Understanding Fraudulent Activities in Online Ad Exchanges

Brett Stone-Gross, Ryan Stevens, Richard Kemmerer, Christopher Kruegel, Giovanni Vigna
Department of Computer Science
University of California, Santa Barbara
bstone,rstevens,kemm,chris,vigna@cs.ucsb.edu

Apostolis Zarras
Institute of Computer Science
Foundation for Research and Technology, Hellas
zarras@ics.forth.gr

ABSTRACT

Online advertisements (ads) provide a powerful mechanism for advertisers to effectively target Web users. Ads can be customized based on a user's browsing behavior, geographic location, and personal interests. There is currently a multi-billion dollar market for online advertising, which generates the primary revenue for some of the most popular websites on the Internet. In order to meet the immense market demand, and to manage the complex relationships between advertisers and publishers (i.e., the websites hosting the ads), marketplaces known as "ad exchanges" are employed. These exchanges allow publishers (sellers of ad space) and advertisers (buyers of this ad space) to dynamically broker traffic through ad networks to efficiently maximize profits for all parties. Unfortunately, the complexities of these systems invite a considerable amount of abuse from cybercriminals, who profit at the expense of the advertisers.

In this paper, we present a detailed view of how one of the largest ad exchanges operates and the associated security issues from the vantage point of a member ad network. More specifically, we analyzed a dataset containing transactions for ingress and egress ad traffic from this ad network. In addition, we examined information collected from a command-and-control server used to operate a botnet that is leveraged to perpetrate ad fraud against the same ad exchange.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce, Security; C.2.0 [**Computer-Communication Networks**]: General

General Terms

Experimentation, Measurement, Security

Keywords

Ad fraud, Ad networks, Online advertising

IMC'11, November 2–4, 2011, Berlin, Germany.

1. INTRODUCTION

Online advertising has developed into a massive economy and is now the main source of revenue for some of the most popular online businesses and search engines (e.g., Google and Yahoo! [5]). In its simplest form, online advertising is a buyer/seller relationship between those who want to show ads (advertisers, who buy space on Web pages) and those who get paid to display ads for others for a fee (publishers, or sellers, who own the Web pages). The process becomes more complicated as more advertisers and publishers are added to the system. To facilitate these endeavors, an intermediary entity called an ad network (or ad commissioner [20]) keeps track of the publishers and advertisers within its domain. It is the ad network's job to take the publishers' ad requests, which are generated when users load a publisher's website, and pair them with the advertisers willing to pay the most for the publishers' niche market. In turn, the network takes a percentage of all revenue that is exchanged in transactions that it oversees. Thus, in the network model of advertising, handling ad requests becomes a broker process that seeks to maximize revenue for all involved parties.

Ad exchanges, such as Google's DoubleClick or Yahoo!'s RightMedia, operate similar to an ad network, where the entities that buy or sell ads are ad networks. This allows one ad network to sell its publishers' ad space to another network or buy ad space for its advertisers, so that advertisers are able to reach a much larger audience than they could if they were part of a single ad network. Unlike an ad network, an ad exchange is not fully connected, and networks in the exchange cannot buy and sell each other's ads freely until they have established a contractual agreement describing how traffic will be bought and sold between them. This allows the ad exchange to handle the distribution of ads from a centralized ad server, which has a public API allowing ad networks to configure and monitor their accounts.

While ad exchanges provide a powerful mechanism for advertisers, ad networks, and publishers to efficiently manage their ad traffic, they have also become a lucrative target for cybercriminals. In particular, miscreants have developed malware that is used to remotely control compromised computers, known as *bots*, and network them into a single *botnet*. A botnet can be instructed to view and click ads, simulating user traffic. As a result, botnet operators can generate revenue simply by creating a website, signing up as a publisher, and directing their bots to view and click on the advertisements contained on their own website.

In this paper, we collaborated with a large ad network that is a member of Yahoo!'s RightMedia, one of the largest ad exchanges. This enabled us to obtain a direct view of how a large real-world ad network operates in the context of an ad exchange, and to analyze the security threats and fraudulent activities that pose the greatest

risks. Due to the sensitive nature of the subject, we will refer to this ad network throughout the paper as NETWORKX. Our goal in this project was to study and understand how an ad exchange operates from the perspective of an individual ad network. Moreover, we applied a number of models to the ad network's data to explore methods that might be able to identify suspicious click traffic in the context of an ad exchange. In addition, we obtained access to a command-and-control (C&C) server that was used to control a botnet that engaged in ad fraud targeted towards the RightMedia exchange. Because of our access to the botnet C&C server, our models were geared more towards fraud that was perpetrated by a botnet; however, we also discuss a number of additional fraud techniques we observed. To the best of our knowledge, this is the first large-scale study of fraudulent activities in online ad exchanges from these vantage points.

2. BACKGROUND

In this section, we introduce online advertising and describe how ad exchanges operate to serve ads on the Internet. We then review types of fraud that are known to exist in online advertising, and we describe which techniques are used to prevent this fraud from occurring. Finally, we describe the ad fraud botnet C&C server that was used to perpetrate fraud on RightMedia.

2.1 Terminology

Below are definitions of the various advertising terms we use in this paper. Note that some of the terms below are defined in the context of Yahoo!'s RightMedia. For a more general definition of the terms, refer to the Internet Advertising Bureau's online glossary [3].

• *Publishers* (or *Sellers*) make money through the exchange by hosting websites with advertisements. Each visitor to their sites generates revenue for the publisher depending on the niche market that describes their websites. In general, the more visitors publishers attract to their websites the more money they earn.

• *Advertisers* (or *Buyers*) pay the ad network to have their ads displayed on publishers' websites. Whenever their ads are shown, they have to pay the ad network, and a percentage is paid to the publisher.

• *Ad Networks* are entities in the exchange that manage publishers and advertisers. They are able to buy and sell ad traffic (in the form of ad requests) internally as well as through other ad networks. Ad networks that can buy and sell traffic between each other are called *linked partners*, and each ad network maintains its own list of trusted partner networks.

• Instead of having static ads that display the same content, publishers load ads dynamically by putting *sections* (also called *zones* or *regions*) on their pages. A section simply refers to a block of space on the page that is able to make a request for an ad dynamically when the page is loaded. In practice, this is often implemented by embedding an iframe that loads some JavaScript in the page, which, in turn, detects if the browser has Adobe Flash and whether browser cookies are enabled. This information is sent to an ad server so that an ad can be served to the user's browser, in a process called an *ad request*.

• A *creative* refers to the content of the actual advertisement, which is what the visitor sees on the page after the ad is served. The ad normally consists of an image or an Adobe Flash animation and an anchor tag that points to the advertiser's website, called a *click-through*.

• The *auction process* refers to how a section is populated by a creative. It involves matching up each ad request with the most profitable advertiser bid for the request. Before any ads are served, publishers and advertisers outline a number of ad serving requirements such as budget, when ads should be shown, and targeting information. These requirements are used to match requests and bids autonomously in the exchange in a way that maximizes profit for the publisher. A single successful auction in the exchange is called an *impression*.

• A *click* event is generated when a user clicks on an ad, and it usually brings more revenue to the publisher than an impression alone. Clicks and impressions are handled separately in the exchange, so a user loading a page and clicking on an ad actually generates two events, an impression and a click. The ratio of clicks and impressions is the ad's *Click Through Rate* (CTR). Publishers' CTRs are also recorded for use in fraud detection.

• *Ad campaigns* are the way in which advertisers specify how much they pay when their ads are shown. There are many different types of campaigns, but the most common type is based on *Cost per Mille* (CPM) impressions, which is simply how much one thousand impressions are worth to an advertiser. In this scheme, an advertiser pays an amount to the publisher for each ad that gets served to the site.

