What You See is NOT What You Get: Discovering and Tracking Social Engineering Attack Campaigns

Phani Vadrevu
University of New Orleans
New Orleans, Louisiana, USA
phani@cs.uno.edu

Roberto Perdisci
University of Georgia
Georgia Institute of Technology
Georgia, USA
perdisci@cs.uga.edu

ABSTRACT

Malicious ads often use *social engineering* (SE) tactics to coax users into downloading unwanted software, purchasing fake products or services, or giving up valuable personal information. These ads are often served by low-tier ad networks that may not have the technical means (or simply the will) to patrol the ad content they serve to curtail abuse.

In this paper, we propose a system for large-scale automatic discovery and tracking of *SE Attack Campaigns delivered via Malicious Advertisements* (SEACMA). Our system aims to be generic, allowing us to study the SEACMA ad distribution problem without being biased towards specific categories of ad-publishing websites or SE attacks. Starting with a seed of low-tier ad networks, we measure which of these networks are the most likely to distribute malicious ads and propose a mechanism to discover new ad networks that are also leveraged to support the distribution of SEACMA campaigns.

The results of our study aim to be useful in a number of ways. For instance, we show that SEACMA ads use a number of tactics to successfully evade URL blacklists and ad blockers. By tracking SEACMA campaigns, our system provides a mechanism to more proactively detect and block such evasive ads. Therefore, our results provide valuable information that could be used to improve defense systems against social engineering attacks and malicious ads in general.

CCS CONCEPTS

• **Information systems → Online advertising**; • **Security and privacy → Social engineering attacks**; **Web protocol security**; *Malware and its mitigation.*

KEYWORDS

Social Engineering, malicious advertisements, web security

ACM Reference Format:
Phani Vadrevu and Roberto Perdisci. 2019. What You See is NOT What You Get: Discovering and Tracking Social Engineering Attack Campaigns. In *Internet Measurement Conference (IMC '19), October 21–23, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3355369.3355600

IMC '19, October 21–23, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6948-0/19/10…$15.00
https://doi.org/10.1145/3355369.3355600

1 INTRODUCTION

The advent of the Internet has caused enormous changes to the way products are advertised and has created a new multi-billion dollar industry. Today, online ads are primarily distributed via large ad networks, the most popular of which are controlled by a few large Internet companies such as Google, Facebook, Twitter, etc. However, the online advertising ecosystem also includes a myriad of lower-tier ad networks, some of which are used by ad-publishing websites as higher-revenue alternatives with weaker content restriction policies compared to more reputable ad networks.

Unfortunately, cyber-criminals have learned to take advantage of online ads to target and exploit innocent victims. Malicious ads often use *social engineering* (SE) tactics to coax users into downloading unwanted software, purchasing fake products or services, or giving up valuable personal information. These ads are often served by unpopular, low-tier ad networks that may not have the technical means (or simply the will) to patrol the ad content they serve to curtail abuse. For instance, recent studies have analyzed malicious ads distributed on specific categories of publisher sites, such as free video streaming websites [33], or dedicated to specific attack vectors, such as tech support scams [30] or survey scams [24]. Others have studied how malicious ads are delivered or how they can be detected using statistical features [28, 40] (related work is discussed in more details in Section 7). However, we are not aware of studies that specifically look at identifying and tracking *Social Engineering Attack Campaigns delivered via Malicious Ads* (SEACMA) independently from the publishing sites or specific attack vectors, including leveraging the SE visual components to identify SE attack campaigns and characterize attack trends.

In this paper, we propose a system for large-scale automatic discovery and tracking of SEACMA campaigns. Our system aims to be generic and to study the SEACMA ad distribution problem without being biased towards studying only specific categories of ad-publishing websites or SE attacks. Using our system, we find numerous SE ad campaigns that lead to concrete SE attacks. As we will discuss later, the type of SE attacks we discover in our measurements mostly belong to known attack categories (e.g., fake software, technical support scams, survey scams, etc.). It is important to notice, however, that our main goal is not to uncover new categories of SE attacks, but rather to discover and track the SEACMA campaigns that lead to generic SE attacks and to study the mechanisms used to deliver such attacks.

Figure 2 presents an overview of our system, which we describe in more details in Section 3. We start with a seed list of low-tier ad networks that we manually compiled by scouting websites and forums that discuss techniques for increasing ad revenue. Using an existing web service, we "reverse" this initial list of ad networks into a large list of websites that publish ads served from those networks. Then, we leverage a custom built scalable crawler farm to visit these websites and log detailed information about candidate SE attack pages reached via ads published on those sites. As a next step, we cluster similar candidate SE attacks into SEACMA campaigns based on their visual components and filter out possible benign content. We then input the remaining SEACMA campaigns into a "milking" process, in which we track the ad campaigns through time. For instance, we track how frequently changing domains are used to evade URL blacklists, track the visual components of the campaigns through time and collect the highly polymorphic unwanted software distributed by some of the campaigns. Finally, we analyze each of the observed SE ad campaigns to attribute them back to the originating ad networks. This allows us to determine which ad networks tend to serve the largest fraction of malicious ads and to discover previously unknown ad networks involved in the propagation of SEACMA campaigns. These new ad networks could then be fed back into our system to further expand our visibility into SEACMA campaigns.

The results of our study aim to be useful in a number of ways. For instance, even though in 2016 Google announced that its Google Safe Browsing URL blacklists would start protecting users from social engineering ads [8], we show that malicious advertisers use a number of tactics to successfully evade them. By tracking SEACMA campaigns, our system provides a mechanism to more proactively detect and block such evasive SE attacks. In addition, our system captures detailed browsing logs related to visiting malicious pages reached through SEACMA-related malicious ads, including a screenshot of the social engineering attack vectors. Because SE attacks have a critical visual component, collecting a record of how such attacks are reached by users and rendered in the browser may allow one to faithfully reproduce real-world SE attacks in a controlled environment, for example as a way to improve users' cybersecurity awareness and training. To this end, we aim to make the dataset of SE attacks collected during this study available to the security research community.

In summary, we make the following contributions:

(1) We present a system for large-scale automatic discovery and tracking of SEACMA campaigns, which allows us to study SEACMA-related malicious ads and the SE attacks they lead to. Our system aims to be generic, allowing us to study the SE ad distribution problem without being biased towards specific categories of ad-publishing websites or SE attacks.
(2) We implement a custom browser based on Chromium that makes use of deep code instrumentation to accurately track JS code execution and understand how attackers leverage (or abuse) ad networks to distribute a variety of large SE attack campaigns targeting different types of victims.
(3) We focus in particular on low-tier ad networks and measure which of these ad networks are most likely to distribute malicious ads and study how SEACMA campaigns attempt to evade existing defense mechanisms, such as popular URL blacklists.
(4) Furthermore, we devise a method for continuously tracking SEACMA campaigns over time, and to enumerate the URLs used by the ads to redirect users to the SE attack pages. Our results provide valuable information that could be used to improve defense systems against SE attacks and malicious ads in general.

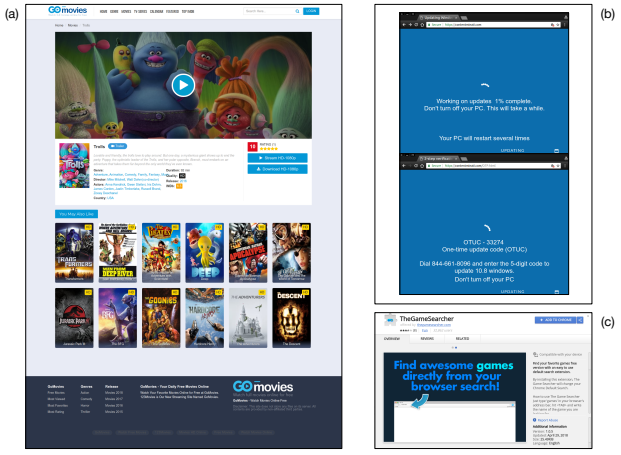


Figure 1: Example of SE attacks reached by clicking on a transparent ad: (a) Publisher page; clicking anywhere on the page will open a new tab (a popup ad) that is likely to redirect the user to SE attacks. (b) SE attack 1 – Tech support scam. (c) SE attack 2 – Malicious browser extension.

2 BACKGROUND

The online ad delivery mechanisms are often complex and involve many entities [36]. In general, online ads are served via ad networks. To display and monetize ads, publisher websites typically include a snippet of code provided by a chosen ad network within their pages[9]. This code runs in the context of the publisher's web page and therefore has fine-grained control on how the page should be modified to inject ads.

Ads injected by low-tier ad networks often operate in a non-traditional way. For instance, instead of displaying a traditional banner ad, the ad network's code may inject an overlay transparent `<div>` element that essentially implements a *transparent ad* covering the entire page. For instance, consider the example of Figure 1[1]. Wherever the user clicks on the page[2], a new tab is opened pointing to a randomly changing ad domain, such as wduyginqbu[.]com, enynwkvdb[.]com, ewopxadcn[.]com, etc., which in turn may redirect the user to a tech support scam, unwanted software[3], or other social engineering attacks (see Figure 1).

