PLACEMENT LAUNDERING AND THE COMPLEXITIES OF ATTRIBUTION IN ONLINE ADVERTISING

ABSTRACT

A basic assumption in online advertising is that it is possible to attribute a view of a particular ad creative (*i.e.,* an impression) to a particular web page. In practice, however, the seemingly simple task of ad attribution is challenging due to the scale, complexity and diversity of ad delivery systems. In this paper, we describe a new form of fraud that we call *placement laundering* which exploits vulnerabilities in attribution mechanisms. Placement laundering allows malicious actors to inflate revenue by making ad calls that appear to originate from high quality publishers. We describe the basic aspects of placement laundering and give details of an instance found in the wild that abuses the the intended functionality of the widely-deployed SafeFrame environment. We describe a placement laundering detection method that is capable of identifying a general class of laundering schemes, and provide results on tests with that method.

KEYWORDS

Advertising; Attribution; Fraud; Placement Laundering; Online Security; SafeFrame Fraud

1 INTRODUCTION

The process of delivering ads at Internet scale on a daily basis has many challenges. When a user accesses a web page or uses an app that includes space set aside for ads (this is the ad's *placement*), vast distributed infrastructures are invoked, which facilitate delivery in a timely fashion (typically under 300ms). The delivery of an ad to a user who views the ad on a particular publisher page is called an *impression*. The registration of an impression by an advertiser's ad server is the starting point for payments that flow from advertisers through intermediaries to publishers. The complexity of the ad serving process, the diversity of entities involved and the absolute amounts of money involved provide compelling motivation and opportunities for fraudsters.

In this paper, we describe a new ad fraud threat that we call *placement laundering*. The first public disclosure of this threat appeared in a blog post [2] where the attack was dubbed *domain laundering*. Since that initial disclosure, we have identified many other instances of laundering attacks, some that are highly sophisticated. These discoveries have suggested to us that the name *placement laundering* describes this threat more accurately. We define placement laundering as intentional misrepresentation of the characteristics of an ad placement with the goal of drawing high priced ads to low quality placements that would otherwise draw only low priced ads from advertisers or exchanges. The placement's quality refers to its prestige. Consider that a high prestige placement might cost $10 CPM (Cost Per Mille or per thousand ads) while a low prestige placement might cost $0.01 CPM.

Placement laundering is facilitated by sending fake information along with ad requests to make low quality placements (*i.e.,* on sites with an innocuous-sounding domain) look like they are coming from high quality publishers (*e.g.,* www. nytimes.com or www.wsj.com). This is enabled by several different aspects of the ad delivery eco-system. First, the ad delivery process makes an implicit assumption that the information that is sent from clients to ad serving infrastructure when a page is rendering is reported faithfully. Second, the information available to advertisers is often limited by iframes and the diverse paths that an ad request can follow before it is finally served. This takes advantage of cross-origin restrictions that all mainstream web browsers enforce. Finally, there are no intrinsic capabilities in web or app infrastructure to assure or verify that an ad was delivered to a particular placement. We call this the *attribution problem*.

Three key challenges stem from the attribution problem: detection, mitigation and prevention. In this paper, our focus is on detection. We provide technical background describing how attribution within the ad ecosystem works. We provide a definition for placement laundering and then discuss details of one instance

of this type of fraud. This instance of laundering illustrates concretely fundamental weaknesses in today's attribution mechanisms.

In Section 5 we describe details of a process and implementation that has proved capable of detecting certain types of placement laundering schemes. To the best of our knowledge, there is no prior work describing a process that is capable of generalizing from an individual example of placement laundering. We apply our detection method to a corpus of 434B publisher page view events and 31B ad impressions that were collected during March 2018. We report that our process identifies five distinct laundering schemes. A critical feature of our implementation is that, although our process is informed by proprietary data sources, an equivalent process could be built using standard data sources such as network-level packet traces and WHOIS queries. Validation of our implementation is informed by instrumentation deployed across 2M residential desktop machines. In addition to providing us with ground truth, this data allows us to review functional details about individual schemes. We also provide new details about an instance that was first reported in November 2017 called Hyphbot.

