Clicktok: Click Fraud Detection using Traffic Analysis

Shishir Nagaraja, Ryan Shah
{shishir.nagaraja,ryan.shah}@strath.ac.uk
University of Strathclyde

ABSTRACT

Advertising is a primary means for revenue generation for millions of websites and smartphone apps. Naturally, a fraction abuse ad networks to systematically defraud advertisers of their money. Modern defences have matured to overcome some forms of click fraud but measurement studies have reported that a third of clicks supplied by ad networks could be clickspam. Our work develops novel inference techniques which can isolate click fraud attacks using their fundamental properties. We propose two defences, *mimicry* and *bait-click*, which provide clickspam detection with substantially improved results over current approaches. Mimicry leverages the observation that organic clickfraud involves the reuse of legitimate click traffic, and thus isolates clickspam by detecting patterns of click reuse within ad network clickstreams. The bait-click defence leverages the vantage point of an ad network to inject a pattern of bait clicks into a user's device. Any organic clickspam generated involving the bait clicks will be subsequently recognisable by the ad network. Our experiments show that the mimicry defence detects around 81% of fake clicks in stealthy (low rate) attacks, with a false-positive rate of 110 per hundred thousand clicks. Similarly, the bait-click defence enables further improvements in detection, with rates of 95% and a reduction in false-positive rates of between 0 and 30 clicks per million – a substantial improvement over current approaches.

1 INTRODUCTION

By definition, click fraud generates no revenue for the advertiser, but inflicts losses on tens of thousands of online advertisers in the order of hundreds of millions of dollars each year [37, 39]. Typically, click fraud is generated by malicious applications (apps) and malware, and is responsible for around 30% of click traffic in ad networks [11, 17].

Early threshold-based defences demonstrated a focus on the volumes of click fraud from bad-listed sources, but failed simply because attackers were able to swiftly discard their publisher accounts after receiving reputational hits from ad network defences [12, 51], and opened new ones. The decrease in the cost of botnet rentals in the underground economy has been a primary driver of fraud [1, 38, 47]. Click botnets can generate massive amounts of fake traffic and ad impressions, by automatically clicking on websites and apps in large numbers. The reduction in the fixed costs of generating clickspam, by several orders of magnitude, significantly reduces the number of fake clicks per host required to run an economically sustainable click fraud operation. Assuming the earning potential of 0.5 cents per click, which is at the lower end of the spectrum, an attacker can cover operational costs with just three to five fake clicks a day, with a few more to run their operation profitably. Based on this, traditional threshold-based defences [11, 12, 32, 33] fail as the levels of click fraud per source goes below the detection threshold.

Subsequently, ad network defences have evolved to generate liveness-proofs by running machine learning models on advertisement clickstreams to distinguish bots from humans. As a response, attackers mimic the actions of legitimate device users in order to generate credible click fraud [19, 39]. Recently, a serious increase in organic click fraud has been noted via mobile malware. Fraudsters develop seemingly legitimate apps or purchase those with high reputation scores. These apps carry out a legitimate activity, such as controlling the torch, but also serve as a mechanism for mining the (organic) click activity of the device user. Moreover, attackers then launder mined clicks back through their installed user-base. Since the click fraud is based on legitimate traces, the clicks are able to pass through ad network filters. The exception is where the attack violates a threshold, such as using a small pool of IP addresses to carry out the attack. Ultimately, this motivates the need for automated detection techniques that can scale the detection of click fraud attacks, whilst ensuring the integrity of the digital advertising ecosystem.

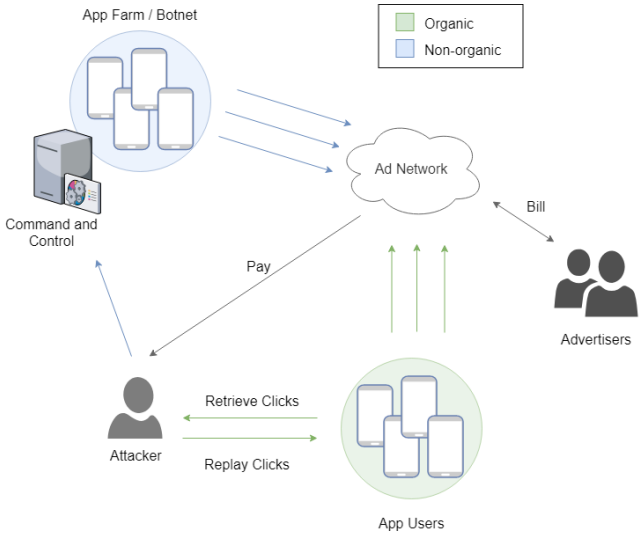

Figure 1: Organic and Non-Organic Click fraud

To address this, we developed Clicktok, a statistical technique that detects clickspam by efficiently searching for *reflections* of click traffic, encountered by an ad network in the past. Clicktok is based on exploiting the timing properties of click traffic, and also unifies the technical response, offering a defence technique which isolates both organic, and the relatively simpler, non-organic click fraud attacks. The focus of this work is on generic click fraud defences, rather than analysing individual click-modules. Current efforts to defeat click fraud have primarily focused on fraud measurement techniques [11], measurement and analysis of publisher fraud [7, 12], and ad placement fraud [29]. While publisher and affiliate- marketing fraud are doubtless of importance, there is limited work that focuses on detecting click fraud from ad network clickstreams supplied to advertisers.

Upon evaluation of our defence, we report significantly improved detection and estimation over past efforts [11, 12, 31–33].

Contributions: First, we have developed two clickspam defenses: the *mimicry defence* and the *bait-click defence*. Second, we have developed a unified algorithm for detecting both organic and non-organic clickspam, which provides the engineering advantage that ad networks need only to adopt a single defence. Finally, our techniques support attribution via their ability to separate malware clicks embedded within legitimate clickstreams. This demonstrates the usefulness of the algorithm in building passive and active click fraud defences using real-world data.

Roadmap: We start by giving a detailed problem description in Section 2. In Section 3, we describe our overall approach and algorithm. We then evaluate the performance of our algorithm on click fraud traffic, embedded within real click traffic data in Section 4. Related work is situated at the end of the paper.

2 THE CLICK FRAUD DETECTION PROBLEM

Click fraud involves directing fraud clicks (or clickspam) at online advertisements (ads), concerning three parties: an advertiser, a publisher, and an ad network. The advertisers participate in a keyword auction, organized by the ad network, where ads owned by the winning advertiser are submitted for circulation by the ad network. The publisher's role is to render the advertisements that are provided by the ad network. When a user clicks on an ad, the ad network: receives the request, updates the billing account of the corresponding advertiser, and redirects the click to a URL of the advertiser's choice. For each click on an ad, the advertiser pays the ad network, who in turn pays the publisher a substantial fraction of the per-click revenue ($\approxeq$70%).

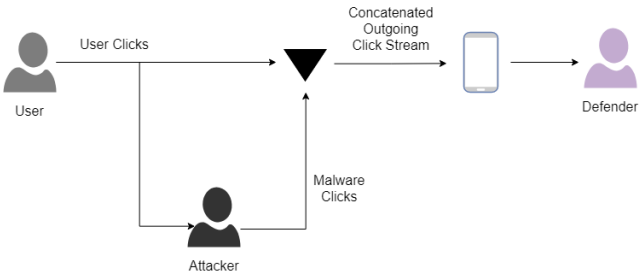

Figure 2: Click fraud detection problem

The click fraud problem is the challenge of distinguishing between clickspam and legitimate clicks, given that the attacker has full knowledge of legitimate click traffic (Figure 2). Typically, an attacker registers as a (large number of) publishers with the ad network. They deploy techniques to send clickspam to advertisements, allowing them to gain a fraction of the money paid out by the advertiser for every (fake) click.

Terminology: A paper on click fraud makes frequent use of certain terminology, which we now cover for ease of readability. *Click traffic*, *click stream*, or *click traces* all refer to an ordered sequence of one or more clicks each corresponding to an advertisement. Clicks are termed *legitimate* or *organic* when generated by a genuine user. Clicks generated for the motive of profit are considered *fake*, *fraudulent*, or *clickspam*. Clickspam generated using legitimate traces is termed *organic* clickspam.