[1] hxxps://gomovies[.]li/trolls/ (URL edited to avoid accidental clicks)
[2] Note that only the first click after page load seems to follow this logic, in this particular example.
[3] E.g., an extension called TheGameSearcher [12], which appears to have gained more than 32,000 users before being removed by Google.

Our analysis, which we performed using techniques described later in this paper (see Section 3.5), found that the randomly changing ad domains mentioned in the above example all belong to a single ad network, namely popads.net; this is likely a tactic used by popads.net to evade ad-blockers.

Definition 1 – SEACMA ad. *We define an* SEACMA-related ad *(or* SEACMA ad*, for brevity) as a malicious online advertisement that, when clicked on, redirects the user to a social engineering attack.*

In the above example, the SEACMA ad is represented by a transparent `<div>` ad (Fig 1a). The SE attack itself is in the content shown to the user on the new tab - Fig 1b: a tech support scam page, Fig 1c: a malicious extension download page. During our study we have observed many other examples of such attacks. For example, some landing pages contained text to lure the user to click 'Allow' on a browser push notification request by promising adult content.(Fig 6e)

As seen in the previous example, SEACMA ads themselves may use social engineering to increase their click-through rates. Transparent ads are probably the most extreme example, in that the user intends to click on desired content but is instead "tricked" into clicking on and visiting pages that host SE attacks. Other examples include popup ads that automatically replaced the visited page with the advertised content, fake download buttons embedded in ad banners and other visually misleading components [31]. Nonetheless, in our definition of SEACMA ads we focus mainly on the fact that the ad directs the user to an SE attack page, rather than the visual content rendered by the ad itself.

It is important to notice that SEACMA-related ads are a subclass of the broader class of malicious online advertisements. For instance, some malicious ads may lead the browser to *drive-by* malware download attacks. Such attacks do not require any action on the side of the user, because the browser is compromised and forced to download and execute malicious software without any user intervention. On the contrary, our focus is on ads that lead to SE attacks, in which the landing page reached via an ad click uses social engineering tactics to force the user into performing actions that can lead to a security and/or privacy breach. Various SE attacks we discovered during our study are presented in Section 4.3 and in Appendix A.

Definition 2 – SEACMA campaign. *In some cases, the very same (or almost identical) SE attack content can be reached via different ads served by different low-tier ad networks and published on a diverse set of websites. We define an SEACMA campaign as the set of SEACMA ads that point to the same SE attack content.*

In Section 4.3, we show that this type of SEACMA ads appear on a large variety of website categories, with some of the publisher websites reaching popularity rankings above 1000.

One common feature among SEACMA campaigns seems to be that they are often intrusive, as they are typically implemented using pop-up or pop-under ads. This intrusiveness helps the ads to counter-act the "ad blindness" effect [23], thus maximizing ad revenues. Such ads may be particularly effective on mobile device, where the limited size of the user interface can trick the user in believing the visible ad page is the only page open on the browser. Furthermore, aggressive JavaScript code that attempts to "lock" the browser on the current page is often used as a strategy to force

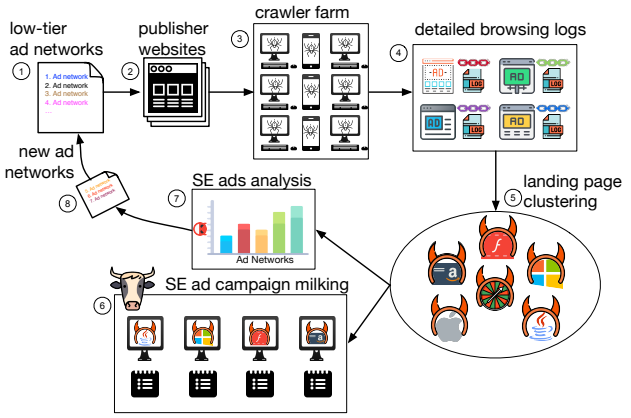


Figure 2: System Overview

inexperienced users into performing undesired actions and falling for SE attacks.

Willingly or not, publisher sites are accomplices in the delivery of this type of malicious ads, in that they allow third party code (the ad networks' scripts) to gain full control of their pages in return for higher ad revenues than what may be gained from reputable ad networks.

3 SYSTEM DETAILS

In this section, we present each module of our system in detail following the system overview depicted in Figure 2.

3.1 Ad Networks and Publisher Sites

To discover SEACMA campaigns, we take an active approach that requires analyzing websites that are likely to advertise (willingly or not) SE attacks. To this end, we first need to assemble a *seed* list of such publisher websites for crawling.

Our personal experience, as well as prior research [33], suggests that SEACMA ads are commonly found on websites that offer illicit content, such as pirated movies and TV shows, or live sport streaming services that violate copyright laws. However, in our study we aim to take a more generic approach that is not biased towards specific publisher websites. Taking a step back, an initial analysis of illicit streaming sites suggests that it's not the publisher sites themselves that are directly responsible for distributing malicious ads. Rather, such websites simply include ads distributed by low-tier ad networks, with little control on what ads will actually be offered to their visitors.

To find a seed list of commonly used low-tier ad networks ①, we manually scouted websites that discuss how to maximize ad revenues [3, 6]. Overall, we compiled an initial list of 11 different ad networks (reported later in Table 3). By creating temporary publisher accounts with these ad networks (without ever needing to actually publish an ad) and by investigating a random set of publisher web pages that used those ad networks, we obtained the JavaScript (JS) code snippets needed to include ads from each of the 11 ad networks.

Then, by analyzing the collected JS code snippets, we immediately noticed that most of these ad networks heavily obfuscate their code and frequently change the domain names from which the JS code is fetched, in order to evade detection by ad blocker extensions. However, we found that it was possible to identify a number of invariant features, such as a specific URL path name, URL structure, or JS variable names that are reused across different versions of JS code snippets belonging to the same ad network. Therefore, we manually collected a number of invariant features for each of the 11 ad networks in our initial list. We then leveraged `publicwww.com`, a legitimate source code search engine, to "reverse" these features and obtain a list of publisher websites that embed code having those invariants. In this way, we were able to obtain 93,427 distinct publisher websites that use one or more of the 11 low-tier ad networks in our list and that are therefore likely to host some malicious ads ②. For each ad network, it took us only about 15 minutes of effort to both derive an invariant pattern and to find the list of associated publisher websites.

3.2 Crawler Farm

To enable the discovery of SEACMA campaigns at scale, we built a highly efficient crawler farm ③. We designed each crawler as a container-based application, allowing us to execute many crawler replicas in parallel. Each crawler in the farm houses a web browsing module, which includes an instrumented full browser capable of logging very detailed information about how web content (including JS code) is rendered, plus custom browser automation code that allows for piloting the browser while avoiding being blocked by anti-automation scripts. Our browser module operates in headless mode and automatically mimics different browser/OS combinations, including mobile operating systems (e.g., screen size and other system properties are adjusted according to the mimicked system). Each crawler visits a selected publisher website and navigates around the site's content by generating mouse clicks, including interacting with ads. During this process, the browser is instructed to also collect screenshots of the visited pages.

Our browsing modules consist of a Docker container running a headless instrumented version of the Chromium browser. Specifically, our Chromium browser consists of a re-implemented and enhanced version of JSgraph [26], which we ported over to a recent Chromium release (64.0.3282) and augmented with automated instrumentation techniques for tracking all JS API calls across the entire Blink-JS bindings (whereas JSgraph [26] can only log a small, manually-instrumented set of JS API calls). All interactions with this headless browser are commandeered with the help of a custom Chrome DevTools API client that we developed to avoid anti-bot checks implemented by some of the ad networks to detect existing automation tools, such as Selenium WebDriver [15].

Via pilot experiments, we observed that many ads, including SEACMA ads, were tailored to the kind of operating system or browser being used. So, in order to collect a diverse set of SEACMA ads, we programmed our crawlers to visit each publisher website by making use of multiple user agent strings. Specifically, we simulated the following Browser / OS combinations: Chrome 66 on macOS, Chrome 65 on Android, IE 10 on Windows, and Edge 12 on Windows. In the case of Chrome 65 on Android, we also made use of Chrome's own device emulation system, which is part of DevTools [7]. This allowed us to also accurately emulate other mobile browser properties, such as screen size, besides the user agent string. We acknowledge that while replacing the user agent string may not be sufficient in some cases to elicit a targeted ad (e.g., the ad network may implement more sophisticated browser fingerprinting techniques), in practice we found it to be very successful.

When visiting a publisher website, each crawler attempts to discover and interact with potential SEACMA ads by issuing mouse clicks (or tap gestures, in the case of the emulated mobile browser). To identify what elements to click on, we developed a number of heuristics based on practical experience. For instance, the crawler identifies elements such as images and iframes, computes their rendering size on the page and sorts them in descending order of their size. Larger images and iframes typically contain the most visually appealing content, which is often associated with event listeners injected by the ad networks. Furthermore, in many cases, as in the example presented in Section 2, the ad network code will listen for clicks anywhere on the page. Our heuristics also cover this case, since clicking on any image triggers the expected ad behavior.