2 AD ECOSYSTEM OVERVIEW

In this section, we provide an overview of the online ad ecosystem. While similar descriptions have been provided in prior studies of ad fraud (*e.g.,* [3–6]) we highlight details that are of particular importance to placement laundering.

2.1 Delivering an Ad to a User

Figure 1 provides an example of data flows that result in an ad being served to a client. In practice there are many different ways in which entities can participate in ad delivery and this example only illustrates one set of critical data flows.

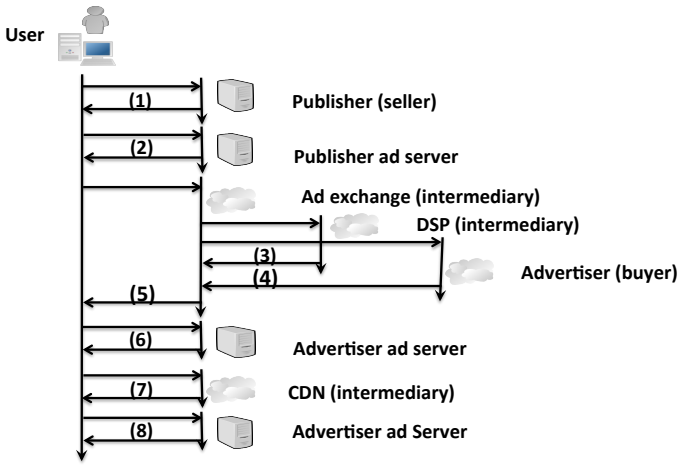

Figure 1: Data flows and entities involved in delivering an ad creative in response to a request from a user.

The process begins with a User requesting a web page from a Publisher. HTML is returned with embedded URLs including those for ad placements (1). Ad requests are forwarded to the publisher's ad server (Publisher ad server), which will select among a number of potential options for delivering the ad. When the choice is made, the server responds with a 302 redirect (2) to the selected ad source, which in this example is an ad exchange (Ad exchange intermediary). The ad exchange can make a number of different decisions about how to respond, including offering the placement to other entities such as Demand Side Partners (DSP) that act on behalf of advertisers who participate directly on the exchange (Advertiser). These entities respond with bids to fill the placement (3,4). The ad exchange makes the final selection and responds with a 302 redirect

to the selected entity (5), which in this example is the advertiser's ad server. The advertiser's ad server selects the appropriate ad creative and responds with a 302 redirect to the Content Delivery Network (CDN) that has the creative locally cached and a 302 redirect back to itself. The CDN responds with the ad creative (7). The second redirect back to the advertiser's ad server is used to register the fact that the ad was served which results in the placement of a tracking pixel on the client (8). It is important to note that a page view is registered by the publisher when it serves the HTML in step 1, while an impression is only registered by the advertiser when the tracking pixel is placed in step 8. These data are used to reconcile billing, which takes place after ads have been served. The multiple redirects in this example also enable intermediaries to place cookies and thereby monitor transactions for billing and other purposes.

2.2 Information Flow Challenges

There are several aspects of the ad serving ecosystem that inhibit faithful communication between participants. First, due diligence by publishers and ad-serving who they partner with varies widely and in some cases is nonexistent. Second, the ability to detect fraudulent views on web pages critically depends on the publisher's site structure as well as the publisher's motivations. Third, there are inherent difficulties in information flow from client browsers to ad-serving entities.

Consider Figure 2, which highlights how nested iframes limit information flow. If code running alongside an advertisement within the innermost iframe attempts to identify the domain that served the outermost frame's content, the result is likely an error. Verification of the publisher is not generally possible due to browser cross-origin policies. Rather, the information that describes the ad's placement must be dutifully passed along, as represented by the chain of i_j's, by all the intermediate parties. In fact, the only interaction that the Advertiser's ad server has is with the client – not with the publisher.