• Additional ad campaign types are *Cost per Click* (CPC) and *Cost per Action* (CPA). CPC deals pay the publisher only when a user clicks the ad that is served; CPA deals only pay the publisher when a user clicks the ad and continues on to perform some action on the site (known as a *conversion*), usually filling in a landing page form. Because the amount of revenue associated with CPC and CPA deals depends on a user clicking the ad, the server estimates how much the ad will pay by calculating the *effective Cost Per Mille* (eCPM) impressions. The formula for this is: eCPM = ((Payout per impression) + (Historical CTR) * (Payout per click) + (Historical actions to impressions) * (Payout per action)) * 1000.

• In addition to the auction process, there is a practice called *arbitrage* that ad networks can use to increase their revenue. Arbitrage is done by ad networks buying impressions from publishers as if they were a real advertiser, and starting a new auction for the ad slot as if they were a real publisher. As we will discuss later, this has a number of implications that have to be accounted for when analyzing the data stream from NETWORKX.

2.2 Structure of an Ad Exchange

An ad exchange is structured as a graph where each node is an ad network, which owns its own set of unique publishers and advertisers. Publishers are only able to request ads (sell traffic), and advertisers can only bid on requests (buy traffic). However, ad networks are able to both buy and sell traffic, allowing them to act as brokers between different parts of the exchange that would not be connected otherwise. Thus, publishers and advertisers are edge nodes, and ad networks form the backbone of the exchange, as shown in Figure 1.

As previously mentioned, the exchange is not fully connected and traffic cannot be bought and sold freely across the exchange. This is because an ad network can only interact with another ad network if they are linked partners. Partnerships define an edge in the graph; when two entities become affiliates they decide on the specific parameters that will define their relationship in the exchange, which is usually a revenue share deal. For example, NETWORKX may take 50% of all publisher revenue for a particular partner who targets traffic from users interested in German news, and 60% of

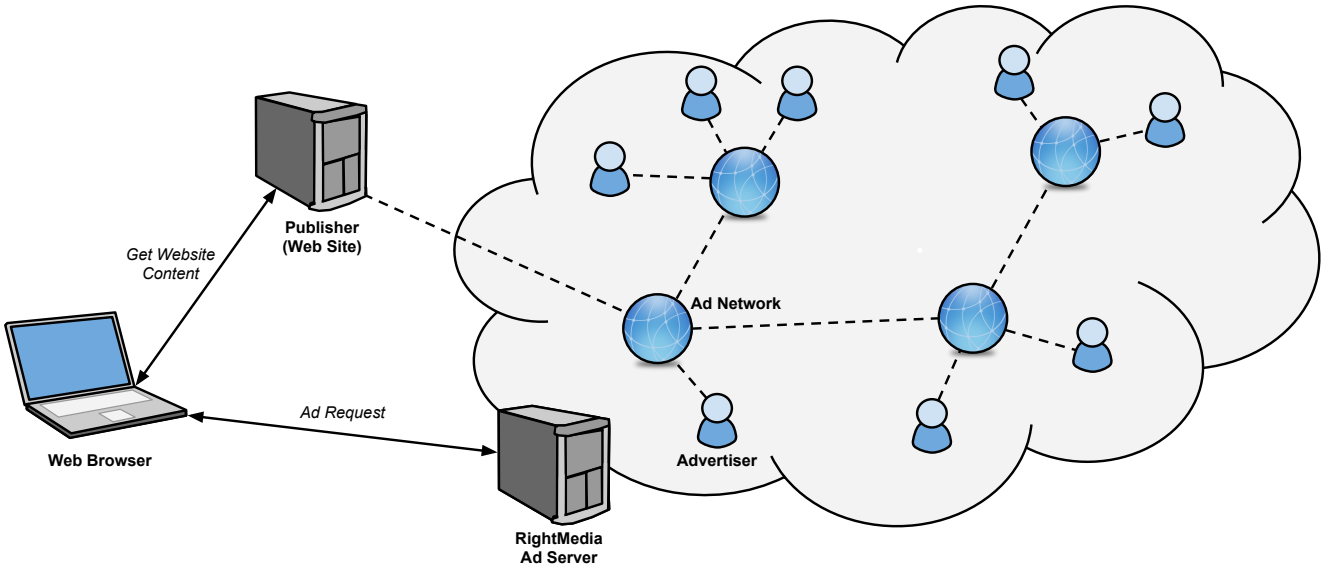

Figure 1: An overview of an online ad exchange.

publisher revenue with another partner that targets American sports traffic.

2.3 How the Auction Process Works

In the ad network model of online ad serving, one advertising network has access to its own advertisers and publishers, and the network searches the list of advertisers for each ad request to determine which would be the most profitable for the publisher and thus the network. The process is similar in an ad exchange, except that the pool of advertisers also includes advertisers who are part of the network's linked partners (and those partners' linked partners, and so on). The process starts when a user loads a Web page that contains a RightMedia HTML iframe or JavaScript popup that initiates a request for an advertisement of a specific size. A unique identifier, known as a *section ID*, enables an ad network to track impressions, clicks, and conversions from a particular publisher. Note that the section ID is the only information that an ad network uses to determine the publisher that should be credited. As we will discuss later, this makes verifying the legitimacy of a single ad request very difficult.

When the ad request is received by the ad exchange's ad server, the ad server is able to lookup which publisher the section ID belongs to. This ad request is then associated with the publisher's ad serving requirements, and it is matched with viable bids, based on the targeting criteria of the advertisers. Each advertiser specifies a maximum bid, but the actual bid amount is determined automatically by the exchange based on the advertiser's Return on Investment (ROI) goals and the number of other advertisers who are able to bid. The bids are always in the form of the effective cost per mille impressions (eCPM), regardless of the types of affiliations between intermediary ad networks. The final amount the bid is worth to the publisher is affected by how many intermediary ad networks are brokering the transaction. For example, consider a publisher and an advertiser who are part of an ad network. The advertiser's campaign with the network is a $1 CPM deal, while the publisher has a 50% revenue share deal. Therefore, the advertiser's bids will only be worth $0.50 to the publisher, because the network is taking half of the revenue. In reality, the ad exchange would take a cut as well. When the value of all available bids has been computed, the exchange simply picks the bid with the highest eCPM for the publisher, and then the ad is served.

2.4 How Arbitrage Works

Arbitrage occurs after the auction process. In order to initiate arbitrage, the network must buy the publisher's ad traffic itself and then resell the traffic in a completely new, independent auction. This is done by serving a new ad tag to the user after the initial auction, instead of an ad. The new ad tag contains a new section ID that is owned by the ad network and not by an actual publisher (note that the ad network could choose to return a non-RightMedia ad tag as well). Whatever revenue is generated in the new auction goes to the ad network that won the first auction, in the hopes that the second auction will make a profit larger than the ad request cost. Since the network must buy the traffic before reselling it, unsold arbitrage traffic is a direct loss for the ad network. Arbitrage can be repeated a number of times across different ad networks until an actual ad is served; this process is called *daisy chaining*. The longer the chain of arbitrage, the longer it takes to load the ad that is finally displayed in the user's browser that first made the ad request.

2.5 Known Types of Fraud

Fraud (or *Ad Fraud*) in the context of an ad exchange is a means by which a member or members in the exchange try to increase their profits at the cost of other members in the exchange. A *fraudster* is simply a member of the exchange who is perpetrating fraud of some kind. The fraudster may be a publisher in the exchange who is attempting to make more money than it deserves, or an advertiser who is targeting other advertisers to reduce ad competition.

The simplest kind of fraud is called *impression spam*, and it involves fabricating HTTP requests to either the publisher's page, or the ad server directly, to artificially inflate the actual amount of traffic. This type of fraud targets CPM deals, but may be mixed in with other types of fraud to remain stealthy [6]. One such kind of fraud that is usually more profitable than impression spam alone is *click spam*, which is done by generating HTTP requests to advertisement click URLs, usually after an ad is served. There are two kinds of click spam fraud. *Click inflation* is the practice of publishers making more money than they deserve through inflating CPC deals, or

increasing their CTR and thus their eCPM. *Competitor clicking* is the practice of advertisers making false clicks against competitor's ads to deplete their advertising budget. The last type of traffic inflation is called *conversion (action) spam*, and is like click spam but requires certain GET or POST parameters, requests a file for download, or follows a specific order of pages to generate a conversion against that advertiser. Like click spam, this can be perpetrated by either publishers or advertisers. This type of fraud only works if the action does not require spending money directly, such as purchasing an item from the website.