The crawler performs a number of clicks per page, until a given (tunable) number of ads have been triggered and the potential resulting SEACMA ad and SE attack have been recorded. As a heuristic, we assume to have exercised an ad if a click triggers the opening of a new tab pointing to a third-party URL, or if the current tab navigates away from the current website (i.e., it loads a web page with a domain that is different from the publisher website). Again, while these heuristics are not perfect and can clearly capture content not related to SEACMA ads, they do work in a large number of cases, especially when SEACMA ads are in fact present on the page, as for the example in Section 2. It is also worth noting that all the logs captured during this phase are post-processed to filter out non-SEACMA ads (see Section 3.3).

At every crawler click, we record a screenshot and full URL of both the publisher page we clicked on, as well as the opened third-party page. Furthermore, our instrumented browser continuously records fine-grained details about events internal to the browser, such as calls to any JS API, all JS code compiled and executed by the browser, all visited URLs (including any redirections), etc.

Many low-reputation publisher websites tend to be "greedy" and include code from multiple ad networks. Often, this results in multiple ads being associated to the same elements in a page, which may be activated in sequence (one per user interaction) to maximize ad revenue. Therefore, whenever we detect that a click at a particular location on the page has triggered an ad, we repeat the same action a (tunable) number of times, in an attempt to trigger additional ads from different ad networks.

After each interaction with the publisher site (if the browser navigates away from the initial page) we simply re-open the browser, re-load the page and interact with the next candidate element in our list of images and iframes to be clicked. We terminate the analysis of a given publisher site when we run out of elements to click on, reach a maximum number of interactions with the page, or exceed a tunable time-out. We apply these heuristics to strike a balance between the depth at which we analyze a page and scalability, since we have a large list of different publisher pages to explore.

Implementation Challenges and Solutions It is well known that some browser automation tools such as Selenium WebDriver and Phantom JS can be easily detected by existing anti-bot JS libraries [5, 17]. To implement a stealthier automated browser, we therefore decided to leverage Chromium's DevTools protocol, which enables a fine-grained and transparent control of the browser. However, even using our custom DevTools client, we were initially unable to properly mine ads from a few target ad networks. We investigated these problematic cases by reverse engineering the JS code provided by the ad networks. This made us realize that Chromium DevTools, when active, sets `navigator.webdriver` property to indicate that the browser is being automated. This was being checked by ad networks' code to detect browser automation. We therefore further instrumented the browser source code to remove this feature, making our DevTools custom client stealthy.

We also noticed that a couple of ad networks in our seed list (namely, Propeller and Clickadu) respond to ad requests from our institution's machines, Tor exit nodes, and Amazon AWS IP ranges differently, compared to ads served to residential IP addresses. Specifically, we empirically observed that these ad networks seemed to never serve SEACMA ads (instead providing only benign ad content), when visiting the publisher sites from non-residential IP space. To work around this obstacle, we deployed multiple instances of our crawler on three different laptops, which we then connected to residential networks for a few days. This allowed us to collect a significant number of SEACMA ads from those ad networks as well.

Many SE attacks are quite aggressive in the way they capture users' attention. For instance, the landing pages reached via SEACMA ads often try to keep their tab "locked" to the screen. If the user tries to navigate away from the tab, the SE page may call the `alert()` JS API or use other tricks to make it very difficult for the user to leave the page. Among the techniques used for "locking" the page, we found that JS modal dialogues, repeated authentication dialogs, or `onbeforeunload` event handlers were the most common. In case when the SE page displays attacks such as tech support scams or fake anti-virus alerts, these page locking tactics may make the user believe something is actually wrong with their machine and thus steer them towards actions they might not otherwise take. As for our browser automation code, these tactics cause problems to being able to automatically taking screenshots and being able to navigate away from the page to mine more ads. To avoid these issues, we instrumented the browser source code that handles all JS modal dialogs, authentication dialogs and `onbeforeunload` event handlers to bypass these page locking tactics.

3.3 Discovering SEACMA Campaigns

After crawling the entire list of publisher websites, our next task is to discover SEACMA campaigns and filter out irrelevant ads. To do this, we leverage a key trait of SEACMA campaigns: the landing pages reached by clicking on SEACMA ads present the same (visually identical or very similar) SE attacks to the user, but are hosted under frequently changing domain names. The main reason for this is clearly to evade URL blacklists. Conversely, benign ad campaigns have no incentive to frequently change their domains.

Before beginning the clustering process, we first extract the SE attack screenshots obtained from crawling the publisher websites (see step ④). We then compute a perceptual hash, specifically a 128 bit *difference hash* (dhash) [11], on all these screenshot images. Perceptual hashing is often used to efficiently find near-duplicate images [39] and for reverse image search (e.g., as in TinEye), because very similar images tend to produce similar perceptual hashes.

For each screenshot, we also extract the effective second level domain[4] (e2LD) from the URL of the page on which the screenshot was taken. This allows us to obtain a set of distinct $(dhash, e2LD)$ pairs, on which clustering is applied (see step ⑤). To cluster them, we define the distance function between such pairs to be the Hamming distance between the dhash values. For instance, given two distinct pairs $s_i = (dhash_i, e2LD_i)$ and $s_j = (dhash_j, e2LD_j)$, we measure the Hamming distance between the respective 128 dhash bit strings, $d(s_i, s_j) = dist(dhash_i, dhash_j)$. Then, to cluster the pairs we use DBSCAN, setting the algorithm parameters $eps = 0.1$ and $MinPts = 3$ by tuning them via pilot experiments.

Finally, we apply a filtering process to the clustering results, to discard clusters that are unlikely to be related to SEACMA ads. Given a cluster, C_k, we filter it out if its total number of distinct domains: $|\{e2LD_j \mid e2lD_j \in C_k\}|$ is less than a tunable threshold θ_c (we set $\theta_c = 5$, in our experiments). This is motivated by the observation discussed above, that SE ad campaigns typically host visually similar SE attacks on many different domains, to evade static URL blacklists. After filtering, each of the remaining clusters represents a different candidate SEACMA campaign.

3.4 Ad Loading Process Reconstruction

The next two steps in our analysis (see Figure 2, steps ⑥ and ⑦) are to continuously track the discovered SEACMA campaigns and attribute back the related SEACMA ads to a specific ad network. For both steps, we need an additional analysis component: a detailed reconstruction of the browser-internal events involved in the ad loading and landing page delivery process, which we describe below.

Our main goal is to reconstruct fine-grained information about the sequence and relationship between all URLs involved in the process of (1) displaying an SEACMA ad on the publisher page and (2) loading the SE attack after the interaction with an ad. While previous research on malicious ads [24, 28, 30] mostly focused on URL redirection chains obtained by analyzing HTML, JS source code, and network logs, this is not sufficient in our case. For instance, during our study we noticed that it is very common for ad networks to use obfuscated JavaScript code in order to make it difficult for others to block their ads. This obfuscated ad delivery code often initiates page redirections in complex ways and can easily suppress the referrer URL from subsequent HTTP requests (e.g., this could be done by leveraging HTML referrer policies [21]). In general, we observed that there exist a variety of ways in which these ad load mechanisms can take place, including different types of HTTP redirect [10], using Meta Refresh [20], JS-based navigations via `window.location`, `history.pushState` and `history.replaceState`, `addEventListener`, `setTimeout` etc.

[4] We use Mozilla's Public Suffix List: `publicsuffix.org`

To get around these difficulties, we leverage our instrumented browser (an enhanced re-implementation of JSgraph [26], porting the code to a more recent version of Chromium - 64.0.3282) to dynamically track JavaScript execution and record the entire set of URLs used to load a SEACMA ad and SE attack content from inside the browser, including all URLs required to load third-party JS code involved in the rendering of the ad. This is necessary to enable attributing a specific SEACMA ad back to the specific ad network that served it in the first place (see Section 3.6). Essentially, in a way analogous to JSgraph [26], our fine-grained browser logs allow us to start from an SE attack page (the final landing page reached after clicking on a SEACMA ad) and build a *backtracking graph* that reveals all URLs that were involved in publishing the ad and reaching the attack page. For an example, see Figure 3.

3.5 Tracking SEACMA Campaigns

We now discuss how we track SE ad campaigns ⑥.

As mentioned earlier, SE attacks reached from SEACMA ads are often hosted on throw-away domains that tend to last for only a short time (on the order of hours to few days). However, while analyzing the ad loading process of SEACMA campaigns, we noticed that there often exists an "upstream" domain in the ad-loading URL sequence which tends to last a lot longer. This is akin to malicious traffic distribution sites discovered in studies about the dark web [27]. There is a reason for why malicious advertisers may want to use this type of architecture. For the advertiser, there is always a risk of the SE attack domains getting blacklisted. If the SE attack domains are frequently blocked, it may be difficult for the advertiser to continuously update the ad network about these changes. Even if updated URL information can be propagated to the ad network in an automated way, there might be a time lag before the new domains come into effect, depending on how the ad network is implemented. This could also happen due to a variety of complications that are involved in the ad distribution process, such as ad exchange networks and ad syndication [22]. In other words, there could be a period of time during which visits to the SE attack pages may be blocked, thus causing revenue loss for the malicious advertiser. Injecting a level of indirection is a mitigation to this potential issue.