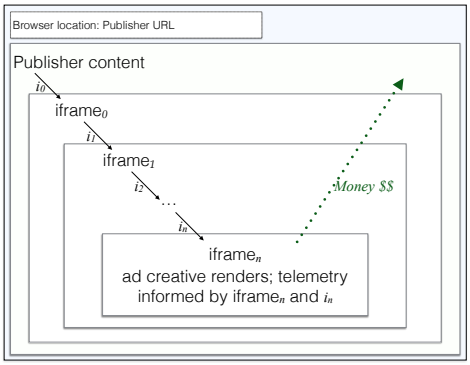

Figure 2: The sequence $i_0 \to i_1 \to \cdots \to i_n$ describes the flow of information within the DOM during the course of delivering an impression. Ad creatives render and campaign telemetry often executes within an iframe that is nested within other iframes. Attribution to the publisher is possible only when the publisher's identity travels between i_0 and i_n without alteration. The flow of money is generally opposite the flow of information.

To summarize, accurate measurement of *attribution* is a challenge for fundamental reasons. Attribution within the ad ecosystem currently relies on trusting the veracity of information that has been passed along by an unknown number of third parties.

3 Methods of Attribution and Placement Laundering

In this section we describe attribution methods that are commonly employed in the ad ecosystem today. We also note the limitations of these methods, which offer opportunities to fraudsters.

3.1 Standard sources of information and their veracity

A common technique for collecting ad campaign telemetry is to send a snippet of Javascript code called a *tag* along with the ad creative when it is delivered to a client. When the tag executes in the client browser, it inspects the local environment and requests the URL of a pixel appended with query parameters (*i.e.,* steps 7 and 8 in Figure 1). This pixel, called a *tracking pixel*, may be served from a host operated by a CDN and is rendered invisibly by the browser. Campaign reporting and attribution is often *completely dependent* on the information in query parameters.

Javascript telemetry can provide a wide range of information about a client and context of an ad, *e.g.,* details of user actions, the user environment, page properties and even portions of the raw HTML page. Unfortunately, identifying the publisher page on which an ad is placed requires information from the top-level browser page. As discussed in Figure 2, this information is typically inaccessible.

Finally, in order that payments for a delivered impression flow and that everybody involved in the delivery process receives credit, account identifiers and tag identifiers that are presented to ad exchanges, DSPs and other sources of ad inventory must be accurate. This is true even when the traffic is otherwise misleading.

3.2 Placement Laundering

We define placement laundering as intentional misrepresentation of the characteristics of an ad placement with the goal of drawing high priced ads to low quality placements that would otherwise draw only low priced ads from advertisers or exchanges.

Companies that buy ad campaigns often require that their advertisements avoid placement on sites that host controversial or offensive content, *e.g.*, hate speech. But if a fraudster is able to substitute an ad's correct attribution on such a site with the name of an innocuous-sounding domain, then laundering is occurring. With the true placement cloaked, the perpetrator is free to engage with a larger set of ad serving platforms and ad campaigns.

3.3 On Validating Placement Laundering Schemes

Validating instances of laundering requires supplementing misleading telemetry with ground truth. As already discussed, the simplest idea, namely to deploy JavaScript alongside a brand's campaign creatives and to instruct the JavaScript to inspect, and then report about the top-most element of a page's DOM, does not work. This is because advertisement creatives are generally rendered within iframe elements and cross-site scripting policies, which are fundamental to the browser security model, restrict what is observable from within such an environment.

An alternative idea is to instrument both advertiser campaigns as well as publisher pages with JavaScript-based telemetry. Analysis of the two data streams reduces to reconciling the logs. However, our experience is that this approach results in an unacceptably high false positive rate. We conjecture that reasons include browser handling of third-party cookies, the widespread use of ad blockers, non-comprehensive deployment of telemetry by publishers and technical complications that arise from advertisements that load and refresh asynchronously with page views.

Since general automated detection is still an unsolved problem, we establish ground truth via a forensic, manual review of suspected instances of laundering. This can involve a direct manual review of all three data types (panel, census and ad campaign - explained below) as well as direct observation of browser behavior. This latter technique often implies an inspection of browser HTTP traces of suspect URLs. Despite the obvious limitations of this technique, it is a rich source of information to either confirm or refute the presence of fraud.