Lastly, *misrepresentation* is the practice of a publisher breaking some rule of the network or exchange by lying about their website contents or about what pages ads are actually being shown on (e.g., by spoofing the referring URL). The publisher normally does this to get higher value ads on their pages than they would be able to get if they did not lie about their website contents, or because their website contents are illegal and would not be allowed in the network otherwise.

2.6 Known Types of Attacks

Below are the known types of attacks that either have been performed or could be performed in the context of an ad exchange:

• *Hired Clickers*: This type of attack involves someone sitting in front of a computer and constantly reloading a page and clicking on ads.

• *Keyword Stuffing*: A type of misrepresentation fraud that increases the value of ads that are shown on the fraudster's pages. This is done by including a certain amount of "invisible" content that contains many high-value advertising keywords. The invisible content is either in hidden HTML tags, text that is the same color as the background, or very small text. When the network crawls the fraudster's page to classify the content, the page will be classified as being more valuable or targeted than it really is. This drives higher value ads to the fraudster's page,

• *Impression Stuffing*: The practice of fraudster's putting excessive numbers of banners on their pages so that they get a large number of impressions for each page view. This also includes "stacking" ads on top of each other so that background ads cannot be seen [8].

• *Coercion*: This attack is perpetrated by fraudsters who convince users to click on their ads for reasons other than the ad content itself. This includes an administrator simply asking users to click on their ads, but also includes obfuscating ads with actual site content to trick a user into clicking on ads (e.g., making all valid links on the page look like ads as well).

• *Custom Clickbots*: These are custom software components developed by fraudsters that perpetrates a particular kind of fraud against certain publishers or advertisers. These clickbots normally sit on one or more static machines and issue HTTP requests to certain URLs to simulate impressions, clicks, or conversions [6].

• *For-sale Clickbots*: These are bots that are available for download or purchase and perform click fraud. The bots can be configured to perform many types of fraud and can be given lists of publishers' pages to visit, ads to click on, and proxies to use to diversify the bot's IP addresses [6].

• *Botnet Clickbots*: Botnets are the most difficult type of fraud to detect from an ad exchange's perspective, and it is the most common source of fraudulent Internet ad traffic [21]. Botnets are unique in that the software required to perpetrate fraud is located on many benign users' machines. The malicious software usually comes in one of two flavors: those that run behind the scenes and act as normal clickbots, and those that attempt to coerce the user of the machine to perform some of the ad fraud actions.

• *Forced Browser Clicks*: An attack that forces the user's browser to follow the click URL of an ad by including some client-side script, normally JavaScript. This type of attack can be avoided by putting all ads in an iframe with the source attribute set to an ad tag located on the ad network's website. By using this technique, the content of the iframe is not accessible to any script that did not come from the same domain as the iframe's source. However, recent research suggests that there are still ways of getting click URLs out of iframes [9].

2.7 Detection and Prevention Methods

Below are the known defenses against ad fraud in the context of an ad exchange.

• *Signature-based Detection*: This type of detection uses static rules to decide which ad traffic should be considered valid and which discarded as invalid. One example of a common rule is that the second of any duplicate clicks (caused from a user double-clicking an ad) is considered invalid. This type of detection is beneficial in finding known attacks by looking for known malicious patterns, but it does not work on attacks whose patterns are not known or do not follow static rules [17].

• *Anomaly-based Detection*: This approach uses historical information about publishers to find sudden changes in ad traffic patterns. This may involve looking for a sudden increase in the number of impressions from a publisher, the CTR of the publisher, or the classification/quality of traffic the publisher is generating (e.g., a search engine that suddenly only queries for high-value ad keywords). This type of detection is useful for identifying publishers who are misbehaving or for when fraudsters change or update the type of attack that they are perpetrating.

• *Reverse Spidering (Auditing)*: The practice of ad networks, ad exchanges, or advertisers crawling the HTTP `Referer` of incoming impressions to ensure that the referring sites have the content they claim to have. The reverse spiders look at keywords in HTML content, JavaScript, and iframes of the pages to look for any potentially illegitimate content. To avoid this kind of detection, fraudsters assign a unique ID to the referring URL each time a fraudulent click or impression is generated. Then, when the audit program crawls the referrer, the website will recognize that the ID has already been used, and will not serve any malicious content to the spider [9].

• *Bluff Ads*: These are ads that an ad network serves to a publisher to detect fraudulent activity. These ads are served in response to a random percentage of all requests that come from the publisher and are unique in that they are purposefully uninviting (meaning they contain little more than a picture with no text that does not try to attract the user's attention or get them to click on it). If the click-through rate and conversion rate of these bluff ads is not much lower than ads normally displayed, this would indicate fraud from the publisher [11].

• *Web Site Popularity and Page Rankings*: The number of impressions a publisher is generating for their Web page can be checked against known, trusted website rankings such as Alexa or Compete. If the publisher has much more traffic than their page ranking would suggest, this would be indicative of fraudulent activity [2].

• *Performance-based Pricing*: Performance based pricing is simply a name for publisher payment schemes that do not pay on impressions but instead on how much return on investment (ROI) an advertiser gets from a publisher. The simplest performance-based pricing model is CPC, but CPA deals fall into this category too. This type of pricing reduces impression fraud by requiring publishers to provide a certain level of measurable benefit to the advertisers to make money. This also reduces cost to advertisers as networks with more fraud will have a lower ROI for their advertisers, meaning that advertisers will have to pay less to get their ads shown. In this way, performance-based pricing mitigates the effect of fraud on the advertisers without actively avoiding it [12].

2.8 Botnet-Related Ad Fraud: A Case Study

In this section, we describe the process that allowed us to obtain access to a command-and-control (C&C) server that controlled a botnet used to commit ad fraud, and the data that we collected. The bot malware first came to our attention in February 2010, when we were investigating botnets in the wild. By analyzing the network connections generated by the malware sample, we were able to identify the location of the C&C server that was in control of the botnet. We then contacted the hosting provider whose server was being used by the fraudsters and provided them with detailed evidence of the abuse. The hosting provider suspended the criminal account in March 2010 and provided us with access to the information stored on the server.

By studying the behavior of a bot sample and the source code of the botnet C&C, we were able to get a complete view of how the entire operation functioned. There were two primary methods the botmasters used to earn money: impression/click fraud and affiliate programs that paid a commission based on conversions (e.g., registration, sales, etc). The mode of operation for the impression/click fraud was managed by a configuration file received from the C&C server. The configuration contained various parameters for controlling the patterns of impressions and clicks, iframes to load within a Web browser, and a list of domains that were used to spoof the source of the impression/click through the HTTP `Referer` field. The iframes are directly loaded by the malware in the background of a running Internet Explorer instance through a browser-helper object extension, and they are invisible to the user of the infected system. When the malware loads an iframe, it generates an ad request to RightMedia using a section ID of the fraudsters and a spoofed `Referer` that was set by the configuration file. By loading the iframe directly from each bot, the malware does not need to visit an actual website to emulate impression and clicks.

Interestingly, we found that the domains that were used to spoof the `Referer` field of the fraudulent ad requests contained legitimate RightMedia ads. We will discuss the practice of spoofing the referrer later in Section 5.2, but we believe that the fraudsters use such legitimate sites to remain stealthier. There were also a small number of domains (and websites) in the configuration file that were set up by the fraudsters and used to register multiple publisher accounts. At first these sites seemed legitimate, but on closer examination we identified that the content on these pages was actually stolen from other sites. We will describe these fake websites in more detail in the next section.