Unfortunately, while the URLs of SE attack pages do get blocked by blacklists such as Google Safe Browsing (GSB), during our study we noticed that their upstream URLs are not typically blocked (at least not as promptly) by GSB. Furthermore, via pilot experiments we observed that in most cases we can reach fresh (not blacklisted) SE attack page URLs by simply re-visiting the upstream URL for the related ad campaign, without having to interact with the publisher page or the ad networks. We refer to these upstream URLs as the "milkable" URLs of a SEACMA campaign.

As an example, Figure 3 shows how we can identify a milkable URL for a scam campaign we discovered during our experiments. The figure depicts all the URLs involved in the process of loading a *Technical Support Scam* SE attack on a Spanish sports streaming website (`verbeinlaliga[.]com`). Besides the publisher site, there were three other domain names involved in leading to the attack. Our ad attribution analysis (to be described in Section 3.6) revealed that the second domain in the chain, `nsvfl7p9[.]com`, belonged

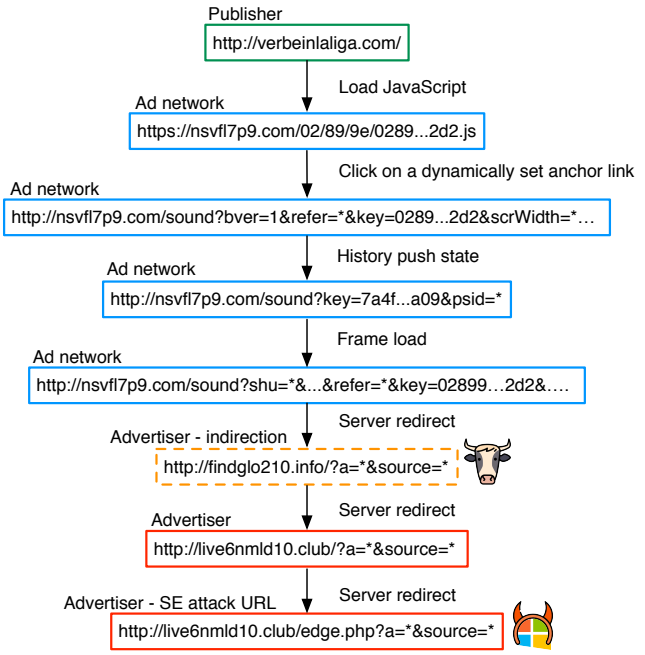


Figure 3: Backtracking graph for an SE attack [5]

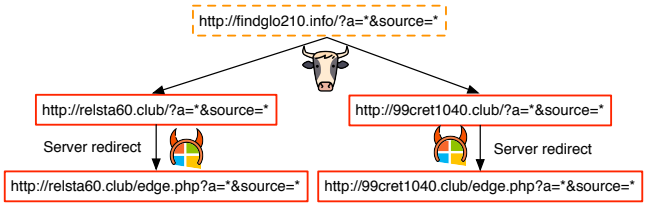


Figure 4: Milking `findglo210.info` [5]

to the AdSterra ad network. The final URL, hosted on `live6nmld10[.]club`, delivered the actual SE attack content. After an hour, this URL became unreachable. However, at this time, querying the "upstream" `findglo210[.]info` URL led to another domain, namely `relsta60[.]club`, that contained the same SE attack with same URL pattern (see Figure 4). A little while later, the `findlog210[.]info` URL led to `99cret1040[.]club` domain containing the same attack and so on and so forth. Because of this behavior, we consider the `findglo210[.]info` URL as a milkable URL.

In order to automatically extract these milkable URLs, we first analyzed the backtracking URL graphs mentioned above for each of the SEACMA campaigns. All these graphs have the URL of the SE attack page as their last node (i.e., the start of the backtracking). Starting from the attack page URL, we navigate through the backtracking graph, until we reach a node associated to a URL that is not hosted on the attack page's domain. We consider such URLs as *candidate* URLs for milking.

We then point our crawler to the candidate URLs and capture a screenshot of the landing page they lead us to. Finally, we compare

[5] Parts of URLs in the figure have been replaced by "*" and "..." to conserve space

the newly "milked" screenshots with our previous collection of screenshot for the SE campaigns. If, given a candidate URL U_m and the landing page p_m it leads to, we find a close match between the screenshot taken on p_m and the previous screenshots from the SE ad campaign from which U_m was derived, we say that U_m is "milkable," and periodically visit it to track new domain names used to distribute the SE attack served by the campaign. This entire process is fully automated.

We setup our crawlers (Section 3.2) to visit each milkable URL once every 15 minutes. Every time we discover a previously unseen SE attack URL (i.e., the landing page with a screenshot similar to previous SE attacks), we store the URL and immediately check it against Google Safe Browsing, to determine how quickly these new attack URLs are typically blocked. Furthermore, using our instrumented browser, we also perform simple interactions with the SE attack URL (e.g., mouse clicks). If this results in a file download, we save the file and upload it to VirusTotal, to obtain anti-virus detection results. Overall, we discovered that the vast majority of milked SE attack URLs, including URLs related to malicious software, are not initially blocked by GSB. We discuss our findings in detail in Section 4.5.

3.6 Discovering New Ad Networks

Our *seed* list of publisher websites was obtained by "reversing" an initial list of low-tier ad networks (see Section 3.1). However, visiting a publisher site on the list does not guarantee to yield ads from one of those ad networks. Like most websites, the publisher websites in our list are dynamic and their content may change in time. This includes the fact that these websites may at some point adopt new ad networks, in alternative to the ones we initially identified. Furthermore, as mentioned earlier, these websites may host ads from several different ad networks, to maximize revenue. Therefore, when crawling these sites we do not have prior knowledge of what ad network is involved in distributing the ads the crawler will interact with.

To link an SE ad campaign to the ad networks that distributed its ads, we implement an *ad attribution* process ⑦. For this, we simply use the invariant patterns for each ad network such as specific URL structures or JS variable names that we previously obtained (see Section 3.1). For each URL in the ad loading and landing page redirection process, we automatically check if it matches the invariant pattern for any of the ad networks. If none of the URLs match any of the patterns, we label that ad as "unknown" and leave it for manual analysis. This allows us to discover new, previously unknown, low-tier ad networks that have served SEACMA campaigns. These ad networks could then be added to our initial *seed* list of ad networks to further expand crawling and SEACMA campaign coverage (see Figure 2).

4 MEASUREMENT RESULTS

To evaluate our system, we used five different Linux-based server machines with 12-32 cores and 24-128 GB of memory per each machine, running Ubuntu 16.04. On average, each crawler had access to 1 core and 2 GB of memory.

In the following, we describe the setup of our measurement infrastructure and report on the SEACMA campaigns we discovered. Notice that many of the SE attacks we discover in our measurements mostly belong to known attack categories (e.g., fake software, technical support scams, survey scams, etc.). However, it is important to remember that our main goal in this paper is not to uncover new *categories* of SE attacks, but rather to discover and track the ad campaigns that lead to generic SE attacks and to study the mechanisms used to deliver such attacks via SE ad campaigns. Furthermore, we measure the number of SEACMA ads delivered by different ad networks, and show that SE attacks delivered via ads are often able to evade existing defenses, such as the popular Google Safe Browsing blacklist.

To facilitate future research, including research in the areas of SE defense, malicious online ad detection and user security awareness training, we are releasing all browser logs and screenshots related to the SE attacks that we collected during our experiments. We are also making available the source code for all components of our system including the crawler, the instrumented Chromium browser, the SEACMA campaign discovery module and the ad attribution module. The code and data will be available here: `https://github.com/phani-vadrevu/seacma`

4.1 Crawling Setup

The list of 11 popular low-tier ad networks that we used as seeds for crawling is shown in Table 3. By leveraging the source code search engine PublicWWW.com, we queried for snippets of code extracted from the the seed ad network scripts and were able to obtain 93,427 distinct publisher websites that used those ad networks. As mentioned in Section 3.2, during pilot experiments we realized that the Propeller and Clickadu ad networks don't serve SEACMA ads from non-residential IP space. Hence, we divided this pool of websites into two datasets, a group of publisher sites that did not appear to include Propeller and Clickadu ads and a second group so sites that did. We visited each website with our containerized automated browser and spent roughly two minutes per session. We repeated the crawling using four different user agents, to quickly mimic different platforms. We crawled publisher sites from the first group from our institutional network, whereas we crawled sites containing Propeller and Clickadu ads using three separate machines (three common Linux-based laptops with Intel i5 processors) with residential Internet access. Specifically, the first group consisted of 59,359 publisher sites, whereas the second group (sites that hosted ads from Propeller and Clickadu) included 34,068 different sites. However, due to the limited bandwidth of residential networks and hardware capacity of laptops we used, we were only able to visit about 11,182 websites from the second group. Therefore, overall, we interacted with 70,541 websites as part of this experiment.