2.1 Challenges

Efficient clickspam detection involves several challenges. First, an ideal detection system should be able to directly observe human input on end user devices, and label click traffic that is suspected to be illegitimate; however this is an impractical approach. Second, legitimate click traffic that arrives at an ad network likely dwarfs the *relatively* smaller amount of clickspam. Third, the variable form and structure of click traffic poses an issue when constructing an accurate baseline model of legitimate click traffic, that is broadly applicable. Finally, click malware can employ a variety of stealth techniques to evade detection, in particular the use of organic click traffic and reducing the number of fake clicks per source, to well below the level set by threshold-based click fraud detection techniques.

2.2 Opportunities

Although adapting fake clicks to match the statistical characteristics of legitimate clickstreams is undoubtedly a stealthy approach, we note that this can be used as a point of detection. Based on this, we identify both passive and active approaches to defence.

Passive Approach: We argue that legitimate click activity has copy-resistance properties, owing to the uncertainty of inter-click times. Therefore, to accurately mimic a user's click activity, an attacker would need to model the timing behaviour of the user, which has some uncertainty that forms the basis for our passive defences. Furthermore, click generation techniques, that are encoded by malware authors, may result in correlations in the inter-click times of clickstreams across users. Thus, another approach is to consider the relative increase in the correlation across clickstreams, due to click fraud.

Active Approach: While considering a passive approach to defending against click fraud, a radically different approach is to consider active interventions. By adding **bait clicks** to legitimate user traffic, we can attract the attention of malware. To measure and understand malware click generation strategies, malware adaptation to a bait clickstream may be a promising approach. This is a novel idea for instrumenting click malware — the ad network injects a pattern of clicks into a user device over a period of time, via a client-side scripts executed by the browser. Only malware will respond to this click pattern, mistaking it for legitimate activity, whilst the ad network ignores these clicks. If click malware is to be present on the device and adapts to the user's activity, it will generate clickspam that can be readily isolated.

3 INFERENCE SYSTEM

3.1 System Architecture

We propose an inference system, depicted in Figure 3, which takes two inputs, click timestamps from the ad network, and an optional seed clickspam input from a bait ad farm. The ad network contains servers which run *click traffic monitors* that store click timestamps. Optionally, to supplement information for click traffic monitors, our inference system may also receive input from a bait ad farm (honeynet) [17]. Classification information from these may be used as an

input to Clicktok, where the suspicion of click fraud is known with a higher probability.

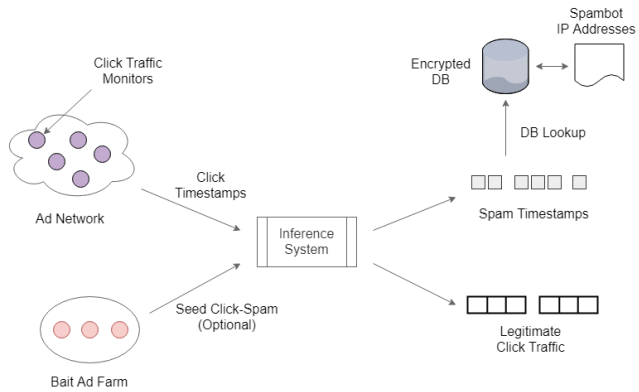

Figure 3: System Architecture

Our primary focus is to design a generic inference algorithm that is, first, based on the fundamental limitations of automated fake click generation techniques, and second, can address both organic and non-organic clickspam.

Clicktok works on the core observation that both organic and inorganic clickspam cause an increase in redundancy, albeit differently within ad network clickstreams. In the case for organic click fraud, to isolate the source of redundancy, we use a compression function in combination with a clustering algorithm, to isolate click traffic whose *timing patterns* are similar to past timing patterns. A timing pattern is an ordered ascending sequence of time offsets, relative to an absolute start time.

Similarly, we noticed that the same intuition can be leveraged to isolate inorganic clickspam. For instance, where malware generates traffic using randomised generators, the traffic with high entropy timing patterns can be clustered together, by exploiting their non-compressibility. Likewise, the injection of small amounts of clickspam per device are evident, when traffic from multiple end-user devices is considered together, thus exploiting the common patterns across infected devices. Ad networks, or backbone routers, provide us with a vantage point into click traffic tainted with clickspam.

The primary challenge in partitioning on the basis of inter-click times, is that both clickspam and legitimate clicks may not be temporally separated. As Figure 5 be superimposed over each other. Interestingly, click modules, such as Zeroaccess [39] and its variants, have been documented to distribute clickspam *only* after detecting some legitimate activity on a device. As well as this, another behaviour has been observed, which involves attackers adding random time offsets to blend in.

Overall, Clicktok addresses the following challenges:

(a) Resisting a mimicking source — an adversarial source that imitates a partially observed source, and
(b) Superimposed legitimate and clickspam time series.

3.2 Inference Algorithm

Clicktok uses a decomposition technique (Figure 4), which partitions click traces into legitimate clicks and clickspam. We make

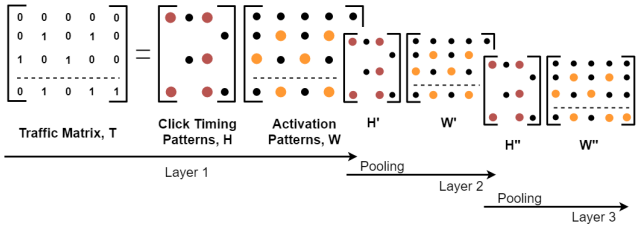

Figure 4: Multi-layer NMF

no assumptions about the shape or form of legitimate traffic, however we assume that click fraud is a minority fraction (< 50%) of the traffic traces. Partitioning leverages the observation that the shape and form of organic clickspam has a close dependency on the legitimate trace that was used to generate a specific clickspam attack. Partitioning is carried out using a multi-layer non-negative matrix factorization (NMF) algorithm.

Traffic matrix construction: Traffic traces from n source identifiers (e.g. source IP addresses or Android ID) are collected at a suitable vantage point such as an ad network or enterprise backbone, to construct a nxm traffic matrix of observations O. Each element O_{ij}, contains the number of clicks from source i during time-interval j.

The goal of the inference algorithm is to decompose the input clickstream into constituent r highly sparse timing patterns called basic-patterns. A basic pattern h_i can be represented as: $h_i = c+m,\ldots,c+k$ where c is the start time, and $0 < m < \cdots < k < \infty$ are offsets. Inference is achieved by the notion of compression. The idea is that the input traffic traces can be compressed down into r basic-patterns and weights. Specifically, the traffic matrix O is decomposed into H and W as:

$$O = HW \tag{1}$$

While using compression to trace stolen click traffic and their usage in click fraud campaigns has obvious potential, the main technical challenge is that click fraud campaigns utilise legitimate traffic as cover as shown in Figure 5. This creates interleaved (superimposed) click traffic which must be unmixed with minimal assumptions about baseline user or attacker behaviour. If this were not the case, a simple application of time-series correlation analysis [48] would reveal click fraud.

The inference algorithm illustrated in Figure 4 has two steps, first multiple layers of partitioning and second a pooling step. Both steps are motivated by Deep Neural Networks (DNN) [26] and are prior art.

Step1: Traffic partitioning First, nested layers of NMF algorithm [28] partition the traffic matrix O. NMF partitions the observed click matrix into sparse timing patterns (matrix H) and activation patterns (matrix W), i.e. $O = HW$. Sparsity is a key property here that incorporates the intuition of compressive partitioning within the decomposition step. NMF is an iterative technique with a multiplicative optimisation function at its core and a stopping critieria of $||O - HW|| \leq \epsilon$ where $||\odot||$ is the Frobenius norm. Nested decomposition layers further promote sparsity. Thus the output of the final decomposition layer is $O^K = H^1(H^2(H^3(\ldots.H^K W^K))))$, and the optimisation function works out to: $\min ||O^1 - H^1(H^2(H^3(\ldots(H^K W^K))\ldots)||$

Step2: Pooling. Second, to reduce sensitivity to synchronisation errors arising from timing misalignment, a moving window function is applied. In DNN literature [26], this is termed as a pooling function. Without this, time-synchronisation errors can cause redundant timing patterns that are slightly time-shifted, to appear within H. Pooling can provide some robustness against such errors. We incorporate pooling by including average-pooling [5] in each partitioning step. This is a moving-window function F whose input is the rows of the weight matrix W^k, and its output is the average over the input window $(j-c, j+c)$. $F(W^k_{i*}) = \overline{\{W^k_{ip} \mid j-c \le p \le j+c, 0 \le j \le N}$. To absorb alignment errors of up to an hour, we set $c = 12$ in our experiments.