4 Case Study: Fraud within the SafeFrame

Placement laundering attacks can happen in many ways, ranging from simple referrer spoofing to far more complex methods where malware alters system parameters. Of the many cases of placement laundering that we have verified, in this section we describe one illustrative instance. Malicious software is *not* required to be installed on the client in order to execute most of the schemes that we have encountered.

4.1 Our data sources

Three types of data inform our view of Internet traffic: publisher page view events, ad campaign impression events, and panelist web traffic records. Each plays an important role in the process of detection and verification of placement laundering. We describe each of these data types, why each is collected, and include high-level statistics to provide a sense of scale.

- **Publisher census** Publishers often partner with third parties to measure audience demographics and reach and also to participate in publisher ranking reports. Participation is straightforward and requires including a Javascript tag or the URL of a tracking pixel on their pages which fire when a browser renders the page. We utilize data from comScore's publisher census, whose participants include 90 of the top 100 US-based publishers and thousands of other domains. Census data typically records over 14B events daily. The information from this source that was used in our study includes timestamp, URL, referrer, and cookie.
- **Ad campaign tags** Similar to publishers, companies that run ad campaigns partner with third parties to verify audience reach and other features of ad campaigns. JavaScript tags are delivered with ads that appear in placements on publisher pages. Impression delivery and campaign accounting must be done in accordance with well-established industry standards concerning viewability, brand safety and audience targeting [1]. As a result the typical ad tag is a fairly complex JavaScript program. The information returned to the third party is often a rich description of the browser and context in which the code executed. comScore deploys its own ad campaign telemetry, the collected data exceeds 1B impressions per day.
- **Panel data** User panels are another way in which third party measurement groups collect information on publisher and ad campaign activity. We utilize data from comScore's panel of 2M voluntary user participants. Each panelist provides detailed personal information and agrees to install monitoring software, called the Meter, on her host machine. This software can report all HTTP(S) traffic to/from their host machine. It also reports detailed diagnostic information such as process names and other identifiers to comScore. Panel participation is voluntary, *i.e.*, panel data is collected with informed consent. comScore is highly sensitive about user privacy issues. Policy details can be found at `http://www.comscore.com/About-comScore/Privacy`.

Case Study: SafeFrame SafeFrame is the industry-standard API-enabled iframe whose functionality is defined by the Interactive Advertising Bureau (IAB). Its purpose is to allow information to pass between ad creatives and publishers in a secure and standardized manner. Google's own implementation of SafeFrame, which was the target of this scheme, is integrated with their Doubleclick for Publishers platform. It has extremely wide deployment and is described in the Google Publisher Tag library documentation. This environment is host to billions of advertisements per day. The initial public disclosure of this issue occurred in close coordination with the Google Traffic Quality team[12].

Figure 3 illustrates how this attack works: An impression's attribution is often reported by including the impression's hosting iframe's URL as a query parameter within a pixel request. Typically, URLs are %-encoded using the `escape` (or some similar) method. In this attack, the global methods of the browser's `window` object are overwritten with malicious code. Third parties who access these global methods will, instead of running native browser code, instead execute malicious code.

The implications of this attack are significant. *Any tag that executes within the tainted environment and does not first verify the integrity of global JavaScript methods is at risk.* Based on direct measurments of campaign data, we observed daily volume varying between 1M and 5M impressions per day over a 30-day period. This attack was active during January 2016.

5 DETECTION AND MEASUREMENT

We now describe details and results of a process that is capable of detecting a general class of placement laundering schemes. Many examples of laundering exhibit characteristics that are trivial to identify, but only

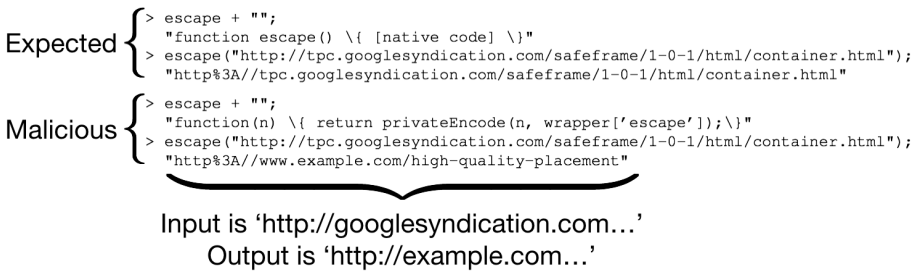

Figure 3: The SafeFrame attack begins when a SafeFrame iframe is populated with a malicous source URL. This URL hosts Javascript that, when executed, overwrites the global `escape` method and several others. All JavaScript that executes within the same scope as the malicious code is impacted.