4.2 Milking URLs Setup

After crawling the publisher sites, we ran a clustering algorithm on the screenshots of the landing pages to discover SEACMA campaigns. The algorithm along with the parameters used is described in Section 3.3. We next processed the URL backtracking graphs belonging of SEACMA campaigns to obtain candidate URLs for milking (see Section 3.5). We then ran a small pilot experiment to ensure that each of these candidate URLs (paired with different

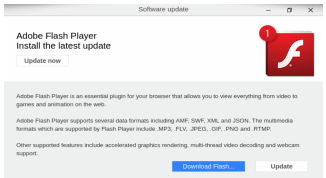
User Agents strings for crawling) were useful milking sources. This resulted in 505 distinct (URL, User Agent) pairs as milking sources. We then began to milk these URLs for a period of 14 days. We performed the milking using a set up similar to the web crawling setup described earlier. Our system automatically interacts with the SE attack pages and offloads all the milking data, including screenshots, logs and downloaded files, to a file server. With this set up, we were able to repeat a total of 505 milking sessions roughly once every 15 minutes.

Every time one of the milking URLs led to a "never-before-seen" domain hosting an SE attack, we added that domain to a list of domains to be looked up using the Google Safe Browsing (GSB) API. All new domains in our GSB lookup list were checked against the GSB API every 30 minutes. Similarly, every time a new file download was triggered by interacting with SE attack pages, we immediately looked up the file hash against VirusTotal. If the file was previously scanned, we simply stored the existing VirusTotal report in our database. Then, at the end of our milking experiment, we sent all unknown files to VirusTotal for first-time scanning.

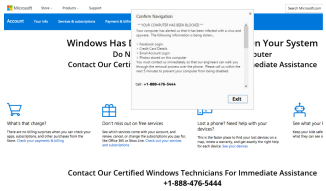
4.3 Discovering SEACMA Campaigns

As described in the previous section, we have done the *Crawling Experiment* to visit about 70,541 publisher websites while masquerading as various different User Agents. During this experiment, clicks (or taps) with about 39,171 publisher websites resulted in third-party landing pages (about 199,400 pages) being opened. In all these cases, screenshots of the third-party pages were taken and were input to a clustering algorithm as described in Section 3.3. This resulted in 130 clusters. We then used various methods to determine the ground truth for each cluster: (1) Visual inspection of one of the sample images in the cluster. (2) Interaction with page and page source code inspection. (3) Verification using external sources such as Google Safe Browsing and VirusTotal. For many attacks such as Chrome Notifications and Fake Lottery, the visual inspection itself was enough to ascertain that the cluster represents a SEACMA campaign (Appendix A). If a cluster did not appear to represent a SEACMA campaign, we considered it to be a *benign* cluster. Out of the 130 clusters we obtained, 108 clusters appeared to be clearly representing various SE attack campaigns. Overall, these campaigns included 11,341 unique domains of publisher sites and 28,923 SE attack instances reached by clicking on the SEACMA ads hosted on those publisher sites. Some screenshots related to these SE attack campaigns we discovered are shown in Figure 5 and in Figure 6 in Appendix A.

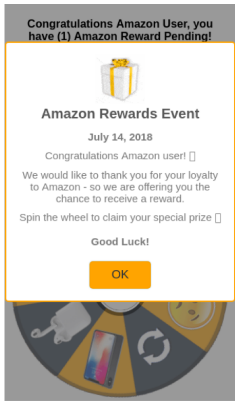
Table 1 groups the SEACMA campaigns into different categories. The table also shows the number of SE attacks seen for each category in the second column. The number of distinct SE attack domains (i.e. different landing pages) is shown in the third column, whereas the number of campaigns that belong to this category is shown in the fourth column. The fifth column shows the percentage of SE attack domains that have been blacklisted by Google Safe Browsing (GSB). The final column shows the percentage of SEACMA campaigns that have *at-least one attack domain* blacklisted by GSB. It is worth noting that many SEACMA campaigns are not at all detected by GSB, though we manually confirmed that the content they deliver is in fact malicious.



(a) Fake Flash Player



(b) Technical Support scam



(c) Lottery Scam

Figure 5: Some discovered SEACMA campaigns

Category	# SE Attacks	# Attack Domains	# SE Campaigns	GSB Detection %	
				domains	campaigns
Fake Software	16802	2370	52	15.4%	73.1%
Registration	2909	474	36	0%	0%
Lottery/Gift	4297	50	9	18%	66.7%
Chrome Notifications	3419	102	3	0%	0%
Scareware	1032	71	5	0%	0%
Technical Support	464	74	3	1.4%	33.3%

Table 1: SE Ad campaign statistics

The SE attack categories represented in Table 1 are briefly described below:

(1) *Fake Software*: These campaigns contain attacks where fake software is advertised as Java updates, Adobe Flash updates or macOS media players. An example is shown in Figure 5a.
(2) *Scareware*: Scareware SE attacks are similar to the Fake Software category. However, the SE tactic is different. Instead of promoting the software as a fake video player update, the user is scared into believing that their machine system is currently infected with malware, for example.
(3) *Technical Support Scam*: Technical support scam attacks are another popular category of SE attacks and have been studied in [30, 34]. An example of this attack is shown in Figure 5b. These are cross-channel attacks that make use of the telephony channel to monetize the SE attack, by convincing the user to reach out to a (fake) Microsoft (or Apple) support center for assistance with their malfunctioning computer. This is accomplished using various web features (such as alert windows, print dialogs, etc.) to give the user the illusion of their system being locked. Our system provides an automatic real-time way to collect these scam phone numbers and add to a blacklist to protect users.
(4) *Fake Lottery*: These SE attacks pretend to be lotteries or gift cards that the user won. They will lead to a survey in which user details are requested. These attacks are specific to mobile platform and we have not observed them on desktop platform. An example of such attacks is shown in Figure 5c.

The survey scams to which users are redirected by interacting with these SE attack pages were recently studied in [24]. Our system provides an automatic way of collecting the gateways for such survey scams.

(5) *Chrome Notifications*: These campaigns are an example of how SEACMA ads have evolved in time. In recent years, Chrome has tried to curb the menace of obtrusive and malicious advertising by using techniques such as an in-built ad-blocker, strict ad policies etc. [4, 8] (see also Section 5). To compensate for lost revenue, ad networks began to look for newer avenues in advertising. One example of this is the push notifications mechanism introduced in the recent versions of Chrome (akin to push notifications on mobile phones) [2]. First, an SE attack is used to lure the user in allowing push notifications from a given landing page. From then on, the user could be sent potentially malicious notifications even if the user never visits the SE attack page directly again.

(6) *SE Registration*: This group of clusters represent networks of websites that drive traffic to media, book, streaming and dating websites who are paying customers of these networks. In doing this, they tend to use SE techniques. Usually, there is a page containing a fake video player (as verified by the source code) that pretend to play the movie that your are looking for (based on the search term). After a couple of seconds it stops and asks for the user to create an account in the advertised site. Upon agreeing, the user will be re-directed to the customer's account registration page. Thus, by employing a SE technique the users are enticed into creating an account on other scam websites which are often accused of a number of fraudulent credit card charges [13, 14, 18, 19].

Among the 130 clusters we obtained, 22 did not appear to be directly related to SEACMA campaigns. After fully investigating these 22 clusters, we were able to categorize them as follows:

- 11 of the 22 clusters represented parked domains (i.e. domains that are expired) or inaccessible domains (due to country restrictions, some domains were not accessible from the US). In both cases, the original content of the landing page is replaced by placeholder content that would be similar across multiple unrelated domains. Most of these domains could be automatically filtered out using parking detection algorithms [38]. We leave adding this automated filtering component to future work.
- In 6 of the 22 clusters the landing pages had some stock images which when clicked on would redirect to different adult websites. Due to the presence of these stock images in different domains, these pages ended up among the perceptual clusters. However, we also noticed that in most cases, the content that is promised is actually not present in the final page. These pages are essentially a way to lure the user and hence could be considered as another form of an SE attack.
- 4 clusters were related to two popular low-tier URL shorteners: `adf.ly` and `shorte.st`. They show framed ads on their own domains before redirecting the users to the target pages. These services have registered many other domain aliases. As a result, the ads tend to be displayed on many different domains with similar page structure and hence tend to be clustered together. Note that many of the ads displayed by these services are in-fact malicious ([30, 32]). Nonetheless, we still consider them as "benign" in our work, because they are associated with a URL shortener service, instead of being directly associated to a particular SE attack. Since the number of such services is small, it is possible to whitelist them.
- Finally, there was a spurious cluster formed from just 5 domains. This was due to some technical issues relating to improper loading of pages.

Using a publicly available API [16], we also tried to categorize the publisher websites that host SEACMA ads. This information is presented in Table 2. The table shows the categories of websites that hosted the SEACMA ads that deliver the SE attack campaigns we discovered. As can be seen, our system is highly generic and is not restricted to any specific category of publishers.