Input: $O \in \mathbb{R}^{m \times n}$
The number of layers K
The number of columns to pool across, poolsize
Output: Partitioning at each layer k
for k *in* $1 : K-1$ **do**
  while $\epsilon > 0.05$ **do**
    $H^k = H^k \odot \frac{O^k (W^k)^T}{H^k W^k (W^k)^T}$
    $W^k = W^k \odot \frac{(H^k)^T O^k}{(H^k)^T H^k W^k}$
    $\epsilon = \sqrt{\sum_i^m \sum_j^n (O^k_{ij} - (H^k W^k)_{ij})^2}$
    $O^{k+1} \leftarrow F(W^k)$
  end
end

Algorithm 1: Traffic partitioning

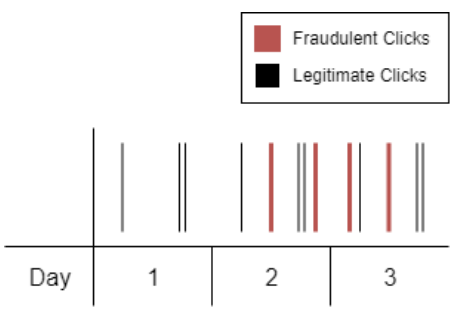

Figure 5: Overlapping legitimate and fraud clicks

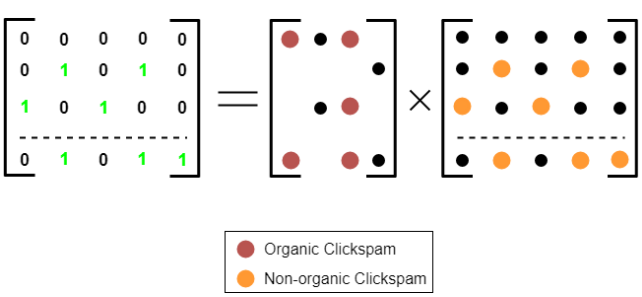

Figure 6: Clicktok traffic partitioning

3.2.1 Uniqueness of click partitioning. In order to detect clickspam, the partitioning process uses multiplicative rules to drive the optimisation function. We now show that the multiplicative update rules used in Algorithm 1 lead to a unique optimal solution.

In each layer of our algorithm, as per Equation 1, O is partitioned into H and W. The probability of observing t clicks given r timing-patterns $H_1, \ldots, H_r$ is given by $P(O_{ij} = t | H_1, \ldots, H_r) = \prod_{k=1}^{r} P(O_{ij} = t | H_k) P(H_k)$, where $P(H_k)$ is the prior probability about pattern H_k (which can be set using a honey pot or drawn from a uniform distribution). We'll ignore the layer number in the discussion that follows.

Assuming that legitimate clicks can occur at any point in time, on a per-source basis, the interclick arrival times between subsequent clicks justifies a Poisson distribution. The average number of legitimate clicks in any time interval can be well approximated by a normal distribution [2]. Making no assumptions about the distribution of clickspam, the probability of observing t legitimate clicks due to pattern H_{*k} in the ith user at the jth time interval is given by the Poisson probability distribution: $P(t, i, j, H_k) = \frac{\lambda^t_{i,j,k} \exp(-\lambda_{i,j,k})}{t!}$. After computing the log-likelihood, we obtain:

$$\ell = \sum_i^n \left(O_{i*} \ln(H^T W_{*i}) - H^T W_{*i} - \ln(O_{i*}!) \right) \tag{2}$$

The observations in matrix O are the number of ad-clicks (per time interval) from multiple simultaneously active (hence superimposed) basic-patterns. Considering all the patterns $H_{*1}, \ldots, H_{*r}$, let H_{kj} be the probability that pattern H_{*k} has been exploited to generate clicks in time interval j. The intensity of click fraud contribution of H_k to traffic from user i is given by W_{ik}. The corresponding vector W_{*k} encodes the commonality of pattern H_k across all users ($\sum_k W_{ik} = 1$). Therefore $O_{ij} = \lambda_{i,j} = \sum_k^r H_{ik} W_{kj}$. Replacing (HW) by λ in Equation 2 and computing derivatives with respect to λ, we obtain:

$$\ell = \sum_t t \ln\lambda - \lambda - \ln(t!) \tag{3}$$

$$-\frac{d\ell}{d\lambda} = \sum_t -\frac{t}{\lambda} + 1 \tag{4}$$

$$-\frac{d^2\ell}{d\lambda^2} = \sum_t \frac{t}{\sqrt{(\lambda)}} \tag{5}$$

From Equation 5, we see that $-\frac{d^2\ell}{d\lambda^2}$ is positive for all values of λ. Hence, our optimisation function is concave. This means that there are no local maxima which will adversely affect optimisation. The solution is unique regardless of the initial values of W and H.

3.2.2 Isolating clickspam. This stage of the application of NMF algorithm is a standard step, which may be familiar to the reader experienced in NMF. Here we reverse the partitioning process, in order to arrive at the traffic of interest i.e. clickspam.

We isolate organic clickspam, by observing the patterns (in matrix H) that repeat (as given by the activation matrix W). Each column of weight matrix W_{j*} gives the extent of repetition of the jth base pattern H_{j*}. All patterns that repeat twice or more, are involved in organic click fraud. Using matrix H, we compute H' as follows; we retain all $H'_{j*} = H_{j*} \forall \sum W_{j*} > 2$ and reset the rest $H'_{j*} = 0 \forall \sum W_{j*} \le 2$ (i.e. we retain all traffic that corresponds to click fraud). The weight

matrix W is also modified to retain weights corresponding to click fraud W', $W'_{j*} = W_{j*} \forall \sum W_{j*} > 2$ and reset the rest $W'_{j*} = 0 \forall \sum W_{j*} \leq 2$. We then compute $O' = W'H'$, activating click fraud patterns alone. Each cell of O' contains the number of fake clicks detected during any 5 minute interval.

Identification of non-organic click fraud is a two-step process. *First*, we cluster closely related pattern vectors using kNN (k nearest-neighbors) [45], a centroid-based clustering technique. The distance function between vectors is the inverse cosine similarity function. *Second*, to determine which clusters of patterns correspond to in-organic clickspam, we use entropy as the validation metric. We compute the average Shannon *entropy* of the distribution over inter-click times within click patterns in each cluster. The use of statistical average is appropriate as the patterns within a cluster are expected to be fairly similar to each other. Together, the weight and entropy of a cluster can be used to isolate different click fraud attacks. All clusters with entropy greater the 0.5 are indicative of legitimate traffic (which have innately higher entropy), the remaining clusters consists of low-rate clickspam generated using randomized or constant time offsets.

3.2.3 Algorithmic complexity. The complexity of our inference algorithm primarily depends on the number of time intervals and the density of clicks in each interval. There are $O(nmr)$ update operations in the worst (dense) case in the first iteration. We assume that the number of legitimate clicks is roughly $log(m)$, based on the sparseness of user click activity where m is the number of time intervals. r – the number of independent sources — is a small constant. Thus the complexity reduces to $O(n \, log \, m)$. The kNN algorithm used to cluster the basic-patterns involves a total of $nlog(n)$ comparisons, where k is a constant. Hence its complexity is $O(nlogn)$.

4 EVALUATION

In order to evaluate our inference algorithm, we require access to click traffic which contain both organic and non-organic clickspam. First, we acquired a pre-labelled dataset, consisting of both legitimate clicks and clickspam, in controlled proportions. To achieve this, we collected traffic within a university network, filtered it, and then exposed it to a testbed; consisting of malicious apps and click malware. As a result, the traffic exiting the testbed contains clicks from both sources of legitimate and fake clicks.

4.1 Dataset Acquisition

To collect **legitimate ad-click traces**, we setup traffic monitors on backbone routers of a university campus network. For each click, an advertisement is requested and the traffic monitors record the following information from the request: ad URL, ad server IP address, publisher page (referrer URL), source IP address, User-Agent string (UAstring) and the timestamp. Overall, we recorded a total of 217,334,190 unique clicks, between June 2015 and November 2017. The data was collected after due process of obtaining ethical approval, and all stored data is encrypted.

4.2 Click Malware and Exposure to Legitimate Traces

Our malware dataset consists of 12,518 binaries (allegedly) associated with clickfraud from a private collection. To verify that the malware dataset is indeed linked to clickfraud, we set up a dynamic analysis environment. The environment is suitably instrumented to study the dynamic behaviour of malware in response to legitimate clicks.