Periodically, an infected computer connects back to the C&C server to report its status and to receive a new list of instructions. We also found that the C&C server had the ability to push arbitrary binary executables to the bots; this was regularly used to upgrade the click-fraud malware to newer versions, but could have been used to push more intrusive spyware or adware onto the victims' machines. The bot malware also used browser hijacking to redirect users from a target site (e.g., freecreditreport.com) to an affiliate site (e.g., incentaclick.com and adsmarket.com). Depending on the referring site, the affiliate site would redirect a user's browser to a similar website (e.g., gofreecredit.com). Based on the records maintained by the botnet controllers, their malware redirected users 3,425,030 times from mid-February to May 2010. However, the number of conversions was not stored on the C&C server, so we cannot determine how many users fell for the scam.

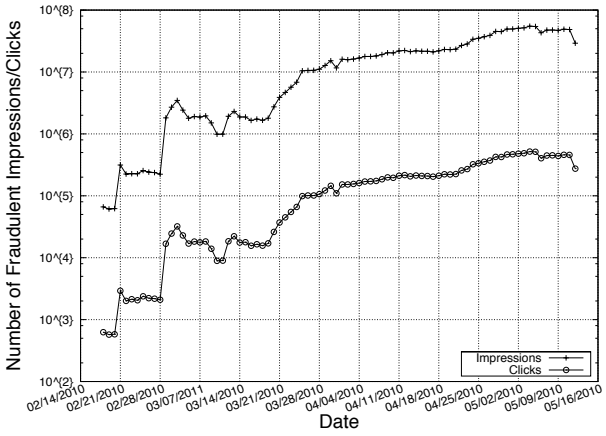

Figure 2: Number of fraudulent impressions and clicks from a click-fraud botnet.

The database for this particular botnet contained records for 530, 985 bot installations, with 1,479,036,685 impressions and 14,060, 716 clicks (an average click through rate of 0.95%) from mid-February to May 2010. The daily number of impressions and clicks is shown in Figure 2. Interestingly, the ratio of impressions to clicks followed each other almost exactly over time. In Section 4, we will use this behavior as a method for detecting which IP addresses are potentially clickbots. As we will discuss in Section 3.1, the average CPM for impressions and cost-per-click were $0.084 and $0.017, respectively. In other words, the cybercriminals behind the botnet may have netted approximately $124,000 for impressions and another $255,000 for clicks during this 2.5 month period.

We would also like to point out that this botnet malware operates based on an affiliate program out of Eastern Europe (commonly referred to as a *partnerka*) similar to other online criminal operations such as fake antivirus ventures [23]. More specifically, attackers are paid to compromise as many computers as possible and infect them with malware. It should be noted that this botnet has been operational for more than a year, and is still active, at the time of writing.

2.9 Fake Web Sites

The information from the botnet C&C server revealed a number of cases in which the fraudsters used fake websites with seemingly legitimate domain names to register as publishers with different ad networks. At first glance, these websites appeared to contain useful content for Web visitors. However, upon further (manual) inspection, we found that there were common patterns shared by many of these sites. The format of these sites was identical, namely a Wordpress blog template with posts only by an "admin" user, no comments, and a large number of ads (from several different ad exchanges) embedded throughout each of the pages. All of them were based on the same HTML template and, most importantly, contained content that was stolen from other websites. In addition, the sites appeared to be hastily set up with parts of the templates

displaying default text (e.g., "text goes here"). Furthermore, we analyzed the WHOIS information for these website domain names and found very similar registration information (name, phone number, address) across many of these domains.

The fraudsters behind the botnet C&C server created a number of these fake websites in order to register as a publisher with several ad networks. After receiving section IDs, the malware spoofed the HTTP `Referer` field and directly loaded the HTML iframe that contained the ads that would normally be found on one of these fake websites. In other words, the malware did not need to visit the fake websites to load advertisements, but rather could bypass the fake websites to reduce the amount of bandwidth and hosting costs.

3. DATA COLLECTION

In this section, we describe the dataset that we utilized to study the RightMedia ad exchange. We also discuss how we manually established a ground truth dataset of good and bad publishers, which we then used to evaluate different models that could be used for fraud detection.

3.1 Data Feed

We obtain a feed of real-world ad data from NETWORKX, which is an ad network that is part of the RightMedia exchange. Every 30 minutes, we receive a batch of the traffic that this ad network has seen in the previous time period.

The traffic can be split into three distinct flows based on the different types of transactions that are allowed in RightMedia. These flows are local publisher traffic, arbitrage traffic, and auction traffic; depending on which flow we were analyzing, different amounts of data were available to us. Local publisher traffic is any traffic that originated from an ad request from one of NETWORKX's own publishers. Of all the flows, local traffic provided us with the most detailed information per impression, both because all the fields in the ad request were populated with meaningful information and because we could look up additional information about the publishers' accounts from NETWORKX's database. Auction traffic is any traffic that NETWORKX made money on through linked partnership as a middleman but that NETWORKX did not buy or sell directly. A number of fields for auction traffic records are suppressed to protect linked partners from being able to learn too much about each other's traffic. Arbitrage traffic is traffic that was either purchased by or sold by NETWORKX itself (see Section 2.4); this was the most difficult type of traffic to analyze. Arbitrage traffic comes in pairs, with one impression representing the purchase of the ad traffic and a corresponding impression for reselling the ad traffic. Unfortunately, there was no way to reliably pair bought impressions with sold ones.

To study the data feed, we implemented an automated system to periodically retrieve NETWORKX's data records and extract relevant information. The data itself was in the form of RightMedia's *Custom Data Feed* format, which contains a log of each impression, click, and conversion that was received by NETWORKX. Each file represented 30 minutes of traffic, was about 200 Megabytes unpacked, and contained, on average, details for approximately 750,000 impressions. The feed had 51 individual fields that were conditionally populated. We found the following fields to be most useful for identifying potential fraud:

- *IP Address*: Right Media only provides an ad network with the first three bytes of the IPv4 address, to preserve users' privacy. However, we were able to estimate how many different users were in each 24 prefix based on how many unique cookie IDs we identified per IP address.

- *Cookie ID*: Unique token given to each browser instance that views an ad; it is stored as a cookie on the local machine and sent with every ad request. Note that this ID is a hash of the actual cookie value.

- *Creative ID*: Unique ID given to each creative to track which specific ad was shown.

- *Section ID*: Unique ID for the particular ad space where the ad was shown. For sold arbitrage traffic, this corresponds to a section ID that NETWORKX owns, and for local traffic, it corresponds to a section ID that a local publisher owns. This field is not populated for auction traffic or purchased arbitrage traffic.

- *Referrer*: Website URL of the referring website for local publisher traffic, or a subdomain of NETWORKX for sold arbitrage traffic. Not populated for auction or purchased arbitrage traffic.

- *Impression/Click/Conversion*: Whether it was an impression, a click, or conversion.

- *Advertiser Cost/Publisher Revenue*: How much the publisher got paid for the ad and how much the advertiser paid out. The difference between the advertiser cost and publisher revenue is how much the intermediary ad networks earned for the ad.

- *Buyer ID*: Unique ID given to each advertiser, or a network-owned buyer ID for bought arbitrage traffic. For auction traffic, this field is populated with the ID of a trusted partner network.

- *Seller ID*: Unique ID given to each publisher, or a network-owned seller ID for sold arbitrage traffic. For auction traffic, this field is populated with the ID of a trusted partner network and not a publisher.

- *User Agent ID*: Identifies the user-agent field that was specified in the HTTP headers. RightMedia currently enumerates 40 different types of browsers and versions.

- *Region ID*: Identifies local geographical areas (state, province, or city) via IP-address-based geo-location services.

Because we were working closely with NETWORKX, we had access to advertiser and publisher accounts that the network owned and managed, which allowed us to glean additional information from the data feed for local traffic. In particular, we could see the domain(s) that publishers used to sign up, how long they have been active, and their traffic statistics. This allowed us to compare our results with RightMedia's software (which was running on NETWORKX's server). With auction and arbitrage traffic, we were not able to relate IDs to real publisher accounts in RightMedia, which limited our analysis on this traffic. Therefore, most of our analysis focused on local traffic.