after the instance has been discovered. Designing a process that is able to identify new schemes without relying on *ad-hoc* manual forensic investigation has proved to be a significant challenge. To our knowledge, no prior published work describes a process that generalizes.

5.1 Background

Our methodology targets a general class of placement laundering instances. A representative from this class was reported in November 2017, which the popular press dubbed "Hyphbot" [13, 14]. In terms of scale, press reports state that Hyphbot was facilitated by a fleet of 500K machines that were infected with malware and that it cost the industry $500K/day. In Section 5 we discuss our findings on Hyphbot and other instances that, to our knowledge have not been reported before.

While our detection method relies on proprietary data, namely a panel data source, our method could equally well leverage data sources other than a panel. For example, for unencrypted traffic, similar signals would be observed by sniffing packet-level data on a network.

We now briefly describe key aspects of attack that enables detection. When a web browser initiates an HTTP request, the message is routed by mapping the domain name appearing in the URL to an IP address. This mapping usually occurs via the global Domain Name System (DNS), though operating systems commonly maintain a local configuration file (*e.g.*, `/etc/hosts`) that also maps domain names to IP addresses. The mappings in the local file have precedence over whatever the global DNS reports. An attack on either the local configuration file or the global DNS that results in high quality domains mapping to non-standard IP addresses will result in the same signature that we seek.

We find empirically that the majority of high quality publishers host their content on the hardware operated by a small number (often just one or two) Internet Service Providers (ISP). The publisher-ISP relationship changes very slowly over time. Conversely, malicious actors can rent low-cost, easy-to-configure web servers using cloud-based services. Our methodology does risk flagging arbitrary traffic that is delivered from a CDN that uses IP addresses that are owned by multiple ISPs to serve content. However, our experience is that in practice, we are robust against this scenario.

5.2 Method and Results

The essential idea of our detection method is to identify IP addresses that are associated with multiple unrelated high quality publishers on panelist machines. This can happen for a variety of reasons including but not limited to a customized `/etc/hosts` file or some other DNS hijack attack. The output of our process is a key-value pair, where the key is the tuple (IP, ISP) and the value is a set of distinct domains that resolved as the IP address in the key by panelist machines. We also store the name of the processes that initiated each HTTP(S) request. We use process name as part of followup analysis to confirm or refute the existence of malware on an indiviudual machine. Observe that the (IP, ISP) pair is the server-side of the HTTP request while the process name is purely client-side (*i.e.*, it is not transmitted over the network).

Our method, which has been calibrated to achieve a low false-positive rate, runs as follows:

- A set of candidate domains is built.
  - Each domain is high quality (this is defined below) and
  - Each domain is observed resolving to several IP addresses and at least two distinct ISPs are represented.
- If an ISP owns an IP address that resolves to at least twenty high quality domains that live in the candidate domain set, then the (IP, ISP) tuple and the set of high quality domains that resolve to the IP are among the key-value pairs of the output.

We label a domain *high quality* by referencing a list of publisher domains maintained by comScore. The Alexa ranking would generate similar results. We restrict our view to the top-ranked 2,000 domains. We find that around 325 high quality domains are typically affected at any point in time.

To label an (IP, ISP) key a true detection, we require that traffic associated with it be generated by process names that are associated with known malware. This additional restriction means we identify only a subset of schemes, but we are highly sensitive to false positives and this step limits alerts to those that are likely to be true positives We note that if a process has the ability to alter protected domain resolution files (*e.g.*, `/etc/hosts`), then it certainly can launch a separate process that has an inconspicuous name.