For Windows binaries, our analysis environment consists of Linux Mint 17.3 servers. Each server includes 4 x AMD Opteron 6376 Sixteen-Core 2.30GHz CPUs equipped with 1TB RAM and can run several VMs. Each VM consists of a Windows 8.1 guest OS installed with instrumentation tools. We used the Selenium IDE 3.1 [41], to inject click traffic into a Firefox 54.0 browser within the VM. Outgoing click traffic from the VM is captured using Virtualbox's network tracing facility. Dynamic analysis is carried out via the following workflow:

(1) Each windows binary is retrieved from the malware dataset and installed on a VM.
(2) A fraction of click sources are chosen from the legitimate traces.
(3) For each source (UAstring), a firefox browser is launched within a fresh VM.
(4) For each source, the webpage hosting the ad (referrer URL) is pushed into the browser and the corresponding (legitimate) ad-clicks are injected into the browser, whilst preserving inter-click times.
(5) The installed malware is thus exposed to legitimate click traffic. Any ad-clicks induced by malware are recorded on the VM and labeled as clickspam, on the basis that the ad-click was induced by malware.

For Android binaries, our analysis environment is based on the same hardware, however the Virtualbox VM is configured to use an Android 6.0 guest image. Our analysis-workflow schedules an app to run on the VM using the Monkeyrunner tool [18]. This tool allows us to inject ad-clicks from our dataset on to the installed app. We used a network monitoring tool to capture mobile clickspam locally on the device. It leverages Android's VPN API to redirect the device's network traffic through a localhost service enabling packet inspection. For Android apps, the workflow consists of the following:

(1) Each android app is retrieved from the dataset and pushed on to a fresh VM where it is executed for a chosen duration of time.
(2) As before, a fraction of click sources are chosen from legitimate traces and injected into the app using the Monkeyrunner tool.
(3) If the app is inclined towards click fraud, we expect to see additional clicks and ad-fetches over and above those injected. These are recorded by monitoring outgoing network traffic using Lumen, and labeled as clickspam.

Out of the 12,271 Windows malware that were initially associated with clickspam, we found that only 9,773 produced clickspam when put through our dynamic analysis environment. Similarly, out of the 247 apps we inspected, 93 produced clickspam. This means that our traffic dataset is based on a total of just over 10k click malware samples.

4.3 Passive Detection – Mimicry

In order to understand the significance of contextual parameters, we examine the traffic from *multiple ad networks*, control for the

effects of the *sizes* of legitimate and fake clicks, and *multiple ad categories*. Furthermore, to evaluate the performance of Clicktok as a passive defence, we are concerned with the false-positive (FPR) and true-positive (TPR) rates. The FP and TP rates are the fraction of legitimate clicks reported as fraudulent clicks, and the fraction of fraudulent clicks detected, respectively.

The first step is to create the traffic matrix, as explained in Section 3.2. The data from each source IP address is loaded into one row of the matrix, i.e. one row per source per day. The input input series is then divided into five minute intervals, thus establishing 288 columns per day. For each interval, we compute the total number of clicks; for which we do not know if the click is a legitimate or spam click.

Time interval (bin size): We must next consider the level of temporal granularity required in isolating fake clicks. Specifically, is the defender happy to know that a fake click occurred within a specific day, hour or five minute interval. Within the limitations of our study, we discovered that the average user clicks on less than 15 advertisements per day, and thus having a small bin size in the range of a few seconds would be excessive in terms of unduly high granularity.

Number of basic patterns: The number of unknown basic patterns, r, is fixed by hand, and thus we need to consider the maximum possible number of basic patterns, $r = n$. consideration is a scenario, where the traffic dataset is not compressible at all and therefore, a simple way to choose r is to simply set it to n. Subsequently, a higher value for r showed no resulting impact on detection efficiency, but simply causes some basic patterns to repeat and cluster together when k-nearest neighbours is applied. An r value greater than 73 did not result in any new basic patterns.

Attack volumes: For evaluating our algorithm, we must also consider the volumes of click fraud attacks. The hardest case for detection involves stealthy click fraud attacks, which our dataset contains, where the level of click fraud is less than 5 clicks per source device, per day (e.g. as induced by the TDSS and TDL-4 botnets [8]). As well as stealthy click fraud attacks, our dataset also consists of *sparse* click fraud attacks, which are mid-range attacks that correspond to between 5 and 15 clicks per day (e.g. Chameleon [24] and Zeus [3] descendants). Due to the minimal amount of clicks, any threshold-based statistical defence technique will find it difficult to detect a stealthy attack. Finally, we also consider *firehose attacks*, which involve attacks with volumes greater than 15 clicks per day, per source (e.g. ZeroAccess [39]). The attack traffic in this case, is distributed amongst publisher sites, to reduce the per-publisher volume below the anomaly threshold [31].

Isolating organic clickspam: When a subset of legitimate clicks is reused, even when click times are partially randomised or multiple legitimate clickstreams are combined, it triggers the optimization criteria within the inference algorithm. The organic click traffic used in the click fraud attack is found in matrix H, while the clickspam is identified by the location of pattern activation in matrix W (weight matrix). The precise mechanism is given in Section 3.2.2.

Isolating non-organic clickspam: For isolating non-organic clickspam, we first set the cosine-similarity threshold to 0.9 over non-repeating column vectors of the set of basic patterns H. We discovered several giant clusters (between 1 and 25) that contained a majority of the clicks from the clickstream (70% to 98%). As well as this, we identified smaller clusters (between 0 and 16) which contained fewer clicks (between 2% and 30% of the clicks). From these observations, we then examined the normalized entropy and weight of each cluster of patterns. As a heuristic, clusters of low entropy (≤ 0.5) correspond to simple click fraud attacks and pattern clusters with a normalised entropy greater than 0.5 are considered to be legitimate flows. The clusters of low entropy consist of attacks where the click modules are characterised by low-variability or near-constant inter-click times. This includes click modules that generate traffic via constant or random offsets to legitimate clicks.

ad network (duration)	Attack	#spam/src/day	% FPR	% TPR
Google (1 week)	stealth-1	1–4	0.066	62.80
	sparse-1	5–15	0.009	74.31
	firehose-1	>15	0.004	87.46
Google (12 weeks)	stealth-12	1–4	0.019	78.03
	sparse-12	5–14	0.006	81.33
	firehose-12	>15	0.004	99.32
adCentre (1 week)	stealth-1	1–4	0.071	67.05
	sparse-1	5–15	0.008	74.52
	firehose-1	>15	0.004	81.07
adCentre (12 weeks)	stealth-12	1–4	0.024	81.79
	sparse-12	5–15	0.005	82.92
	firehose-12	>15	0.003	98.36

Table 1: Passive detection — detection and error rates of inference

Detection and error rates of inference: The results of applying Clicktok are summarized in Table 1. We observed fairly serviceable detection rates, between 70% to 100%. More importantly, the false-positive rates are fairly low, with rates between 3 clicks to 66 clicks per hundred thousand clicks, for high-volume and stealthier low-volume attacks respectively. These results are consistent across ad networks, which aids with verifying the evaluation results of Clicktok in a realistic setting.

Size of background traffic: Upon evaluation of Clicktok, we observed that larger traffic sizes improve inference. For the Google ad network, the detection rate of stealthy attacks improved by 12%, whilst the FPR reduce from 66 to 19 clicks per hundred thousand. Interestingly, we also observed this with adSense (Microsoft), which increases confidence in the result. For stealthy attacks, this significant reduction is achieved by exploiting correlations across user clickstreams. As we consider the traffic for multiple users together, even a low attack rate can be detected across click traffic from multiple users. Understandably, we observed a lesser improvement with the firehose attacks, as attack rates already afford better detection rates and lower false-positive rates, even with just a few days of traffic.

Effects of click category: We examined click fraud in three categories: sponsored, contextual and mobile ads. Sponsored search ads are advertisements displayed by search engines, based on the keywords within a user's query alongside search results. Contextual ads are a more generic form of advertisements, which are displayed on a webpage based on the keywords present on that webpage. For instance, an advertisement on purchasing gold bars may be displayed on a webpage that contains information about investing in gold. Finally, mobile ads is a category of advertisements which are exclusively displayed on mobile devices. In all cases, the detection rates are fairly high, as well as a serviceably low false-positive rate. The detection of mobile ad click fraud has a slightly lower FPR compared to other categories.