Category	# Publisher Domains	% of Total Domains
Suspicious	1632	15.81
Pornography	1396	13.52
Web Hosting	914	8.85
Entertainment	678	6.57
Personal Sites	667	6.46
Malicious Sources/Malnets	645	6.25
Dynamic DNS Host	475	4.60
Technology/Internet	415	4.02
Piracy/Copyright Concerns	404	3.91
Games	321	3.11
TV/Video Streams	282	2.73
Phishing	254	2.46
Business/Economy	186	1.80
Adult/Mature Content	178	1.72
Sports/Recreation	157	1.52
Education	154	1.49
Social Networking	112	1.08
Placeholders	108	1.05
Health	104	1.01
Society/Daily Living	101	0.98

Table 2: Top 20 categories of SEACMA ad publisher sites

According to PublicWWW.com, which also provided us with the popularity ranking for each seed website, we found that 52 publisher websites were ranked among the top 10,000 most popular domains, showing that some popular websites are also impacted by SEACMA ads. Furthermore, 4 of the publisher sites were in the top 1,000 most popular websites.

4.4 Ad network analysis

To link the SE attacks described above to the ad network responsible for distributing them, we used the ad attribution process described in Section 3.6. We were able to link a majority of the discovered SE attacks to one of the seed 11 ad networks we started out with. Of all the SE attacks we observed, 23,435 (81%) were associated with one of those 11 ad networks. This is expected, since all publisher websites that we crawled were known to host ads from those networks. Table 3 shows the number of SE attacks that were distributed from each of these ad networks. As mentioned in Section 3.1, some ad networks use a large number of domains to host the JavaScript code responsible for the loading the ads. On the other hand, some ad networks use only a single domain for this purpose. The second column in the table shows these numbers. As we can see, ad networks

such as RevenueHits and AdSterra tend to use many domains for hosting their code. The third column shows the total number of third-party landing pages that were opened as a result of clicks made on ads from each of these networks. The fourth column depicts the number of SE attack pages that were among those landing pages. The final column shows the percentage of SE attack pages from each ad network. Somewhat surprisingly, for three of the ad networks more than 50% of their ads led to SE attacks. These high percentages suggest that some of these ad networks may be aware of the fact that they are a vehicle for delivering SE attack, but may not have the means to counter this type of abuse, or potentially even choose to be complicit by intentionally allowing malicious content to be reached through the ads they distribute.

Ad network	# Ad network domains	# Landing Pages	# SE Attack Pages	% SE Attack Pages
RevenueHits	517	15635	3075	19.67%
AdSterra	578	15102	7644	50.62%
PopCash	2	9734	6256	64.27%
Propeller	4	8206	3470	42.29%
PopAds	3	4658	873	18.74%
Clickadu	10	2814	848	30.14%
AdCash	14	1698	955	56.24%
HilltopAds	46	1198	77	6.43%
PopMyAds	1	1194	103	8.63%
AdMaven	39	496	122	24.60%
Clicksor	4	276	12	4.35%
Unknown	-	-	5488	-

Table 3: SE attacks from each ad network

From Table 3, we can also see that 5,488 SE attacks were reached through unknown ad networks. By manually analyzing 50 of these SE attack logs from our system, we were able to discover a few new low-tier ad networks that were not part of our initial ad networks seed list. This analysis was very quick as our logs already contain the backtracking graphs and associated JS code snippets for each attack. It was just a matter of identifying URLs or JS code artifacts that are similar to one another and investigating those using search engines to find out what ad network is associated with them. We found an ad network called Ero Advertising that appears to be predominantly focused on advertising on adult-oriented websites. We also discovered two other ad networks called Yllix and AdCenter, which were also distributing ads that led to SE attacks. Also, using our logs, we were quickly able to identify the patterns in their ad-serving source code that can be used to identify ads from these networks. Using again the PublicWWW.com search engine, we were able to find 8,981 new publisher websites that host ads from these three networks. This entire process took less than one hour. This shows that analyzing a small amount of "unknown" SE attacks enables the enrichment of the publisher websites that can be crawled to expand the tracking of SEACMA campaigns.

In addition to the experiments reported above, we also ran a number of pilot experiments to test if SEACMA ads delivered by the 11 ad networks in Table 3 could be blocked by the latest Chrome release with the most recent AdBlock Plus extension. We found that only ads provided by Clicksor did not display in this configuration. All the other 10 ad networks continued to provide malicious ads, as during our crawling experiments.

4.5 Milking

Using the setup described in Section 4.2, we milked the discovered SEACMA campaigns during a 14-day period. This resulted in more than one million browsing sessions. From this, we were able to discover a large number of new domain names that were being used for the SEACMA campaigns. These domains were completely new, in that they were never observed during the initial crawling and SEACMA campaign discovery. As explained previously, we looked up each such domain in Google Safe Browsing (GSB) at an interval of once every 30 minutes. We continued these lookups for 12 more days after the 14-day milking period. Then, after 2 months of time, we performed an additional lookup for each of the domains previously undetected by GSB, to measure any late additions to the GSB blacklist. Table 4 summarizes the results. The second and the third columns show the GSB detection rate for the domain at the time of milking and at the latest lookup, respectively. The table also shows the break up of the domains that were milked by category of the SEACMA campaign. We observed that in many cases, soon after a particular domain is added to the GSB list, the upstream milking sources tend to redirect to new SE attack domains that are not blocked by GSB. This is evidenced by the initial low detection rate of GSB. However, even after two months time, GSB detected only a small percentage (16.2% overall) of SE attack domains. For some categories of SEACMA campaigns, such as "Technical Support", "Scareware" and "SE Registration", GSB appears to be even less effective than others. Furthermore, for those domains that were finally detected by GSB, we measured how late GSB is in adding these domains to the blacklist, when compared to harvesting the domains by milking. On an average, GSB is more than 7 days slower in detecting these SE attacks domains.

During the milking process, the automated crawler interacted with the landing pages reached via SEACMA ad clicks. These interactions often resulted in file downloads for SE attacks such as "Fake Software" and "Scareware" categories. Overall, we were able to milk 9,476 files in a 14-day period. We found a majority of these files to be Windows (PE) and MacOS (DMG) executables. For all the downloaded files, we looked up the VirusTotal database for previous Anti-Virus scan reports. We discovered that only 1,203 files were already known to VirusTotal. This is likely because the binaries distributed by those SE campaigns tend to be highly polymorphic, in an effort to evade anti-viruses. We then uploaded all the files to VirusTotal for scanning and waited for a period of three months to request a rescan. This allowed the AVs sufficient time to catch-up on the signatures for the files. In the end, we found that more than 9,000 of the milked files were marked as malicious. More than 4,000 of them were flagged by at least 15 different Anti-Viruses. Trojan, Adware and PUP were among the most popular labels associated with the files we submitted.

5 DISCUSSION AND LIMITATIONS

For all our measurements, we used a heavily instrumented Chromium browser. The instrumentation has multiple purposes: (i) capturing highly detailed logs that allow us to identify and study the JavaScript code related to online ad networks; (ii) automatically pilot user-like actions, such as clicks on pages elements likely related to ads; (iii) capture all types of URL redirections, including

Category	# Domains	GSB-init	GSB-final
Fake Software	1665	1.28%	18.59%
Technical Support	258	2.99%	4.70%
Scareware	45	0.00%	2.27%
SE Registration	47	0.00%	0.00%
Lottery/Gift	27	3.70%	55.56%
Total	2042	1.42%	16.21%

Table 4: Tracking SEACMA Campaigns (milking)

complex JS-driven redirections, to enable discovery of milkable SEACMA campaign URLs and ad network attribution; and (iv) bypass common SEACMA ad "cloaking" techniques, to maximize our ability to collect malicious ads and reach SE attack landing pages. However, our anti-cloaking measures are based on a number of empirically-derived rules. While they help us identify a large number of malicious ads and SE attacks that we would otherwise not be able to witness, the anti-cloaking mechanisms we implemented are by no means complete. Malicious ads (or ad networks) that implement sophisticated browser fingerprinting and robot detection mechanisms may still be able to identify our crawlers and thus avoid exposing the attacks they advertise. Furthermore, to simulate different types of environments, such as different OS and browser types (e.g., Edge on Windows), we only rely on changing the browser's User-Agent string, because recreating the same type of heavy browser instrumentations we implemented on Chromium on other architectures would require additional very significant engineering effort. Nonetheless, simply changing the User-Agent string and other parameters (e.g., screen size, resolution, etc.) allowed us to discover SE campaigns that are clearly targeted to different systems, such as mobile browsers, Microsoft Windows, or macOS-based systems.