Label	# ISPs	# IPs	# Days seen in March	# of top 2K domains	# Machine IDs	Average # of daily requests
Hyphbot	4	25	31	726	141	326,239
Scheme$_\beta$	1	4	31	208	15,901	60,328
Scheme$_\gamma$	1	4	31	163	13,619	18,878
Scheme$_\omega$	1	1	30	364	134	6,704
Scheme$_\lambda$	1	6	6	168	80	1,611

Table 1: We identify five schemes by applying our process to the March 2018 data snapshot. High-level statistics about each scheme are shown.

Over the course of a day, our methodology typically flags $O(10)$ candidate (IP, ISP) pairs. The set of pairs detected by our process is fairly stable for timescales of about 1 week. We observe the ISP space is more stable than the IP address space, which aligns with our understanding of how cloud services allocate their resources to their customers. The set of ISPs identified by our process changes slowly, with $O(1)$ emerging or disappearing each week. Manual review of the ISPs that are identified suggests that cloud-based infrastructure is being used.

Instances of laundering are distinguished from each other by several features, including server-side infrastructure, client-side process names, the User Agent field of the HTTP request and URL structure. Identifying the distinguishing features of laundering schemes is currently a manual effort. Automation of this would represent an enhancement, but is separate from our main goal of detection so we leave it to future work. The purpose in manually distinguishing unrelated schemes here is to show that our detection method generalizes. Within our data snapshot, we identify five distinct instances of placement laundering. Table 1 displays a high-level summary.

To quantify the differences among schemes identified, we use the Jaccard index of the set of domains that are among the top 2K domains. The Jaccard index is a statistic that applies to set pairs. For arbitrary sets A and B, the Jaccard similarity index between A and B is $J(A,B) := |A \cap B|/|A \cup B|$. The top 2K domains are all likely targets of schemes, so this metric alone is not sufficient to distinguish unrelated schemes from each other. That said, it is illustrative. Table 2 lists the Jaccard index between the schemes we classified.

Schemes are often distinguished from each other via the executable process names that generate their traffic. The executable(s) associated with Hyphbot and Scheme$_\omega$ were recorded as either whitespace or empty strings. We hypothesize that this is deliberate and that the process name is configured to evade detection.

We now compare our observations about Hyphbot to earlier public reports. The regular expression patterns derived from URL lists that were described in the press [14] match 1.5% of the traffic that our method

	Hyphbot	Scheme$_\beta$	Scheme$_\gamma$	Scheme$_\omega$	Scheme$_\lambda$
Hyphbot	1	0.09	0.08	0.30	0.18
Scheme$_\beta$	-	1	0.32	0.07	0.04
Scheme$_\gamma$	-	-	1	0.05	0.03
Scheme$_\omega$	-	-	-	1	0.39
Scheme$_\lambda$	-	-	-	-	1

Table 2: The Jaccard index between the sets of domains that are associated with each scheme is shown. Only the upper triangular portion of the matrix is populated since the Jaccard is symmetric, *i.e.*, $J(A,B) = J(B,A)$.

identified. We have observed schemes evolve over time, and detection methods that are informed by URL lists or regular expressions are easily circumvented. We are able to estimate impact as 250K impressions per day. We arrived at this figure by matching URLs that resolve to Hyphbot infrastructure to direct measurements of campaign traffic.

6 Related Work

Fraud in online advertising has been addressed in a large body of prior work over the past decade. Click fraud received significant attention as search-based advertising grew in popularity. These pay-per-click-based advertising systems led to a variety of threats including botnets that had specific capabilities for click fraud [17–19]. Examples of botnets with click fraud capability include the Bamital botnet [20] and the ZeroAccess botnet [21]. Prior work has also focused on developing methods for identifying click-fraud *e.g.,* [22,23]. Haddadi develops the idea of using *bluff ads* for measuring fraudulent clicks and to generate IP blacklists to mitigate the threat [24]. Similarly, Dave *et al.* develop novel methods for measuring and detecting click fraud in ad networks [4,25]. Thomas *et al.* conduct a large-scale empirical study of the impact of *ad injectors*, which are browser extensions that overwrite ads that would otherwise be delivered to users [26]. Our efforts are similar to these in that we use logs of billions of ad impressions as the basis for our investigations. To the best of our knowledge there are no prior research studies on placement laundering.