(a) Google 84M

Fraud-type	#spam	% FPR	% TPR
Sponsored	16795	0.005%	93.595
	138345	0.005%	95.005
	1332910	0.004%	95.692
Contextual	22394	0.005%	87.806
	171883	0.005%	89.202
	1818777	0.004%	90.546
Mobile	18475	0.004%	91.379
	108999	0.003%	92.833
	1165221	0.003%	92.654

(b) Microsoft adSense 78M clicks

Fraud-type	#spam	% FPR	% TPR
Sponsored	18219	0.005%	89.11
	123442	0.004%	90.70
	912480	0.004%	92.17
Contextual	20380	0.006%	88.30
	323302	0.004%	91.68
	2198249	0.004%	90.93
Mobile	10594	0.005%	90.33
	141077	0.003%	91.52
	1161338	0.003%	94.76

Table 2: Detection and error rates of Inference across multiple clickstreams

ad network (duration)	Attack	#spam/src/day	% FPR	% TPR
Google (1 week)	stealth-1	1–4	0.051	66.40
	sparse-1	5–15	0.010	78.61
	firehose-1	>15	0.004	93.48
Google (12 weeks)	stealth-12	1–4	0.004	89.34
	sparse-12	5–15	0.004	91.62
	firehose-12	>15	0.003	96.77
Microsoft (1 week)	stealth-1	1–4	0.060	51.02
	sparse-1	5–15	0.003	75.14
	firehose-1	>15	0.005	92.60
Microsoft (12 weeks)	stealth-12	1–4	0.004	90.78
	sparse-12	5–15	0.003	92.44
	firehose-12	>15	0.002	95.41

Table 3: Active defence — detection and error rates of inference

4.4 Active Detection – Bait Clicks

The idea of the active defence is that the ad network injects *bait clicks* with a well defined inter-click delay pattern along with the ad it serves. Any mimicking of the injected pattern is evidence of click fraud. This approach is loosely motivated from traffic-analysis literature [42]. We used an injected pattern where consecutive injected clicks are a $\delta_n = \delta$ time apart $t_n = \delta_n + t_{n-1}$. The ad network can use more than one injected patterns each defined by a different δ in order to keep the bait-clicks discrete.

To implement this defence, as before the traffic matrix O is initialized with each row containing the time series from a 24 hour output of the testbed. We then decompose O using our inference algorithm and $r = n$, however there are two important changes. First, the initial r rows of H are set to the injected patterns instead of being initialized with random value. Second, instead of applying the update functions for H from Algorithm 1 for all $i < r'$, we fix the values of H_{i*} for all $i \leq r$, i.e. the rows of H representing injected patterns are not altered. This step allows the other rows of H to be suitably altered so as to represent legitimate timing patterns. The final step is the isolation of fraud clicks. This is done by analyzing W. W_{ij} gives the influence of pattern H_{j*} on click traffic time series O_{i*}. Since the injected timing patterns are located on the first r rows of H, the fraction of fraud clicks for time-window i of click traffic is simply computed as: $\frac{\sum_{j=1}^{r} W_{ij}}{\sum_{l=1}^{n} W_{il}}$.

In engineering terms, an ad network sends ads encapsulated with JavaScript code into the user's browser. Code execution is triggered by a suitable JavaScript event (such as when a page has finished loading). For the ad network the network and computational overheads are constant time per user and scales linearly with the number of users. The impact on the user device is also fairly minimal, dispatching three-four mouse clicks on an advertisement.

Results: Active defence improves detection rates by almost 10%. The results of active defences are documented in Table 3. In both Google and AdSense, active defences are very successful (> 89%) at detecting fake clicks at all ranges of attack traffic volumes from stealthy to a firehose, at low FPR of 30–40 per million clicks. The reduction in FPR for low-rate attacks is most improved compared to passive defences, indicating the importance of considering active attack approaches in fighting click fraud.

However, when active defences are presented with poor context (1-week traffic set), i.e. applied over only few clicks per user, we observe that detection and FPR are similar to passive defences. To understand why, we must note that looking for a response to injected traffic mainly detects mimicking attacks (as opposed to other variable-rate attacks such as randomly generated fake clicks). In our dataset, a week's traffic contains between 2 and 43 user clicks per day. At attack rates of two per day, Clicktok fails to detect the attack. As attack rates which are still fairly stealthily increases, click fraud attacks are readily detectable; For an increase in fake click rate from 1% to 10% of legitimate traffic, we observe a reduction in FPR by an order of magnitude for stealth attacks, while detection rate increases from 50% to 70%. For higher attack rates, the click fraud campaigns are mostly randomly generated fake clicks, and these result in modest changes in FPR.

5 DISCUSSION

Successful click fraud campaigns not only need to *scale*, they must also be *credible (pseudo-legitimate)*, and *stealthy*. Attackers achieve scale using compromised apps to automate click fraud campaigns and achieve stealth by using organic clicks to institute click fraud. To combat stealth, Clicktok detects click fraud based on timing characteristics of click traffic feeds received at ad networks. Unlike threshold based defences, it exploits correlations in timing behaviour across multiple compromised end-user devices.

Detecting click fraud using timing information exploits a fundamental property of attacks based on organic click fraud — some attack traffic is a function of historical legitimate traffic. Clicktok's strength lies in its generic approach. Instead of assuming specific attack methods for click fraud attack strategies, Clicktok uses a notion of a *compressive* optimization function to isolate clicks generated by different sources without prior knowledge about what those sources are. Thus Clicktok can isolate non-organic clickspam such as those generated at pseudo-random times and the replay of organic clickstreams.

Our techniques show some promise: passive defences have a detection rate of around 78%–81% with a false-positive rate of 190 per million clicks; and, active attacks work best, with the false positive rate entering a region of 40 to 70 false positives per million clicks. This means the ad network charges the advertiser for 40 to 70 clicks per million clicks received, as compared with a false-positive rate of between 200,000–300,000 clicks per million in current ad networks [11, 17].

Passive defences: When Clicktok is applied as a passive defence, it acts as a (lossy) compression function that partitions the input time series into basis patterns, such that all basis patterns from the same

distribution are grouped together. Thus given a clickstream containing clicks generated from different statistical distributions (random variables) over generation times, Clicktok creates as many partitions.

Active defences: We model a clickstream as a timing channel between a user and the ad network composed of inter-click times between consecutive ad clicks. The ad network watermarks the channel periodically using bait-clicks. When an attacker harvests and reuses a (legitimate) clickstream, it sets off watermark detectors located in the ad network. Unlike conventional watermarks however, the timing pattern induced by click fraud may not appear as an isolated series of consecutive clicks. Clickstreams arriving at the ad network may therefore contain clickspam that's thoroughly mixed (superimposed) with legitimate clicks. Thus, the challenge of designing the watermark detector system is to *unmix* the stream back into legitimate and clickspam. Clicktok develops the algorithmic basis for carrying out this work. In order to bypass the active defences, the attacker must distinguish between the bait-click watermark and legitimate traffic, which roughly speaking, shifts the burden of solving the click fraud problem to the attacker.

5.1 Limitations of Clicktok

Metrics used: We use entropy as a validation metric with a normalised threshold of 0.5. However, this may be vulnerable to the adversary generating non-organic clickspam using pseudo-random number generator that generates clicks from distributions with high entropy. While we haven't observed this taking place in our testbed across 1.5 years, it is a reasonable countermeasure for the attacker. Clicktok can still function in these circumstances by leveraging a honeypot to populate the pattern matrix H as follows: columns of the pattern matrix are pre-populated with traffic from the honeypot and fixed, i.e. excluded from the application of multiplicative rules in Algorithm 1 while the rest of the columns are optimised. Upon convergence, the weight matrix will contain high weights if any traffic similar to the honeypot is observed in the traffic matrix O.

IP aggregation and churn: Often, enterprise networks may deploy DMZ or other traffic aggregators to avoid exposing IP addresses to the outside world. This impacts the extent of attribution. Malice will therefore be at best traced back to the aggregator and further investigation would be required to isolate the actual source behind the aggregator. Churn causes similar record-keeping problems. In order to positively attribute malice to a source, the source-IP addresses involved must be reconciled with the local DHCP records. We note that the impact on detection itself is minimal; an short DHCP expiration policy, simply implicates both the previous and new source IP addresses of a malicious source.