We investigated in depth ten days worth of data collected from NETWORKX. More precisely, we analyzed the data collected between April 20 and April 30, 2011, which was 513,644,248 total impressions across all the data flows. Overall statistics for each traffic flow are outlined in Table 1. We chose to perform our analysis on the local publisher traffic, because it contained the most information for each impression. Out of NETWORKX's 1,600 publishers, only about 300 are active and generating ad requests. Of these 300, the most popular 1% are responsible for roughly 40% of the local traffic, and the top 10% are responsible for 92% of the local traffic. We observed the impressions, clicks, and conversions,

Traffic Flow	Traffic (%)	Impressions (per hour)	CTR	Conversion Rate
Auction	37.7%	305,318	0.16%	-
Publishers	2.0%	15,794	0.56%	0.01%
Arbitrage	60.3%	489,184	0.12%	0.007%

Table 1: Statistics for each traffic flow.

and we measured a click-through-rate (CTR) of 0.56%, with conversions being 2.22%. In addition, we measured an average CPM of $0.084, an average cost-per-click (CPC) of $0.017, and an average cost-per-action (CPA) of $0.055. We found that there are about 1.5 unique cookie IDs per IP address in an hour, which are responsible for 2.4 impressions.

3.2 Establishing Ground Truth

To determine how well analysis and detection models can identify fraudulent behavior, we require ground truth about known bad and known good publishers. NETWORKX's traffic was coming from relatively few publishers, and the remaining publishers had so little traffic that they could not cause noticeable harm to NETWORKX's advertisers. Therefore, we only had to analyze the top 100 publishers. We chose to manually analyze each of the publishers by visiting the referrer URLs of their ad requests. Since it is not trivial to determine what sites are good and bad, we used the following heuristics (many of these were established from observing the fake fraud websites outlined in Section 2.9):

- The site does not serve content, or the content that it serves does not render.
- The site contains entirely ad content or significantly more ad content than actual user content.
- The site is hosting illegal content or content against the terms of use of RightMedia.
- The content on the site is stolen or contains leftovers from an HTML template.

The following heuristics were used for establishing which sites were not perpetrating fraud:

- The site is well-designed, good looking, and usable.
- The site has a very good Alexa ranking, especially compared to their amount of NETWORKX traffic.
- The site is full of legitimate content that has user support, such as comments or "likes" from various social networks.

For a site to be flagged as either good or bad, it had to conform to more than one of the above heuristics in one category and none of the heuristics in the other. Of the 100 publishers analyzed, 11 were picked out as owning sites that likely perpetrated fraud and 20 were picked out as likely not participating in fraud. The finalized ground truth list was brought to the attention of the development team and the CEO of NETWORKX, who verified our work. They also terminated the malicious publishers' accounts as a result of what they found. None of the publishers whose accounts were terminated attempted to contact NETWORKX for outstanding payments or account renewal.

4. IDENTIFYING SUSPICIOUS TRAFFIC

In this section, we present techniques that could be used for detecting fraudulent ad traffic. In particular, we discuss two main approaches: First, we introduce a detector based on checking the referring web page for signs that indicate fraud. Then, we discuss techniques based on statistical properties of various fields in ad requests.

4.1 Reverse Spidering

To detect the malware described in Section 2.8, we developed a reverse auditing system that crawled the referrers of each impression in our data feed. We deployed 50 virtual machines that utilized the SELENIUM [1] software-testing framework for web applications, which has the ability to programmatically control a real Web browser. The primary benefit that Selenium provided was the ability to interpret JavaScript code. For each audit, we collected the raw traffic, and then we extracted the section ID fields on the web page that is responsible for generating revenue for the publisher. Next, we compared the section ID values from NETWORKX's data feed with the observed section ID values found on that referrer's site. Our intuition was that if the referring page did not contain the section ID that initiated the request, then the request must have been hijacked.

Unfortunately, this detection method was not effective in the real world. After several months (August 2010 - April 2011) of running our reverse spidering system, we noticed that 79.2% of the referred pages in NETWORKX's data feed contained no section IDs when examined. Clearly, this high percentage of traffic could not be fraudulent, so we manually analyzed some of the sites that did not contain section IDs. We found that many sites used IP-address-based geo-location to serve specific ads to particular regions. In addition, some of the sites used ad delivery optimization services, such as Casale Media, that dynamically choose different ad exchanges to maximize conversions. In other words, a visitor may have received a RightMedia ad based on his location, but our crawlers, which are based in the U.S., were delivered ads from DoubleClick's ad exchange. Even if we had found RightMedia ad tags with different section IDs than what we would expect, this would not indicate fraud, as one publisher may have many legitimate RightMedia accounts. As a result, we conclude that reverse spidering for the purpose of fraud detection suffers from significant limitations.

4.2 Modeling Ad Requests

We introduce a number of features that model properties of ad traffic. Similar to traditional intrusion detection systems, we use these features to establish models of normal, expected traffic. When certain requests or sets of requests violate a previously-built model, we consider these requests to be suspicious. We use the fraction of suspicious traffic per publisher to detect fraud. More specifically, when a certain publisher produces a high percentage of requests that are suspicious, this publisher is more likely to be involved in malicious activity.

In the next paragraphs, we discuss our approach for building models. Then, we describe the effectiveness of individual features.

4.2.1 Building Models

Features. We considered a number of simple features for detecting anomalous ad traffic:

• *Impressions Per Cookie*: The number of impressions each cookie generated. Our motivation was that, since each cookie ID rep-

resents a unique browser instance, any cookies generating a very large amount of traffic would be fraudulent.

• *CTR Per Cookie*: The click-through-rate for each cookie. On average, CTRs are rarely above 2%, so any cookies generating very high click-through-rates would be suspicious.

• *Publisher Revenue Per Cookie*: How much revenue each cookie generated for a publisher. Our motivation was that for fraud to be effective, fraudulent traffic would have to be generating revenue in addition to just impressions.

• *Unique IP Addresses Per Cookie*: The number of unique IP addresses a cookie is generating requests from. Since cookies are assigned per browser instance, we would not expect one cookie ID to come from many 24 IP subnets within a short period of time.

• *Impressions Per IP Address*: The number of impressions that each IP address generates. Our motivation for this feature is that a naive clickbot might remain at one static IP address and generate many requests.

• *CTR Per IP Address*: The CTR of each IP address. Most users do not click on an ad more than 2% of the time, thus, a very high CTR from a single IP address indicates fraud.

• *Publisher Revenue Per IP Address*: How much revenue each IP address generated for a publisher. Unusually high values are suspicious.

• *Deviation of CTR Per IP Address*: This feature is slightly different than the others because it cannot be computed for traffic in a single time interval. Instead, we compute the standard deviation of the CTR values for a number of consecutive time slots. Malware that commits click fraud often exhibits characteristic behavior in that its click through rates over time are very consistent, even when the number of impressions generated changes. Thus, a low feature value is suspicious.

A number of traditional detection features could not be applied to our dataset due to lack of data in certain fields, especially the IP address field. For example, because we only had the first three octets of the IP address, we could not detect sites that had an unusually high amount of traffic from IP addresses that belong to known Internet proxy services, a behavior that would indicate the presence of static IP address clickbots.

Thresholds. We use a dynamic threshold to determine which features are considered suspicious. Each feature threshold is applied to the feature value computed for a time window of one hour. After that, the threshold is recomputed, based on historical (previously-seen) data. More precisely, a threshold is determined by first computing the mean and variance for a particular feature over all previous time windows. Then, we set the threshold equal to the mean plus N standard deviations. The value of N can be used to tune the sensitivity for each "detector." We empirically determined good values for N based on a subset of the traffic dataset. The concrete values are shown in Table 2.

The threshold for the *CTR standard deviation feature* is computed slightly differently. In particular, suspicious values for the standard deviation are not those values that *exceed* a normal baseline. Instead, a small standard deviation is suspicious, since it indicates a high regularity typically associated with automated (bot) traffic. We empirically found that a value of 0.02 (2%) produces good results.