7 Summary and Conclusions

The huge amounts of money that flow though the online advertising ecosystem make it a compelling target for fraudsters. The scale, diversity and complexity of the systems and processes that are employed to deliver online ads offer a variety of opportunities for fraud. While some methods for fraud are well known and can be defeated through careful monitoring and detection techniques, new threats are always emerging.

In this paper, we report on a new form of ad fraud that we call placement laundering. The objective of placement laundering is to offer ad placements on low quality sites that appear as if they are coming from higher quality sites in order to extract higher payments for each ad that is delivered. If done effectively, this can result in an increase in payments per ad by several orders of magnitude. The result is an increase in revenue without an increase in traffic. The basis for placement laundering is the fact that there are no inherent mechanisms for ensuring that an ad was delivered to its intended placement on a publisher page. We call this the attribution problem.

We illustrate the details of placement laundering methods with one case study that highlights how misrepresentations of placements can result in ad requests being attributed to higher quality publishers. This instance is significant because it exploits the functionality of the industry-standard API called “SafeFrame”.

We describe and report on a placement laundering detection method. We show this method is capable of identifying traffic from a general class of laundering schemes. The method provides invaluable insight into the evolution and birth of different placement laundering schemes. Importantly, it generalizes. To validate our method, we report on observable characteristics particular to each scheme. Finally, we report on the overall impact of the general class of placement laundering we target.

REFERENCES

[1] IAB: Interactive Advertising Bureau. http://www.iab.com/

[2] Kline, J.: Domain Laundering Emerges as Significant Threat in Digital Ad Ecosystem. http://www.comscore.com/Insights/Blog/Domain-Laundering-Emerges-as-Significant-Threat-in-Digital-Ad-Ecosystem (December 2014)

[3] Springborn, K., Barford, P.: Impression Fraud in Online Advertising via Pay-Per-View Networks. In: Proceedings of USENIX Security Symposium, Washington D.C. (August 2013)

[4] Dave, V., Guha, S., Zhang, Y.: Measuring and Fingerprinting Click-Spam in Ad Networks. In: Proceedings of ACM SIGCOMM '12, Berlin, Germany (December 2013)

[5] Stone-Gross, B., Stevens, R., Kemmerer, R., Kruegel, C., Vigna, G., Zarras, A.: Understanding Fraudulent Activities in Online Ad Exchanges. In: Proceedings of ACM Internet Measurement Conference (IMC '11), Berlin, Germany (November 2011)

[6] Zhang, Q., Ristenpart, T., Savage, S., Voelker, G.: Got Traffic? An Evaluation of Click Traffic Providers. In: Proceedings of WebQuality '11, Hyderabad, India (March 2011)

[7] LUMAPartners: Lumascape. http://www.lumapartners.com/resource-center/ (2015)

[8] Thomas, C., Kline, J., Barford, P.: IntegraTag: a Framework for High-Fidelity Web Client Measurement. In: ITC 28 Conference, Würzburg, Germany (September 2016)

[9] Facebook: Link Shim - Protecting the People who Use Facebook from Malicious URLs. https://www.facebook.com/notes/facebook-security/link-shim-protecting-the-people-who-use-facebook-from-malicious-urls/10150492832835766 (2012)

[10] W3C: Referrer Policy W3C Working Draft. http://www.w3.org/TR/referrer-policy/ (2016)

[11] Mozilla: RefControl. https://addons.mozilla.org/en-US/firefox/addon/refcontrol/ (2016)

[12] Release, P.: comScore Labs Discovers Placement Laundering Attack in SafeFrame. http://www.comscore.com/Insights/Press-Releases/2016/4/comScore-Labs-Discovers-Placement-Laundering-Attack-in-SafeFrame (April 2016)

[13] O'Reilly, L.: Fake-ad operation used to steal from publishers is uncovered. Wall Street Journal (2017)