Cookies and deletion: A relatively reliable approach is to use cookies instead of source-IP addresses to keep track of malice. ad networks can track clickstreams on a per user basis using authenticated HTTP sessions, instead of solely depending on source IP addresses. Cookies can help address churn issues where a pool of source IP addresses are cycled among many users.

5.2 Comparison with related work

We compared Clicktok's efficiency with click fraud detection techniques proposed in the literature. While some of these techniques were not intended to work with the challenges of embedded fake clicks or stealthy click fraud attacks, we can nonetheless compare against them on real datasets to get a sense of how Clicktok compares against these approaches. Several techniques to differentiate fake clicks from legitimate clicks have been proposed. Work in this space can be categorized into *threshold-based* techniques that build a baseline of benign behaviour and analyze deviations from the baseline, and on timing analysis techniques. We choose two methods from timing analysis and a recently proposed threshold technique for detection efficiency.

Technique	Attack	#spam/src/day	% FPR	% TPR
Clicktok	stealth	1–4	0.066	62.80
	sparse	5–15	0.01	74.31
	firehose	>15	0.004	87.46
Similarity Seeker	stealth	1–4	14.41	57.49
	sparse	5–15	9.68	59.82
	firehose	>15	0.78	85.21
ViceROI	stealth	1–4	10.23	60.03
	sparse	5–15	2.65	66.13
	firehose	>15	0.5	78.29
PubCrawl	stealth	1–4	4.70	52.64
	sparse	5–15	3.24	67.28
	firehose	>15	0.85	77.91

Table 4: Comparative analysis of Clicktok (Passive) vs others

- Threshold-based approaches detect hotspots of activity between click-malware and publishers. One set of techniques detect traffic hotspots [31–33]. Another technique is to examine publisher-user pairs with above-average click rates [12]. All the techniques in this approach develop a normative baseline of activity and detect malicious behaviour beyond a threshold distance from the baseline. The idea is that fraudsters need to scale their activity to a level where their turnover (from a click fraud campaign) covers their costs as well as generate a profit.
- In time-series analysis, techniques such as auto-correlation, partial-correlation, and cross-correlation techniques [48] are used to process the input clickstream. Each clickstream is converted into a time-series by counting the number of clicks within each time interval, just as in Clicktok. Parsing Clicktok's traffic matrix in a row-major fashion gives the required time-series vector over which analysis techniques are applied. Recently, PubCrawl [19] extended this idea by combining it with learning approaches, where a classifier is trained with a labelled dataset containing time series for both honest and fraud clicksets.

The algorithmic complexity of both is similar to Clicktok so we were able to readily run them on the 1-week Google ad network dataset. Results from our comparison are shown in Table 4. We found that the performance of Clicktok is similar to existing solutions for high-rate (hose and firehose) attacks in terms of detection rate, but has a much better FPR for all attacks. For low-rate attacks (stealth and sparse), Clicktok significantly outperforms all existing solutions. For example, for a hose attack, Clicktok's FPR is 0.04% whereas FPR for other approaches ranged from 0.5%—7.65%. As another example, for a stealth attack, Clicktok's FPR is 0.066% whereas FPR for other approaches ranged from 4.7%–14.7%.

Our experiments show the limitations of using threshold based approaches, which can be defeated by reducing the network loads

placed by attackers. In the case of ViceROI, the cost of renting a botnet has fallen by three orders of magnitude which allows attackers to scale up the number of attack hosts for the same amount of click fraud, pushing the level of fraud per user and per publisher below the detection threshold. Similarly, with Similarity-Seeker which looks for traffic hotspots, but without using bluff ads to generate a baseline, the hotspot is diffused with increase in the number of attack hosts and malicious publishers. Interestingly, PubCrawl was fairly successful at detecting stealthier attacks than Similarity Seeker and ViceROI. PubCrawl also uses time-series analysis, however it is not designed to handle the case where clickspam and legitimate traffic are temporally overlapped. This capability gap is especially evident when the fake clickstream size is smaller as any overlap tends to severely 'damage' the signal that PubCrawl detects.

6 RELATED WORK

Click fraud demonstrates a serious economic impact on the Internet sub-economy of the click marketplace, bringing rise to a growing body of academic research to address the problem. Click fraud inflicts losses for tens of thousands of online advertisers, causing upwards of hundreds of millions of dollars each year [37].

Several papers have highlighted the importance of the field of advertising and click fraud [44, 50]. Keeping ad networks free of fraud is highlighted by the work of Mungamuru et al. [37], who show that ad networks free of click fraud results in a competitive advantage over rival ad networks and thus attracts more advertisers. Research shows that even the largest advertising platforms are affected by click fraud [25] and are tackling the problem, by primarily employing data mining techniques to distinguish legitimate, fraudulent or bot-generated click events. Clickbots are a leading attack vector for carrying out click fraud; around 30% of the clicks are fraudulent across major ad networks [11, 17] and originate from malware networks (rent-a-botnet services).

Clickbots and Malware Networks: Malware networks are primarily operated by malicious publishers or a malicious advertiser who depleted the budget of competing advertisers such as the WOW botnet [46]. Chen et al. [8] discovered that the TDSS/TDL4 botnet — one of the most sophisticated botnets — incurred an average of $340 thousand in daily losses to advertisers. The losses scale up to ten times more than the daily impact some previous botnets had to the advertising ecosystem [30, 40]. Naturally, a number of studies have focused on botnets used for click fraud. Daswani et al. [10] reverse engineered clickbot and followed by Miller et al. [34] on Fiesta and 7cy. These works focused on the C&C traffic structure of the attack infrastructure. We consider the timing characteristics of click-generation algorithms used by botnets instead of the structure of the C&C traffic or the malware binaries. Unlike these specialized studies which are solely focused on specific botnets, our work has wider application including future botnets.

Online advertisement networks employ a variety of heuristics to detect click fraud and apply corresponding discounts on advertisers they invoice [9]. These heuristics involves tracking and bounding user click behaviour often on a per-IP-address basis. In response, criminals are using large distributed attack networks of bots for launching replay click fraud attacks. Bots are unique in that they collectively maintain a communication structure across infected machines to resiliently distribute commands from a *command and control* node. The ability to coordinate and upload click fraud attack commands gives the botnet owner vastly increased power against major ad networks.

Traffic analysis: Several countermeasures [31–33] are based on traffic analysis. These methods isolate machines being used for click fraud attacks by identifying traffic hot-spots between the attack machines and publisher sites. Not only non-stationary traffic, but also stealth occurrences of malicious behaviors both introduce issues to analyse anomalous network traffic [22]. Machine learning frameworks have been used to help analyse network behaviour, identify click fraud and adapt to changes in traffic [22, 36]. However, most of of this research is done on botnet-induced click traffic, and does not identify the effectiveness of these methods on other forms of fraudulent clicks.

Detecting rogue publishers: In [12], Dave et al. point out that the conversion rate of malicious publishers is abnormally high. Thus a heuristic based on the conversion rate could be used as part of a click fraud defence. However, this is easily circumvented by distributing attack traces across a few publisher accounts. Or, alternately after an account is "tainted", it is abandoned. This strategy imposes additional effort on the attackers (they need to register new publisher accounts with high frequency) but this is cheap. Such a strategy is already well in use to evade reputation blacklists for domain names [15] due to which the mean life-time of domains is a mere 2.5 hours.

Click honeypots: Haddadi proposed the use of Bluff ads [17] – a fake ad that is of little interest to the legitimate user but would attract click fraud attacks in the wild. Bluff ads can be used to collect attack traffic and in conjunction with Click fraud to extract basis patterns corresponding to the static timing characteristics of malware click-modules. Instead of bait-ads, Clicktok applies bait clicks to entice click fraud apps into responding with watermarked click fraud. This is a new approach towards instrumentation of click fraud apps at scale, and will form part of our future work.

Human bots: A human-centric approach to setting up a call-centre to generated handcrafted clickspam [27] and tricking people into clicking ads on porn websites and mobile gaming apps [13]. Zhang et al. studied click fraud by purchasing click traffic feeds. They also showed that the explicit existence of human clickers [51]. Our work is focused on clickspam generation from malware alone.