Anomaly Detection Algorithm	**Threshold (Standard Deviation)**
Impressions Per Cookie	3
CTR Per Cookie	3
Publisher Revenue Per Cookie	2
Unique IP Addresses Per Cookie	2
Impressions Per IP Address	4
CTR Per IP Address	4
Publisher Revenue Per Cookie	3

Table 2: Detector thresholds.

Classifying publishers. For each time window of analysis (one hour), the detectors produce those cookies and IP addresses that violate at least one of the feature thresholds. Using this information, we identify all requests (impressions) that originate from a suspicious IP or that contain a suspicious cookie. In the next step, we determine the publishers that are associated with these requests (based on the publisher ID in the request). Using this, we can determine the number of suspicious and the number of total requests for each publisher and each time window.

Using the number of suspicious and total requests, we can easily compute a publisher's *fraction* of requests that are suspicious (both for individual time intervals and for an entire observation period). We consider this fraction as an anomaly score for the publisher. The anomaly score can be compared to an anomaly threshold; when the threshold is exceeded, we consider the publisher to perform fraudulent activity in the time period under analysis.

4.2.2 Evaluating Models

We wanted to explore the ability of different detectors to identify fraudulent traffic. To this end, we applied the models described in the previous section to our data set (ten days worth of traffic, as described in Section 3.1). We then computed, for each model, an anomaly score for every publisher. Finally, we compared these anomaly scores to varying thresholds (ranging from 0 to 1, in increments of 0.001). This yielded different sets of publishers that would be classified as suspicious or legitimate.

In the next step, we leveraged our ground truth (see Section 3.2) of good and bad publishers. More precisely, for each model and each threshold value, we could determine the fraction of known, malicious publishers that the models would have correctly identified. Also, we could see the fraction of honest publishers that were incorrectly attributed as suspicious.

Discussion. In this paragraph, we discuss the performance of the individual detection features. Overall, we found that no single model would work as a reliable detector. For example, the best detection features with good threshold settings yielded detection rates between 60% and 80%. However, with these settings, a detector would also incorrectly blame 10% of the legitimate publishers. This indicates that there is a significant variety in the ad traffic, and simple, statistical measures are not sufficient to precisely distinguish between malicious and legitimate players. Nevertheless, the simple models are useful to guide a human analyst (employed by ad networks) to focus on those publishers that produce the largest fraction of suspicious traffic and, hence, are most likely to participate in fraudulent activities.

Impressions per cookie/IP features: The number of impressions per cookie and the number of impressions per IP address were the best indicators for fraudulent activity (and cookies perform better than IP addresses). The good performance of the impressions-per-cookie detector is likely a result of cookie replay attacks, where

one cookie ID is used over a long period of time to generate many impressions (see Section 5.1).

CTR per cookie/IP features: Interestingly, the CTR (click through rate) features, both per cookie and per IP, perform very badly for fraud detection. This is most likely because CTRs are closely monitored in the ad industry, and having publishers or users with very high CTRs is one of the simplest indicators of fraud. Thus, fraudsters who want to survive in the exchange need to keep their CTR low to remain undetected.

Revenue per cookie/IP features: Because the number of impressions and the publisher's revenue are related, one would expect that the two feature sets perform fairly similarly. Interestingly, however, publisher revenue models perform much worse. This could be the result of RightMedia's performance-based pricing model, which devalues impressions from publishers who do not exhibit measurable gain to advertisers (in the form of purchases or conversions). Thus, it is significantly easier for fraudsters to generate impressions rather than actual revenue without a sophisticated system or human user to issue conversions. As we will show in Section 5.4, we found a malicious publisher that was generating a very large amount of traffic, but that had an eCPM of only $0.02. Because the eCPM of fraudulent publishers tends to be lower, they are less likely to have unusually high publisher revenue from a single cookie or IP address.

Unique IP Addresses per cookie feature: When studying the ad fraud botnet, we observed that the same cookie was reused by multiple different machines. Thus, one would expect the number of unique IP addresses per cookie to be effective at detecting fraud (in particular cookie replay attacks that are perpetrated across a number of machines). However, we found that this feature produces many incorrect detections. In particular, we found that there are several sites (mostly forums) where users login with the same cookie ID from many different networks (which is also due to networks that frequently assign new IP addresses to its clients). These sites also increased the false positive rate of our impression-based feature because users would remain or revisit the site with the same cookie or IP address over and over again, thus generating many ad requests with that cookie/IP address. The presence of such sites is one of the reasons why gleaning reliable fraud data from an impression stream is difficult for an individual ad network.

Deviation of CTR feature: This model successfully detected a number of malicious publishers that seemed to have used automated tools to convert impressions into clicks. However, since not all malicious publishers were using simple, automated tools to commit fraud, the overall detection capability of this feature was modest.

Note that our models operate under the assumption that each cookie ID and IP address belongs to a unique user, and they report fraud when a cookie or IP exhibits behavior inconsistent with the behavior of a single user. However, as mentioned previously, RightMedia only provides the first three octets of an IP address to an ad network, and, in addition, network address translation (NAT) might hide many users behind a single IP address. As a result, a single IP address value could encompass a large number of individual users. Thus, cookie-based detection techniques typically produce more consistent results than IP-based techniques.

5. OBSERVATIONS

In this section, we examine anomalies that our results indicate as fraudulent activities in the RightMedia ad exchange. In particular, based on our study of ten days of ad traffic, we observed a number of patterns that are associated with fraudulent activity. We looked for similar patterns in our entire data set, which contains traffic for a much longer period of time (specifically, four months from February 2011 until April 2011). This section presents some of our interesting findings.

5.1 Cookie Replay Attacks

As a result of cookie impression analysis on the data feed, we were able to find a number of instances of cookies that had many hundreds or thousands of impressions spread over a week or more. In particular, there was one cookie ID that was consistently generating traffic during the entire time period we analyzed (up to September 2011), and was active before the start of 2011. We observed this cookie's behavior in NETWORKX's local traffic, arbitrage traffic, and auction traffic. Normally, a cookie is associated with a single browser instance, and thus a single unique IP address, browser, and geographic region. However, the data for the suspect cookie indicated that the cookie was coming from 28 different types of browsers, 746 global regions on 666,429 different 24-bit IP subnets, and using 28 browser languages. We also observed this cookie coming from 236 unique local publisher accounts. The cookie's conversion rate is 18 times greater in Table 3 than for the overall traffic in Table 1. Because of the large amount and uncharacteristic nature of this cookie's traffic, we consider it and other cases of cookies with very large amounts of traffic a type of fraud we call *cookie replay attacks*.

The reason why an attacker would randomize the browser version, language, and other fields, but not refresh the cookie value is that cookies set by RightMedia are encrypted with a server-side key, and a Hash-based Message Authentication Code (HMAC) is used to verify its integrity. Thus, generating new values to populate the cookie field of the HTTP header of an ad request is not trivial. To set the cookie properly, the browser must execute some client-side JavaScript embedded within the ad tag that is able to generate the encrypted cookie. Of course, this is possible to automate (our reverse auditing system used Selenium to do it), but this limits potential fraud, both in processing resources and because the IP address field cannot be spoofed when querying the servers for the cookie ID.

From the local publisher traffic flow, which constituted 7% of all NETWORKX's traffic, we calculated that the cookie was generating $32 in revenue for the fraudster per month, and costing advertisers $56 per month. Since these values were collected from only the local traffic flow, the real amount of fraud across the entire exchange would be much larger than this. To get an idea of how much the fraud scales across the exchange, we looked at how much revenue the cookie was generating across all the traffic passing through NETWORKX, and found that the cookie was generating $235 in revenue and $409 in cost every day. However, it is important to highlight the fact that NETWORKX is just one of several hundred ad networks in the RightMedia exchange, so the potential loss due to fraud from this type of operation is likely far greater. Despite the loss from these attacks, it appears that the current fraud systems that are in place are not yet effective enough to detect this type of activity.