[14] AdForm: How adform discovered hyphbot. `https://site.adform.com/media/85132/hyphbot_whitepaper_.pdf` (November 2017)

[15] Lab, I.T.: Ads.txt – Authorized Digital Sellers. `https://iabtechlab.com/~iabtec5/wp-content/uploads/2016/07/ADSTXT_OpenRTB_Transparency.pptx.pdf` (2017)

[16] Derke, J., Richter, N., Bjorke, P.: Transparency and the Evolution of OpenRTB – Addressing Counterfeit Inventory. `https://iabtechlab.com/~iabtec5/wp-content/uploads/2016/07/ADSTXT_OpenRTB_Transparency.pptx.pdf` (2017)

[17] Alrwais, S., Gerber, A., Dunn, C., Spatscheck, O., Gupta, M., Osterweil, E.: Dissecting Ghost Clicks: Ad Fraud via Misdirected Human Clicks. In: Proceedings of the ACM Annual Computer Security Applications Conference, Orlando, FL (December 2012)

[18] Miller, B., Pearce, P., Grier, C., Kreibich, C., Paxson, V.: Whats Clicking What? Techniques and Innovations of Todays Clickbots. In: Detection of Intrusions and Malware, and Vulnerability Assessment. (2011)

[19] Daswani, N., Stoppelman, M.: The Anatomy of Clickbot. A. In: Proceedings of the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA (April 2007)

[20] Kirk, J.: Microsoft, Symantec Take Down Bamital Click-fraud Botnet. http://www.infoworld.com (February 2013)

[21] Magazine, I.: ZeroAccess is Top Bot in Home Networks. http://www.infosecurity-magazine.com (February 2013)

[22] Zhang, L., Guan, Y.: Detecting Click Fraud in Pay Per Click Streams of Online Advertising Networks. In: Proceedings of the International Conference on Distributed Computing Systems, Beijing, China (June 2008)

[23] Metwally, A., Agrawal, D., Abbadi, A.E.: Using Association Rules for Fraud Detection in Web Advertising Networks. In: Proceedings of the International Conference on Very Large Databases, Trondheim, Norway (August 2005)

[24] Haddadi, H.: Fighting Online Click-fraud Using Bluff Ads. ACM SIGCOMM Computer Communications Review **40**(2) (2010)

[25] Dave, V., Guha, S., Zhang, Y.: ViceROI: Catching Clickspam in Search Ad Networks. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Helsinki, Finland (August 2012)

[26] Thomas, K., Bursztein, E., Grier, C., Ho, G., Jagpal, V., Kapravelos, A., McCoy, D., Nappa, A., Paxson, V., Pearce, P., Provos, N., Rajab, M.: Ad Injection at Scale: Assessing Deceptive Advertisement Modifications. In: Proceedings of the IEEE Security and Privacy Conference, Oakland, CA (May 2015)

[27] Atkins, D., Austein, R.: Threat analysis of the domain name system. In: DNS). RFC 3833, Internet Engineering Task Force. (2004)

[28] Jackson, C., Barth, A., Bortz, A., Shao, W., Boneh, D.: Protecting browsers from dns rebinding attacks. ACM Transactions on the Web (TWEB) **3**(1) (2009) 2

[29] Bernstein, D.: The libresolv security disaster. `http://cr.yp.to/djbdns/res-disaster.html` (2002)

[30] Schneier, B.: Lesson from the dns bug: Patching isn't enough (2008)

[31] Fielding, R., Reschke, J.: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. `https://tools.ietf.org/html/rfc7230#section-5.4` (2014)

[32] Bureau, I.A.: Invalid Traffic Detection and Filtration Guidelines Addendum. http://www.iab.com/tag (2015)

[33] Google, I.: Ad Traffic Quality Resource Center. http://www.google.com/ads/adtrafficquality/index.html (2013)

[34] Kelleher, T.: How Microsoft Advertising Helps Protect Advertisers from Invalid Traffic. http://community.bingads.microsoft.com (November 2011)

[35] Yahoo: Traffic Quality: We Work to Protect You in a Variety of Ways. http://advertisingcentral.yahoo.com (2013)