Trustworthy clicks: Gummadi et al. [16] propose to combat bot activity through detection mechanisms at the client. The client machine has a trusted component that monitors keyboard and mouse input to attest to the legitimacy of individual requests to remote parties. In addition, Juels et al. [20] likewise propose dealing with click fraud by certifying some clicks as premium or legitimate using an attester instead of attempting to filter fraudulent clicks. Zingirian and Benini [52] show that a single adversary can increase the click count for given advertised subscribers, from a single IP address, even if the paid clicks are not linked with legitimate advertisements. They evaluate security tradeoffs and revenue losses from discarded clicks claimed as being illegitimate, and formulate an algorithm to control the tradeoff; which shows to induce a very low impact even with a large volume of clicks. Kintana et al. [23] created a system designed to penetrate click fraud filters in order to discover detection vulnerabilities. Blundo et al. [4] propose to detect click fraud by using Captchas

and proof-of-work schemes. In a proof-of-work based scheme [11], the ad network serves Captchas probabilistically in response to ad-clicks to verify the authenticity of clicks. The main problem here is that malware often delegate the problem of solving a CAPTCHAS to a third party. For example, via a gaming app where the players are asked to periodically solve Captchas for "authentication purposes".

Detecting fake search engine queries: A number of click fraud malware use cover traffic in the form of automated search engine queries leading to the malicious publisher's website. Researchers have dedicated considerable effort to methods for differentiating search queries from automated and human sources. Kang et al. [21] propose a learning-based approach to identify automated searches but exclude query timing. Yu et al. [49] observe details of bot behaviour in aggregate, using the characteristics of the queries to identify bots. Buehrer et al. [6] focus on bot-generated traffic and click-through designed to influence page-rank. These efforts do not examine timing behaviour, focusing instead on techniques for the search engine to identify automated traffic. Shakiba et al.[43] propose the detection of spam search engine queries using a semi-supervised stream clustering method, which characterises legitimate and illegitimate users using linguistic properties of search queries and behavioural characteristics of users; shown to be accurate 94% of the time with low overhead. Graepel et al. [14] define a scalable Bayesian click-through rate prediction algorithm, mapping input features to probabilities, for Sponsored Search in the Bing search engine. They showed that their new algorithm was superior to, and outperformed a calibrated Naive Bayes algorithm — regardless of the new algorithm being calibrated.

Machine-learning based defences are vulnerable to mimicry attacks: Many machine-learning based approaches [44] have been proposed for click fraud detection. Many of these rely on rich feature sets. While there are clear benefits to a diverse feature set we also know that some features are highly predictable. For instance, accurately predicting ads of interest to a user is possible [14], reducing the utility of using features such as keywords in detecting click fraud — malware can target advertisers relevant to the user. Our finding is that timing information is relatively hard to predict and should be used in conjunction with other features to combat click fraud.

Conventional time-series analysis techniques don't work: Conventional time-series techniques are auto-correlation, partial-correlation, and cross-correlation techniques [48] which can find statistically similar subsets across two or more time series. Correlation tools are typically applied to wavelet coefficients or to the inverse transform of wavelet coefficients at carefully selected coefficient levels. Correlation-based techniques can detect sub-similar features if the time series signal is not contaminated by convolutions of the signal with other signals or high-amplitude noise. Unfortunately, in the case of click fraud detection the traffic is contaminated by time-overlapping legitimate and spam clicks as shown in Figure 5. Indeed as prior art has noted, compromised-apps or click-malware generate clickspam only when some "trigger" fires such as the presence of legitimate clicks on the compromised device [35].

Time series analysis has been used previously for detecting click fraud. In PubCrawl [19], the main idea is that users tend to be a lot more noisier than crawlers who have a relatively stable behaviour pattern across time. The challenge we address is significantly harder: what happens when machines (crawlers) mimic user-behaviour by closely following past user behaviour rather than the random strategies (which PubCrawl is designed to isolate).

7 CONCLUSION

Online advertising is a funding model used by millions of websites and mobile apps. Criminals are increasingly targeting online advertising with special purpose attack tools called *click malware*. Click fraud instituted via malware is an important security challenge. Static attacks involving large attack volumes are easily detected by state-of-the-art techniques. However, dynamic attacks involving stealthy clickspam, that adapt to the behaviour of the device user, are poorly detected by current methods. We found that timing analysis can play a compelling role in isolating click fraud, instituted via both static and dynamic attack techniques. We applied NMF, a technique that identifies clickspam by exploiting the relative uncertainty between clickspam and legitimate clickstreams. It achieves this by detecting repetitive patterns that arise within ad network clickstreams from organic clickspam. We analysed a corpus of malware within an instrumented environment, that enabled us to control the generation of clickspam by exposing malware to legitimate clickstreams. We tested a passive technique which shows some promise. We also evaluated an active defense, where we injected watermarked click traffic into the analysis environment, that works better still. While timing analysis is well studied within the field of information hiding, for its ability to unearth hidden communication, its potential has yet to be fully explored in understanding stealthy click fraud attacks. Our work indicates that timing analysis might indeed be relevant to building better click fraud detection.

REFERENCES

[1] 2016. You can now rent a Mirai botnet of 400000 bots. https://www.bleepingcomputer.com/news/security/you-can-now-rent-a-mirai-botnet-of-400-000-bots/

[2] Patrick Billingsley. 1995. *Probability and Measure* (3 ed.). Wiley-Interscience. http://www.worldcat.org/isbn/0471007102

[3] Hamad Binsalleeh, Thomas Ormerod, Amine Boukhtouta, Prosenjit Sinha, Amr Youssef, Mourad Debbabi, and Lingyu Wang. 2010. On the analysis of the zeus botnet crimeware toolkit. In *2010 Eighth International Conference on Privacy, Security and Trust*. IEEE, 31–38.

[4] Carlo Blundo and Stelvio Cimato. 2002. SAWM: a tool for secure and authenticated web metering. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering (SEKE '02)*. ACM, New York, NY, USA, 641–648. https://doi.org/10.1145/568760.568871

[5] Y-Lan Boureau, Jean Ponce, and Yann Lecun. 2010. A Theoretical Analysis of Feature Pooling in Visual Recognition. In *27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, HAIFA, ISRAEL*.

[6] Gregory Buehrer, Jack W. Stokes, and Kumar Chellapilla. 2008. A large-scale study of automated web search traffic.. In *AIRWeb (ACM International Conference Proceeding Series)*, Carlos Castillo, Kumar Chellapilla, and Dennis Fetterly (Eds.). 1–8.

[7] Neha Chachra, Stefan Savage, and Geoffrey M. Voelker. 2015. Affiliate Crookies: Characterizing Affiliate Marketing Abuse. In *Proceedings of the 2015 Internet Measurement Conference (IMC '15)*. ACM, New York, NY, USA, 41–47. https://doi.org/10.1145/2815675.2815720

[8] Yizheng Chen, Panagiotis Kintis, Manos Antonakakis, Yacin Nadji, David Dagon, and Michael Farrell. 2017. Measuring lower bounds of the financial abuse to online advertisers: A four year case study of the TDSS/TDL4 Botnet. *Computers & Security* 67 (2017), 164–180.

[9] Click-spam accounting [n. d.]. The lane's gift v. google report. http://googleblog.blogspot.in/pdf/Tuzhilin_Report.pdf.

[10] Neil Daswani and Michael Stoppelman. 2007. The Anatomy of Clickbot.A. In *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets (HotBots'07)*. USENIX Association, Berkeley, CA, USA, 11–11. http://dl.acm.org/citation.cfm?id=1323128.1323139

[11] Vacha Dave, Saikat Guha, and Yin Zhang. 2012. Measuring and fingerprinting click-spam in ad networks. In *Proceedings of the ACM SIGCOMM 2012*

conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM '12). ACM, New York, NY, USA, 175–186. https://doi.org/10.1145/2342356.2342394

[12] Vacha Dave, Saikat Guha, and Yin Zhang. 2013. ViceROI: Catching Click-spam in Search Ad Networks. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*. ACM, New York, NY, USA, 765–776. https://doi.org/10.1145/2508859.2516688

[13] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. 2011. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 3–14.

[14] Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-Scale Bayesian Click-Through rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Johannes Fürnkranz and Thorsten Joachims (Eds.). Omnipress, Haifa, Israel, 13–20. http://www.icml2010.org/papers/901.pdf

[15] Chris Grier, Lucas Ballard, Juan Caballero, Neha Chachra, Christian J. Dietrich, Kirill Levchenko, Panayiotis Mavrommatis, Damon McCoy, Antonio Nappa, Andreas Pitsillidis, Niels Provos, M. Zubair Rafique, Moheeb Abu Rajab, Christian Rossow, Kurt Thomas, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. 2012. Manufacturing Compromise: The Emergence of Exploit-as-a-Service . In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*.