Traffic Flow	Impressions (per hour)	CTR	Conversion Rate
Auction	6,103	0.3%	-
Publishers	248	0.6%	0.185%
Arbitrage	13,962	0.6%	0.114%

Table 3: Suspicious cookie statistics for each traffic flow.

5.1.1 Clicks from the Cloud

Interestingly, we identified traffic that originated from Amazon's Elastic Compute Cloud (EC2) and that was being used to perpetrate a portion of these cookie replay attacks. While ad traffic from the cloud is not by itself suspicious (Web users may proxy browser traffic through the cloud), we observed the cookie ID discussed previously being used in a large number of requests that originated from the cloud from April 10, 2011 to April 13, 2011. Thus, we believe that this is a strong indication that attackers are using Amazon's cloud (possibly the free tier that allows for 30GB of transfer per month) to generate fraudulent ad impressions and clicks.

5.2 Spoofing the Referrer

Referrer spoofing is performed by clickbots that want to hide their fraudulent traffic across multiple referrers, so that large numbers of impressions do not come from referring domains that are not very popular or well-known. The bots rotate through a list of referrers while performing the fraud, but they always use a section ID that the fraudster owns. We observed this type of fraud on the click fraud botnet command and control server that we had access to. In February and March of 2010, we observed the command-and-control server issuing large numbers of referrers to the bots that included both the fraudster's sites and popular sites that they could not have owned, such as citibank.com. A year later, in April 2011, we observed the command-and-control server issuing only a few possible referring URLs, which only included the fraudster's fake sites. We suspect that this change in behavior was caused by the fact that RightMedia realized that their traffic was coming from a wide variety of referrers that did not match their registered publisher websites. According to RightMedia's online user guide, "If the [referer] report shows a long list of unfamiliar domains for a publisher, it is likely that the publisher has provided their tags to adware or spyware companies for their own profit" [4]. To avoid being caught by this kind of simplistic detection scheme, a malicious publisher would simply include a less varied list of domains in the `Referer` field of their fabricated HTTP request. In the following section, we analyze other publishers that exhibit similar behavior.

5.3 Unrecognized Referrers

During our manual analysis phase we became interested in looking at unrecognized referrers and why they would be generating impressions for a particular section. We define an *Unrecognized Referrer* as any referring site that did not register the section ID that they are using to generate ad traffic. In RightMedia, section IDs are allowed to be placed on sites other than the site that the publisher had originally registered with, so simply observing that a section is getting impressions from unrecognized referrers is not enough to classify the impressions as fraud. Thus, we were not able to develop an automated way to detect fraudulent activity based on the `Referer` field (which can be spoofed anyway, as we saw in the previous section), but we did unearth a number of attacks that can be perpetrated as a result of this policy. Because local publisher traffic is the only traffic with the `Referer` field set, we could only perform this analysis on local traffic, and we found that unknown referrers made up 43.2% of the traffic. The results of our analysis for April 2011 are outlined in Table 4. We looked both at publishers that had a large number of impressions from unknown referrers and those that used a large variety of unknown referrers. From this, we were able to observe a number of sites whose registered domains were no longer active but were still generating impressions through other domains, which we will discuss in the next section. In addition, this analysis allowed us to identify a malicious publisher who was generating impressions across hundreds of unique referring domains.

Publisher	Unknown Referrers	Impressions
PUBLISHERA	300	3,624,162
PUBLISHERB	46	2,720,146
PUBLISHERC	63	1,640,597
PUBLISHERD	1	1,153,357
PUBLISHERE	55	702,209
PUBLISHERF	19	511,066
PUBLISHERG	6	319,442
PUBLISHERH	22	200,334
PUBLISHERI	6	157,033
PUBLISHERJ	1	155,809

Table 4: Top publishers with unknown referrers (April 2011)

5.3.1 Missing-In-Action Sites

It was interesting to see that many publishers had homepages that were down, yet they were still generating impressions from other referring sites. We call such sites Missing-In-Action (*MIA*) sites. We calculated that out of NETWORKX's 1,600 publishers, 10% had unreachable domains and 5% were 404 errors, which did not include publishers whose domains were now parking pages. While looking at these sites, we observed a specific instance of a local NETWORKX publisher who was performing a kind of misrepresentation fraud with his MIA site. This allowed the publisher to host ads on a page that had illegal content that violated RightMedia's terms of use. First, the fraudster registered as a benign publisher, in this case PUBLISHERC's site, and received a number of section IDs to use on the site from NETWORKX, who did not find anything wrong with the site's content. Instead of placing the ad tags on the benign site, the fraudster placed them on a site that contains illegal content, in this case something like *full-free-games.com*. Because there is no check to ensure the referrer matches the page, impressions generated from *full-free-games.com* still made money for the fraudster.

5.4 Malicious Publishers

By analyzing cookie replay attacks and unknown referrers, we were able to identify a particularly malicious publisher who was the source of a large amount of fraudulent traffic for NETWORKX. This publisher, which we call PUBLISHERA, had three section IDs that had already been shut down by RightMedia for generating fraudulent traffic, but he was still perpetrating fraud with one section ID that had not been flagged. We first investigated this publisher because it had by far the most unknown referring domains and impressions from these domains, as shown in Table 4. Because many of the impressions were coming from seemingly random sites, there was evidence of an ongoing referrer spoofing attack. In addition, we computed that this publisher was generating 20% of the suspicious cookie traffic but accounted for only 0.2% percent of all of NETWORKX's local publisher traffic. After being notified of our results, those in charge at NETWORKX decided to take action and ban PUBLISHERA from their network.

PUBLISHERA's historical data provides us with insight into the amount of money a fraudulent publisher can make through a single ad network. PUBLISHERA was part of NETWORKX from July 2010 to May 2011, and over that period earned approximately $6,700 on 277,043,885 impressions. This means his eCPM was only $0.02. The fact that he had such a low eCPM is evidence of RightMedia's performance-based pricing, which results in the drop of a publisher's CPM if these publisher's impressions do not bring measurable revenue to advertisers.

6. FRAUD IN AD EXCHANGES

Although the RightMedia exchange contains a number of features to monitor the legitimacy of traffic and provide historical reporting to each ad network, we know from analyzing the data feed that a large amount of fraud goes unnoticed. The nature of online ad serving and ad exchanges is such that there is not a strong sense of accountability between entities in the exchange. This is done to protect the privacy of these entities, but it also keeps the ad networks from being able to do adequate checks on the validity of their own traffic.

6.1 Suppression of Data Fields

The lack of accountability in the exchange is likely done to protect each ad network's private ad serving data, but it also makes it very difficult for an ad network to verify the legitimacy of ad requests from partner ad networks, leaving their advertisers open to fraud. In particular, the suppression of the last octet of each IP address is very limiting when trying to find machines exhibiting a particular kind of behavioral pattern. In addition, all brokered auction traffic does not have a referrer or section ID field, and conversions are not reported on the brokered traffic, even if it was sold to a locally-owned advertiser. The suppression of these fields allows a fraudster to register with a malicious or naive network and perpetrate fraud across many networks that may be more vigilant.

6.2 Consistency vs. Flexibility

As we discussed earlier, RightMedia does not verify or enforce the basic premise that the referrer must match the publisher's registered site's domain and assigned sections. The primary reason for this appears to be that RightMedia wants their service to be user-friendly, and it would be inconvenient if a publisher had to re-register and get new section IDs if they change their domain. We were able to identify one instance of a benign MIA site, where the publisher chose to relocate his site to a new domain and keep his old section IDs. Our analysis tools flagged them as suspicious because their original site *benign-golf-site1.com* (obfuscated) gave us a 404 error, but there were a large number of impressions coming from the referrer *benign-golf-site2.com* (obfuscated). Manual inspection verified that the site had legitimate content, and the change of domain most likely came from the owner wanting a more lucrative domain. Finally, we suspect RightMedia does not verify whether the referrer matches the section ID. This might be because RightMedia considers the ad networks to be responsible for monitoring the validity of their traffic and to filter any potentially fraudulent instances.