Among the 70,541 publisher websites that we crawled, only 11,341 (16%) were observed to host SEACMA ads. However, it should be noted that visiting a publisher site will not always result in displaying a SEACMA ad, even if the publisher site embeds code from an ad network that commonly offers malicious ads. Because of the dynamicity of online advertisements, one might need to crawl the same publisher site multiple times, before encountering a SEACMA ad. To improve scalability and limit the amount of traffic or possible side effects on a given website, we crawl each publisher site in a limited way. Specifically, we do this by avoiding multiple visits to a publisher website with the same user agent. Therefore, it is possible that we missed to observe some SEACMA campaigns that are not frequently displayed to visitors. Also, while some of the categories of SE attacks we we discovered have been studied in the past [24, 30, 33, 34], our main contributions are in building a system for discovering and tracking SEACMA campaigns in a general way, studying how SEACMA ads and attacks are delivered, attributing SEACMA campaigns to the responsible ad networks, measuring the extent to which SEACMA campaigns evade current defenses such as GSB, and providing insights that may help to build more effective defenses against SE attacks.

Automation. While we attempted to automate the entire measurement process to the maximum extent possible, there are still a few manual steps that are difficult to avoid. These are: (1) the compilation of seed list of ad networks, (2) identification of invariant patterns for ad networks (used for both publisher site mining and ad network attribution of SEACMA ads) and (3) discovery of unknown ad networks (Section 3.6). Compilation of seed lists takes a very short time, as there are only a limited number of popular low-tier ad networks. We were able to compile this list in just a few minutes. As already mentioned in Section 3.1, identification of invariant patterns can also happen quickly. On an average, it took us about 15 minutes to obtain an invariant pattern given an ad network's name. It is hard to automate this part as each ad network uses its own methods of obfuscation or URL patterns. However, one can easily find an invariance feature upon inspecting multiple code snippets from different pages using this ad network. In total, we estimate that we spent less than 5 hours on manual components of this study.

Evasion of existing browser defenses. Google announced in February 2016 that its Google Safe Browsing service will stop SE attacks from being displayed [8]. However, even three years later, our results from Section 4.5 show that GSB is still very ineffective in detecting SEACMA ads and SE attack pages. Earlier in 2018, Google further stepped up the battle against SEACMA ads by rolling out an in-browser ad blocking mechanism. This mechanism is intended to block all ads on websites that are prone to show intrusive ads [4]. After several months into the deployment of our system we manually tested the most recent Chromium browser releases against some of the SE attack campaigns we discovered and verified that most of the SEACMA ads remain unblocked. Similarly, popular ad blocking plugins like AdBlock Plus appear to be very ineffective in curbing these ads, as discussed in Section 4.4. One reason for this is the continuous evolution of strategies used by low-tier ad networks to avoid detection and improve effectiveness. In Section 4.4, we have already discussed how ad networks tend to use a large number of domains to host their JavaScript code snippets, to evade detection. Also, as discussed in Section 4.3, we observed SEACMA campaigns that try to abuse the new *browser notifications* feature recently introduced by the Chrome browser to deliver malicious ads. Our system can be used to discover and track these new trends in SEACMA campaigns and thus can help design new and more effective defenses.

6 ETHICAL CONSIDERATIONS

In line with previous malvertisement studies [33], our crawling experiments involved generating artificial clicks on ad-publishing websites, which resulted in advertisers' landing pages being loaded and rendered in the browser. Because our main goal is to discover and track SEACMA campaigns on a large scale, we argue that this type of research would not be possible without actually rendering and clicking on ads. Moreover, our crawler doesn't even target any specific "ad elements" for clicking. In most cases, our crawler interacts with large native elements of the page. However, ad networks set up click event listeners on many elements of the web page using obfuscated JavaScript. So, our crawler invariably triggers the pop-ups of landing pages regardless of where it clicks on the page. But, we limited the amount of harmful affects on legitimate advertisers by avoiding multiple visits to publisher websites using the same user agent. We devised the milking experiments as explained in

Section 3.5 in order to directly track SEACMA campaigns without influencing the advertising costs.

Nonetheless, it is important to consider what the potential impact on third parties may be. To this end we estimated the monetary cost that legitimate advertisers may have incurred during our crawling experiments, and found it to be negligible. We proceeded as follows: given a non-SE domain d (we exclude all SE attack domains from this analysis), we considered all automated clicks produced by our system that led to d, which allows us to estimate the *cost per domain* that an advertiser (i.e., the third party who owns the domain) might experience. We found that the worst case was a page from a legitimate (i.e., non-SE) domain that was opened 1,209 times as a result of our clicks. Assuming a CPM (cost per thousand ad impressions) of USD $4 (based on CPM estimates of low-tier ad networks [1]), we estimate that the advertiser was charged about USD $4.8 due to our experiments. In average (the above example is the worst case in our data), there were about 9 clicks per legitimate (non-SE) domain, resulting in about $0.04 per domain. This shows that our crawling experiment and the proposed system ensured minimal financial losses for legitimate advertisers while yielding results that have multiple larger benefits for security research. Our results show how various ad networks are responsible for distributing SE attacks, how we can discover new SECMA campaigns, how existing URL blacklists can be enriched to include and protect from many new web pages that contain SE attacks.

7 RELATED WORK

Li et al. have developed MadTracer [28] to enrich the malvertisement URLs that can be identified by blacklisting services such as GSB, with the help of similarities in features such as URL patterns. Unlike [28], our system can discover new SEACMA campaigns irrespective of whether or not any of the URLs in the campaign are detected by GSB. This is very important, as we have seen that many SEACMA campaigns are very effective in completely evading GSB (see Table 1). Similarly, Zarras et al. [40] measured malvertisements after identifying them with the help of blacklists and JS-based dynamic analysis, which is again not very effective in the case of SE ads. Nelms et al. [31] studied SEACMA ads via passive network traffic analysis. They developed an SE software download detection engine based on features such as whether the downloads are ad-driven, the reputation of the domains involved, etc. Unlike [31], our system works actively by crawling for SEACMA ads and can discover SEACMA campaigns by visual analysis irrespective of whether or not they result in the download of malicious files.

Recent works such as [24, 30, 33, 34] focus only on specific types or origins of SEACMA campaigns, such as survey scams, Windows support scams, or SEACMA ads on live video streaming websites. Our system is much more generic and is able to discover and track previously unstudied SEACMA campaigns, including SE attacks based on Chrome notifications (see Section 4.3). Supervised classification systems such as SpiderWeb [37], SURF [29], and WarningBird [25] focus on detecting malicious web pages based on the redirection chains of URLs leading them to those pages. On the contrary, our system is completely unsupervised and leverages the visual similarity typical of SE attacks belonging to the same SEACMA campaign.

Starov et al [35] have detected malicious web campaigns with the help of Web Analytics IDs associated with malicious websites. While [35] focuses on the similarity of web analytics being used by the scammers, our work makes use of the similarity of the attack content being hosted on different domains, regardless of whether or not web analytics is being used by these domains. Due to this difference, the two systems are orthogonal to each other and could work in a complementary fashion. Our system also helps to study SEACMA campaigns from the point of view of the ad networks. For instance, we have identified, attributed and quantified the major ad networks that are responsible for distributing SE attacks (see Section 4.4) and we have built our system to enable the discovery of future evolving ad networks.

8 CONCLUSION

In this paper, we have proposed a system for large-scale automatic discovery and tracking of SEACMA campaigns, which we used to study the SEACMA ad distribution problem without being biased towards specific categories of ad-publishing websites or SE attacks. Starting with a seed of low-tier ad networks, we measured which of them are the most likely to distribute malicious ads and proposed a mechanism to discover new ad networks that are also leveraged by attackers to support the distribution of SE campaigns.

The results of our study can be useful in many different ways. We showed that malicious advertisers use a number of tactics to successfully evade URL blacklists. We also analyzed in detail some of the largest SEACMA campaigns we observed and provided insights into the tactics used to propagate SEACMA ads and attack pages. Our results provide valuable information that could be used to improve defense system against SE attacks and malicious ads in general.

ACKNOWLEDGEMENTS

We thank Jeremy Blackburn for serving as our shepherd and the anonymous reviewers for their constructive comments and suggestions for improvement.

REFERENCES

[1] 2019. AdSpyglass - Top Ad Networks List. https://www.adspyglass.com/best-ad-networks?stream=2&country_id=487&type_id=2#best-networks.
[2] 2019. AdSterra - Push Notifications. https://blog.adsterra.com/web-push-notifications-publishers-guide/.
[3] 2019. BloggersIdeas: List of 15 Best Pop-Under Ad Networks 2018. https://web.archive.org/web/20180102104340/http://www.bloggersideas.com:80/best-pop-under-ad-networks/.
[4] 2019. Chromium Blog: Under the hood - How Chrome's ad filtering works. https://web.archive.org/web/20180322060825/https://blog.chromium.org/2018/02/how-chromes-ad-filtering-works.html.
[5] 2019. Distil Networks - Bot Detection. https://www.distilnetworks.com/block-bot-detection/.
[6] 2019. Earning Guys: 14 Best Pop-Under Ad Networks. https://web.archive.org/web/20180104191440/https://www.earningguys.com/advertisement/best-pop-under-ad-network/.
[7] 2019. Google Developers: Simulate Mobile Devices with Device Mode. https://developers.google.com/web/tools/chrome-devtools/device-mode/.
[8] 2019. Google Security Blog: No More Deceptive Download Buttons. https://web.archive.org/web/20180605081413/https://security.googleblog.com/2016/02/no-more-deceptive-download-buttons.html.
[9] 2019. Google Support: Where to place the ad code in your HTML. https://support.google.com/adsense/answer/7477845.
[10] 2019. Green Bytes: Test Cases for HTTP Redirects. http://test.greenbytes.de/tech/tc/httpredirects/.