[16] Ramakrishna Gummadi, Hari Balakrishnan, Petros Maniatis, and Sylvia Ratnasamy. 2009. Not-a-Bot (NAB): Improving Service Availability in the Face of Botnet Attacks. In *NSDI 2009*. Boston, MA.

[17] Hamed Haddadi. 2010. Fighting online click-fraud using bluff ads. *SIGCOMM Comput. Commun. Rev.* 40, 2 (April 2010), 21–25. https://doi.org/10.1145/1764873.1764877

[18] Google Inc. Accessed Mar 2018. Monkeyrunner reference. https://developer.android.com/studio/test/monkeyrunner

[19] Gregoire Jacob, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. 2012. PUBCRAWL: Protecting Users and Businesses from CRAWLers. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. USENIX, Bellevue, WA, 507–522. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/jacob

[20] Ari Juels, Sid Stamm, and Markus Jakobsson. 2007. Combating click fraud via premium clicks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium (SS'07)*. USENIX Association, Berkeley, CA, USA, Article 2, 10 pages. http://dl.acm.org/citation.cfm?id=1362903.1362905

[21] Hongwen Kang, Kuansan Wang, David Soukal, Fritz Behr, and Zijian Zheng. 2010. Large-scale Bot Detection for Search Engines. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 501–510. https://doi.org/10.1145/1772690.1772742

[22] Sara Khanchi, Nur Zincir-Heywood, and Malcolm Heywood. 2018. Streaming Botnet traffic analysis using bio-inspired active learning. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–6.

[23] Carmelo Kintana, David Turner, Jia-Yu Pan, Ahmed Metwally, Neil Daswani, Erika Chin, and Andrew Bortz. 2009. The Goals and Challenges of Click Fraud Penetration Testing Systems. In *International Symposium on Software Reliability Engineering*.

[24] G Kirubavathi and R Anitha. 2014. Botnets: A study and analysis. In *Computational Intelligence, Cyber Security and Computational Models*. Springer, 203–214.

[25] Brendan Kitts, Jing Ying Zhang, Gang Wu, Wesley Brandi, Julien Beasley, Kieran Morrill, John Ettedgui, Sid Siddhartha, Hong Yuan, Feng Gao, et al. 2015. Click fraud detection: adversarial pattern recognition over 5 years at Microsoft. In *Real World Data Mining Applications*. Springer, 181–201.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90. https://doi.org/10.1145/3065386

[27] Nir Kshetri. 2010. The Economics of Click Fraud. *IEEE Security & Privacy* 8, 3 (2010), 45–53. http://dblp.uni-trier.de/db/journals/ieeesp/ieeesp8.html#Kshetri10

[28] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *In NIPS*. MIT Press, 556–562.

[29] Bin Liu, Suman Nath, Ramesh Govindan, and Jie Liu. 2014. DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. USENIX Association, Seattle, WA, 57–70. https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/liu_bin

[30] Wei Meng, Ruian Duan, and Wenke Lee. 2013. DNS Changer remediation study. *Talk at M3AAWG 27th* (2013).

[31] Ahmed Metwally, Divyakant Agrawal, Amr El Abbad, and Qi Zheng. 2007. On Hit Inflation Techniques and Detection in Streams of Web Advertising Networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*. IEEE Computer Society, Washington, DC, USA, 52–. https://doi.org/10.1109/ICDCS.2007.124

[32] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2007. Detectives: detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)*. ACM, New York, NY, USA, 241–250. https://doi.org/10.1145/1242572.1242606

[33] Ahmed Metwally, Fatih Emekçi, Divyakant Agrawal, and Amr El Abbadi. 2008. SLEUTH: Single-pubLisher attack dEtection Using correlaTion Hunting. *Proc. VLDB Endow.* 1, 2 (Aug. 2008), 1217–1228. http://dl.acm.org/citation.cfm?id=1454159.1454161

[34] Brad Miller, Paul Pearce, Chris Grier, Christian Kreibich, and Vern Paxson. 2011. What's Clicking What? Techniques and Innovations of Today's Clickbots. In *Proceedings of the 8th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'11)*. Springer-Verlag, Berlin, Heidelberg, 164–183. http://dl.acm.org/citation.cfm?id=2026647.2026661

[35] Andreas Moser, Christopher Kruegel, and Engin Kirda. 2007. Exploring Multiple Execution Paths for Malware Analysis. In *Proc. of the IEEE Symposium on Security and Privacy*.

[36] Riwa Mouawi, Mariette Awad, Ali Chehab, Imad H El Hajj, and Ayman Kayssi. 2018. Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing. In *2018 International Conference on Innovations in Information Technology (IIT)*. IEEE, 88–92.

[37] Bob Mungamuru and Stephen Weis. 2008. In *Financial Cryptography and Data Security*, Gene Tsudik (Ed.). Springer-Verlag, Berlin, Heidelberg, Chapter Competition and Fraud in Online Advertising Markets, 187–191. https://doi.org/10.1007/978-3-540-85230-8_16

[38] G. Ollmann. 2009. Want to rent an 80-120k DDoS Botnet? Blog: Damballa. http://bit.ly/W9Hh2x

[39] Paul Pearce, Vacha Dave, Chris Grier, Kirill Levchenko, Saikat Guha, Damon McCoy, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. 2014. Characterizing Large-Scale Click Fraud in ZeroAccess. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, USA, 141–152. https://doi.org/10.1145/2660267.2660369

[40] Paul Pearce, Vacha Dave, Chris Grier, Kirill Levchenko, Saikat Guha, Damon McCoy, Vern Paxson, Stefan Savage, and Geoffrey M Voelker. 2014. Characterizing large-scale click fraud in zeroaccess. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 141–152.

[41] The Selenium Project. Accessed Oct 2017. Selenium IDE. https://docs.seleniumhq.org

[42] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. 1998. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications* 16, 4 (1998). citeseer.ist.psu.edu/reed98anonymous.html

[43] Tahere Shakiba, Sajjad Zarifzadeh, and Vali Derhami. 2018. Spam query detection using stream clustering. *World Wide Web* 21, 2 (2018), 557–572.

[44] Brett Stone-Gross, Ryan Stevens, Apostolis Zarras, Richard Kemmerer, Chris Kruegel, and Giovanni Vigna. 2011. Understanding Fraudulent Activities in Online Ad Exchanges. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC '11)*. ACM, New York, NY, USA, 279–294. https://doi.org/10.1145/2068816.2068843

[45] Thanh N. Tran, Ron Wehrens, and Lutgarde M.C. Buydens. 2006. KNN-kernel density-based clustering for high-dimensional multivariate data. *Computational Statistics & Data Analysis* 51, 2 (2006), 513 – 525. https://doi.org/10.1016/j.csda.2005.10.001

[46] Western Division of Washington at Seattle United States District Court. June 2009. United States District Court: Microsoft vs Eric Lam et. al., Civil Case Number CO 9-0815. http://graphics8.nytimes.com/packages/pdf/business/LamComplaint.pdf

[47] Jialu Wei. 2016. DDoS on internet of thingsâĂŤa big alarm for the future.

[48] William Wu-Shyong Wei. 1994. *Time series analysis*. Addison-Wesley publ.

[49] Fang Yu, Yinglian Xie, and Qifa Ke. 2010. SBotMiner: Large Scale Search Bot Detection. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM '10)*. ACM, New York, NY, USA, 421–430. https://doi.org/10.1145/1718487.1718540

[50] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. 2014. The Dark Alleys of Madison Avenue: Understanding Malicious Advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)*. ACM, New York, NY, USA, 373–380. https://doi.org/10.1145/2663716.2663719

[51] Qing Zhang, Thomas Ristenpart, Stefan Savage, and Geoffrey M. Voelker. 2011. Got Traffic?: An Evaluation of Click Traffic Providers. In *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality (WebQuality '11)*. ACM, New York, NY, USA, 19–26. https://doi.org/10.1145/1964114.1964119

[52] Nicola Zingirian and Michele Benini. 2018. Click Spam Prevention Model for On-Line Advertisement. *CoRR* abs/1802.02480 (2018). arXiv:1802.02480 http://arxiv.org/abs/1802.02480