pay-Per-Click and Other Web Advertising Schemes. In *Proceedings of the 8th WWW International Conference on World Wide Web*, pages 1091–1100, 1999.
- [3] B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970.
- [4] C. Blundo and S. Cimato. SAWM: A Tool for Secure and Authenticated Web Metering. In *Proceedings of the 14th ACM SEKE International Conference on Software Engineering and Knowledge Engineering*, pages 641–648, 2002.
- [5] A. Broder. Data Mining, the Internet, and Privacy. In *Proceedings of the 1st WEBKDD International Workshop on Web Usage Analysis and User Profiling*, pages 56–73, 1999.
- [6] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-Wise Independent Permutations (Extended Abstract). In *Proceedings of the 30th ACM STOC*

- Symposium on Theory Of Computing*, pages 327–336, 1998. An extended version appeared in the *Journal of Computer and System Sciences*, 60(3):630–659, 2000.
- [7] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proceedings of the 6th WWW International Conference on World Wide Web*, pages 391–404, 1997.
- [8] E. Burns. Online Seizes More of the Advertising Mix. *ClickZ News*, February 13 2006.
- [9] V. Cardellini, E. Casalicchio, M. Colajanni, and P. Yu. The State of the Art in Locally Distributed Web-Server Systems. *ACM Computing Surveys*, 34(2):263–311, 2002.
- [10] CERT Coordination Center. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks. <http://www.cert.org/advisories/CA-1996-21.html>, September 19 1996.
- [11] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th ACM STOC Symposium on Theory Of Computing*, pages 380–388, 2002.
- [12] E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *Proceedings of the 1st USENIX SRUTI Workshop on Steps to Reducing Unwanted Traffic on the Internet*, 2005.
- [13] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: a Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.
- [14] M. Jakobsson, P. MacKenzie, and J. Stern. Secure and Lightweight Advertising on the Web. In *Proceedings of the 8th WWW International Conference on World Wide Web*, pages 1101–1109, 1999.
- [15] V. Khu-smith and C. Mitchell. Enhancing the Security of Cookies. In *Proceedings of the 4th ICISC International Conference on Information Security and Cryptology*, pages 132–145, 2001.
- [16] D. Klein. Defending Against the Wily Surfer-Web-based Attacks and Defenses. In *Proceedings of the 1st USENIX ID Workshop on Intrusion Detection and Network Monitoring*, pages 81–92, 1999.
- [17] P. Lerma. When Should Online Come Before TV? *ClickZ News*, September 20 2005.
- [18] M. Liedtke. Google to Pay \$90M in 'Click Fraud' Case. *Washington Post Magazine*, March 9 2006.
- [19] M. Liedtke. Yahoo Settles 'Click Fraud' Lawsuit. *MSNBC News*, June 28 2006.
- [20] C. Mann. How Click Fraud Could Swallow the Internet. *Wired Magazine*, January 2006.
- [21] R. McGann. Study: Consumers Delete Cookies at Surprising Rate. *ClickZ News*, March 14 2005.
- [22] A. Metwally, D. Agrawal, and A. El Abbadi. Duplicate Detection in Click Streams. In *Proceedings of the 14th WWW International World Wide Web Conference*, pages 12–21, 2005.
- [23] A. Metwally, D. Agrawal, and A. El Abbadi. Using Association Rules for Fraud Detection in Web Advertising Networks. In *Proceedings of the 31st VLDB International Conference on Very Large Data Bases*, pages 169–180, 2005.
- [24] A. Metwally, D. Agrawal, and A. El Abbadi. On Hit Inflation Techniques and Detection in Streams of Web Advertising Networks. Technical Report 2006-06, University of California, Santa Barbara, 2006.
- [25] A. Metwally, D. Agrawal, and A. El Abbadi. DETECTIVES: DETECTing Coalition hiT Inflation attacks in adVERTISING nETworks Streams. In *Proceedings of the 16th WWW International World Wide Web Conference*, 2007.
- [26] D. Morgan. Making the Cookie Case to Consumers. *ClickZ News*, March 24 2005.
- [27] M. Naor and B. Pinkas. Secure and Efficient Metering. In *Proceedings EU-ROCRYPT International Conference on the Theory and Application of Cryptographic Techniques*, pages 576–590, 1998.
- [28] T. Novak and D. Hoffman. New Metrics for New Media: Toward the Development of Web Measurement Standards. *World Wide Web Journal*, 2(1):213–246, 1997.
- [29] S. Olsen. Click Fraud Roils Search Advertisers. *CNET News*, March 4 2005.
- [30] M. Reiter, V. Anupam, and A. Mayer. Detecting Hit-Shaving in Click-Through Payment Schemes. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pages 155–166, 1998.
- [31] G. Shaw. Spyware & Adware: the Risks Facing Businesses. *Network Security*, 2003(9):12–14, 2003.
- [32] J. Thaw. Online Ad Growth Accelerates, Outpacing Newspaper, TV Spending. *Bloomberg News*, December 28 2005.
- [33] E. Tomita, A. Tanaka, and H. Takahashi. The Worst-Case Time Complexity for Generating All Maximal Cliques. In *Proceedings of the 10th COCOON Annual International Conference on Computing and Combinatorics*, pages 161–170, 2004.
- [34] D. Vise. Clicking To Steal. *Washington Post Magazine*, page F01, April 17 2005.
- [35] T. Zeller Jr. With Each Technology Advance, a Scourge. *The New York Times*, October 18 2004.

²When a publisher signs up with a commissioner, the commissioner sends the publisher an e-mail on the domain signed up, with a secret key. The publisher can activate the account only using this key. This ensures that the publisher has an e-mail account on the domain (s)he signed up for.