6.3 Hiding Fraud in the Exchange

The distributed nature of an ad exchange makes it a platform to commit fraud. Except for Yahoo! and the exchange itself, no entity has a full picture of what is going on, and fraudsters use this to appear far less malicious than they are in reality. For example, in Section 5.1, we showed a case where a malicious cookie was generating 300 times more revenue in NETWORKX's auction traffic than in their local traffic alone. However, NETWORKX only sees a small portion of all auctioned traffic in the exchange, thus knowing how much total fraudulent traffic involving this cookie is impossible with our limited view. Moreover, every fraudulent site that we identified, whether it was a fake site (as discussed in Section 2.9) or one of our manually-identified bad publishers (from Section 3.2), had a large number of ads from many ad networks and ad exchanges. So, the total cost of the fraud gets distributed among many independent and often competing entities (DoubleClick and RightMedia, for example), which makes the fraud harder to identify.

6.4 What RightMedia Does Right

RightMedia does not ignore the problem of fraud. Their user interface provides many tools for ad network administrators to identify the most blatant cases of fraud and shut down any malicious accounts or suspicious partnerships. In addition, especially in our experience with NETWORKX, it seems that the general attitude of ad networks is to stay actively involved in their ad serving process to ensure that their advertisers are protected from the worst cases of fraud. RightMedia's built-in malicious behavior detection system is called SCOUR, and it was able to identify the severe case of fraud outlined in Section 5.4 and take steps to limit (but not stop) the fraud. According to the RightMedia online user guide, SCOUR "searches for patterns exhibited by desktop software and flags sections that exhibit what, in our opinion, may be malicious traffic patterns" [4]. Based on the description, it appears that the system focuses on finding bot signatures and flagging publishers who have large amounts of traffic coming from machines with these bot signatures. With a full view of the exchange, a modified version of this system should be able to detect some of the more sophisticated types of fraud outlined in the paper.

7. RELATED WORK

Previous work focused on various aspects of detecting click-fraud. Majumdar *et al.* proposed a content delivery system to verify broker honesty under standard security assumptions [18]. Efficient algorithms for detecting duplicate clicks were proposed by Metwally *et al.* in [19] and Zhang *et al.* in [26]. Studies also have shown how malware can exploit ad networks [7, 10].

Juels *et al.* proposed a cryptographic approach for replacing the pay-per-click model with one where pay-per-action can attract premium rates and unsuccessful clicks are discarded [14]. Immorlica *et al.* studied fraudulent clicks and presented a click-fraud resistant method for learning the click through rate of advertisements [13]. In contrast, Kintana *et al.* created a system designed to penetrate click-fraud filters to discover detection vulnerabilities [16].

Recent work has examined botnets and researchers have infiltrated or seized control of parts of the botnet infrastructure to gain more insight into their inner-workings [15, 22, 24, 25]. Note that these botnets were targeted at sending spam email and engaging in acts of financial theft.

In contrast to previous work, our analysis is the first that uses near real-time data to investigate the problem of ad fraud from inside an ad exchange and from the vantage point of a botnet controller. This offers us a unique opportunity to study the ad exchange structure in depth and to discover its weaknesses. Unfortunately, many ad networks are still reluctant to provide researchers with access to their data streams. As a result, the effectiveness of the proposed method in preventing fraud and even determining the amount of fraud that occurs in actual ad exchanges is not clear.

8. CONCLUSIONS

In this paper, we described how online ad exchanges work and focused in particular on Yahoo!'s RightMedia. We found that the complexity of the ad exchange provides criminals with an opportunity to generate revenue by developing malware that impersonates legitimate user activities. Regrettably, there is a trade-off between the security of the exchange and the flexibility offered to publishers and ad networks to maximize their profits.

Acknowledgements

This work was supported by the Office of Naval Research (ONR) under Grant N000140911042, by the U.S. Army Research Laboratory and the U.S. Army Research Office under MURI grant No. W911NF-09-1-0553, and by the National Science Foundation (NSF) under grants CNS-0845559 and CNS-0905537.

9. REFERENCES

[1] SeleniumHQ. Web Application Testing System. http://seleniumhq.org/.

[2] Secure Accounting and Auditing on the Web. volume 30, pages 541 – 550, 1998.

[3] IAB Interactive Advertising Glossary. http://www.iab.net/wiki/index.php/Category:Glossary, 2011.

[4] RightMedia Exchange Knowledge Base. https://kb.yieldmanager.com/, 2011.

[5] C. Borgs, J. Chayes, O. Etesami, N. Immorlica, K. Jain, and M. Mahdian. Dynamics of Bid Optimization in Online Advertisement Auctions. In *Proceedings of the International Conference on World Wide Web*, 2007.

[6] N. Daswani, C. Mysen, V. Rao, S. Weis, and S. G. K. Gharachorloo. Online Advertising Fraud. In *Proceedings of Crimeware*, 2008.

[7] N. Daswani and M. Stoppelman. The Anatomy of Clickbot.A. In *Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnet*, 2007.

[8] B. Edelman. Securing Online Advertising: Rustlers and Sheriffs in the New Wild West. In *Harvard Business School NOM Working Paper No. 09-039*, 2008.

[9] M. Gandhi, M. Jakobsson, and J. Ratkiewicz. Badvertisements: Stealthy Click-Fraud with Unwitting Accessories. In *Journal of Digital Forensic Practice*, 2011.

[10] F. Hacquebor. Making a Million: Criminal Gangs, the Rogue Traffic Broker, and Stolen Clicks. http://blog.trendmicro.com/making-a-million%E2%80%94criminal-gangs-the-rogue-traffic-broker-and-stolen-clicks/, 2010.

[11] H. Haddadi. Fighting Online Click-fraud Using Bluff Ads. volume 40, April 2010.

[12] Y. Hu. Performance-based pricing models in online advertising. Number March, 2004.

[13] N. Immorlica, K. Jain, M. Mahdian, and K. Talwar. Click Fraud Resistant Methods for Learning Click-Through Rates. *Internet and Network Economics*, pages 34–45, 2005.

[14] A. Juels, S. Stamm, and M. Jakobsson. Combatting Click Fraud via Premium Clicks. In *Proceedings of the USENIX Security Symposium*, 2007.

[15] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2008.

[16] C. Kintana, D. Turner, J. Pan, A. Metwally, N. Daswani, E. Chin, and A. Bortz. The goals and challenges of click fraud penetration testing systems. In *In Proceedings of the International Symposium on Software Reliability Engineering*, 2009.

[17] N. Kshetri. The Economics of Click Fraud. volume 8, pages 45 –53, May-June 2010.

[18] S. Majumdar, D. Kulkarni, and C. Ravishankar. Addressing Click Fraud in Content Delivery Systems. In *Proceedings of the IEEE Conference on Computer Communications*, 2007.

[19] A. Metwally, D. Agrawal, and A. Abbadi. Duplicate Detection in Click Streams. In *Proceedings of the International Conference on World Wide Web*, 2005.

[20] A. Metwally, D. Agrawal, and A. E. Abbadi. DETECTIVES: DETEcting Coalition hiT InïňĆation attacks in adVertising nEtworks Streams. In *Proceedings of the International Conference on World Wide Web*, 2007.

[21] L. Rodriguez. http://www.washingtonpost.com/wp-srv/technology/documents/yahoo_may2006.pdf, 2006.

[22] B. Stock, J. Gobel, M. Engelberth, F. Freiling, and T. Holz. Walowdac âĂŞ Analysis of a Peer-to-Peer Botnet. In *Proceedings of European Conference on Computer Network Defense*, 2009.

[23] B. Stone-Gross, R. Abman, R. Kemmerer, C. Kruegel, D. Steigerwald, and G. Vigna. The Underground Economy of Fake Antivirus Software. In *Proceedings of the Workshop on Economics of Information Security*, 2011.

[24] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2009.

[25] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns. In *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2011.

[26] L. Zhang and Y. Guan. Detecting Click Fraud in Pay-Per-Click Streams of Online Advertising Networks. In *Proceedings of the IEEE Conference on Distributed Computing Systems*, 2008.