[11] 2019. Hacker Factor: Kind of Like That. https://web.archive.org/web/20180507130827/http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html.

[12] 2019. Malware Fixes: TheGameSearcher unwanted software. https://malwarefixes.com/remove-thegamesearcher-google-chrome/.

[13] 2019. Online Threat Alerts: Grovelfun reviews. https://www.onlinethreatalerts.com/article/2017/3/1/beware-of-www-grovelfun-com-it-is-a-fraudulent-website/.

[14] 2019. ScamGuard: Funwraith reviews. https://www.scamguard.com/funwraithcom/.

[15] 2019. Selenium - Web Browser Automation. https://www.seleniumhq.org/.

[16] 2019. Symantec WebPulse. https://sitereview.bluecoat.com/.

[17] 2019. thebot.net - forums. https://thebot.net/threads/300-bounty-selenium-detection-help.375708/.

[18] 2019. Trustpilot: Gnomicfun reviews. https://www.trustpilot.com/review/gnomicfun.com.

[19] 2019. Trustpilot: Playperks reviews. https://www.trustpilot.com/review/playperks.net.

[20] 2019. W3C: Meta Refresh. https://www.w3.org/TR/2011/WD-html5-author-20110809/the-meta-element.html.

[21] 2019. W3C: Referrer Policies. https://www.w3.org/TR/referrer-policy/.

[22] Muhammad Ahmad Bashir, Sajjad Arshad, William K. Robertson, and Christo Wilson. 2016. Tracing Information Flows Between Ad Exchanges Using Retargeted Ads. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. 481–496.

[23] Jan Panero Benway. 1998. Banner Blindness: The Irony of Attention Grabbing on the World Wide Web. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 42, 5 (1998), 463–467. https://doi.org/10.1177/154193129804200504 arXiv:https://doi.org/10.1177/154193129804200504

[24] Amin Kharraz, William K. Robertson, and Engin Kirda. 2018. Surveylance: Automatically Detecting Online Survey Scams. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. 70–86.

[25] Sangho Lee and Jong Kim. 2012. WarningBird: Detecting Suspicious URLs in Twitter Stream. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*.

[26] Bo Li, Phani Vadrevu, Kyu Hyung Lee, and Roberto Perdisci. 2018. JSgraph: Enabling Reconstruction of Web Attacks via Efficient Tracking of Live In-Browser JavaScript Executions. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018$_0$7B-4$_{\text{Li}_{\text{p}}}$aper.pdf

[27] Zhou Li, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. 2013. Finding the Linchpins of the Dark Web: A Study on Topologically Dedicated Hosts on Malicious Web Infrastructures. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13)*. IEEE Computer Society, Washington, DC, USA, 112–126. https://doi.org/10.1109/SP.2013.18

[28] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. 2012. Knowing Your Enemy: Understanding and Detecting Malicious Web Advertising. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*. ACM, New York, NY, USA, 674–686. https://doi.org/10.1145/2382196.2382267

[29] Long Lu, Roberto Perdisci, and Wenke Lee. 2011. SURF: detecting and measuring search poisoning. In *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*. 467–476.

[30] Najmeh Miramirkhani, Oleksii Starov, and Nick Nikiforakis. 2017. Dial One for Scam: A Large-Scale Analysis of Technical Support Scams. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*.

[31] Terry Nelms, Roberto Perdisci, Manos Antonakakis, and Mustaque Ahamad. 2016. Towards Measuring and Mitigating Social Engineering Software Download Attacks. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*. 773–789. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/nelms

[32] Nick Nikiforakis, Federico Maggi, Gianluca Stringhini, M. Zubair Rafique, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna, and Stefano Zanero. 2014. Stranger danger: exploring the ecosystem of ad-based URL shortening services. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*. 51–62.

[33] M Zubair Rafique, Tom Van Goethem, Wouter Joosen, Christophe Huygens, and Nick Nikiforakis. 2016. It's free for a reason: Exploring the ecosystem of free live streaming services. In *Proceedings of the 23rd Network and Distributed System Security Symposium (NDSS 2016)*. Internet Society, 1–15. https://doi.org/10.14722/ndss.2016.23030

[34] Bharat Srinivasan, Athanasios Kountouras, Najmeh Miramirkhani, Monjur Alam, Nick Nikiforakis, Manos Antonakakis, and Mustaque Ahamad. 2018. Exposing Search and Advertisement Abuse Tactics and Infrastructure of Technical Support Scammers. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 319–328.

[35] Oleksii Starov, Yuchen Zhou, Xiao Zhang, Najmeh Miramirkhani, and Nick Nikiforakis. 2018. Betrayed by Your Dashboard: Discovering Malicious Campaigns via Web Analytics. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 227–236.

[36] 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

[37] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2013. Shady paths: leveraging surfing crowds to detect malicious web pages. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. 133–144.

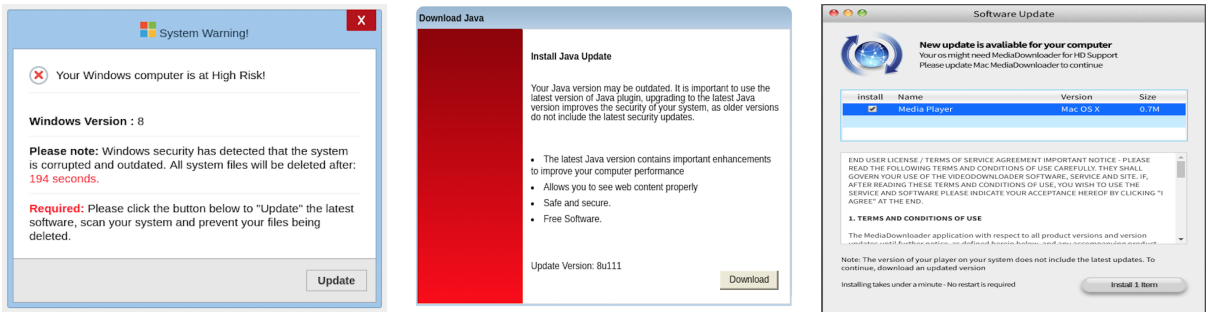
[38] Thomas Vissers, Wouter Joosen, and Nick Nikiforakis. 2015. Parking Sensors: Analyzing and Detecting Parked Domains. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*.

[39] Savvas Zannettou, Tristan Caulfield, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini, and Guillermo Suarez-Tangil. 2018. On the Origins of Memes by Means of Fringe Web Communities. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*. 188–202.

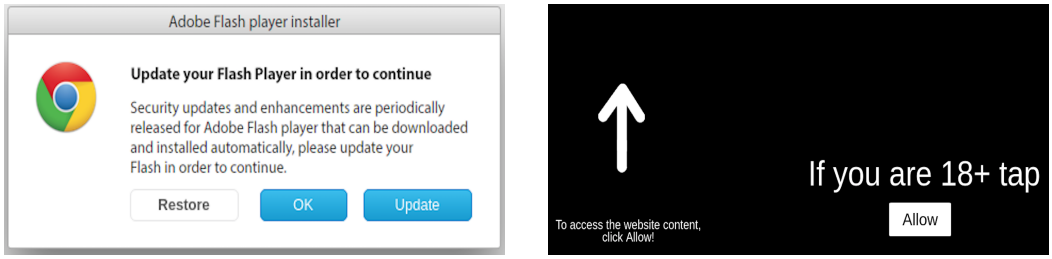
[40] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. 2014. The Dark Alleys of Madison Avenue: Understanding Malicious Advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)*. ACM, New York,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

A CATEGORIES OF SEACMA CAMPAIGNS

Figure 6 shows screenshots of content from various SE attack web pages seen in SEACMA campaigns that we discovered. Many of these images show how various elements such as OSX and Windows' GUI windows (Fig 6c, 6a, 6b), Facebook's page style (Fig 6f, 6g) and Chrome's logos (Fig 6d) are copied to create trust in the users. These images also show how visual inspection of these SE attacks can be made. The fake OS windows rendered inside a browser, the fake Chrome browser window (Fig 6d) and the fake comment boxes (Fig 6f, 6g, 6h) are all tell-tale signs of an SE attack. The Push notification attack screenshot (Fig 6e) shows how the SE attackers are trying to lure the users to click on the "Allow" button of the browser notification request message with the promise of adult content by asking if the user is an adult or not (a message typically shown on adult websites).



(a) Scareware - Windows (b) Fake software - Java (c) Fake software - Mediaplayer



(d) Fake software - Flash (e) Browser push notifications

(f) Fake gift (g) Fake lottery (h) SE Registration

Figure 6: Screenshots of content from various SE attacks seen in SEACMA campaigns