computers

MDPI

Article

Click Fraud in Digital Advertising: A Comprehensive Survey

Shadi Sadeghpour * and **Natalija Vlajic**

Department of Electrical Engineering and Computer Science, York University, Toronto, ON M3J 1P3, Canada; vlajic@cse.yorku.ca
* Correspondence: shadisa@cse.yorku.ca

Abstract: Recent research has revealed an alarming prevalence of click fraud in online advertising systems. In this article, we present a comprehensive study on the usage and impact of bots in performing click fraud in the realm of digital advertising. Specifically, we first provide an in-depth investigation of different known categories of Web-bots along with their malicious activities and associated threats. We then ask a series of questions to distinguish between the important behavioral characteristics of bots versus humans in conducting click fraud within modern-day ad platforms. Subsequently, we provide an overview of the current detection and threat mitigation strategies pertaining to click fraud as discussed in the literature, and we categorize the surveyed techniques based on which specific actors within a digital advertising system are most likely to deploy them. We also offer insights into some of the best-known real-world click bots and their respective ad fraud campaigns observed to date. According to our knowledge, this paper is the most comprehensive research study of its kind, as it examines the problem of click fraud both from a theoretical as well as practical perspective.

Keywords: digital advertising fraud; Web-bots; click bot

check for updates

Citation: Sadeghpour, S.; Vlajic, N. Click Fraud in Digital Advertising: A Comprehensive Survey. *Computers* **2021**, *10*, 164. https://doi.org/10.3390/computers10120164

Academic Editor: Wenbing Zhao

Received: 28 October 2021
Accepted: 29 November 2021
Published: 1 December 2021

1. Introduction

Over the past decade, online digital advertising has become the most prominent and profitable form of marketing for many businesses [1]. Unfortunately, in addition to unparalleled benefits and opportunities, the rise of digital advertising has brought with it a number of challenges, the most significant of them being advertising fraud (i.e., different actions involving manipulation of views, clicks, pages and other metrics for the purpose of deception [1]). The actual harm caused by advertising fraud is often hard to quantify, but it is estimated that globally this loss is likely in the order of billions of dollars [1].

Click fraud is a specific subcategory of advertising fraud that is carried out by producing fake clicks on an advertisement in order to generate illegal profits. Fraudulent clicks damage the health of online businesses by depleting their advertising/marketing budget while increasing the profit of the fraudulent click operators (i.e., fraudsters/criminals). They achieve this by clicking on advertisements with no real interest in the content. Click fraud can be conducted in two main ways: (1) hiring a group of people to increase fraudulent traffic, and (2) deploying automatic clicks/click bots. Shreds of evidence show that using a click bot to launch a click fraud attack is much more effective and thus far more common than the manual type of attack involving humans [2]. Bots account for approximately 45% of all Web traffic, with most of this traffic being related to click fraud [3]. In a number of recent studies, enterprises have reported that up to 70–90% of the clicks they receive are, in fact, generated bots. Cavazos in [3] estimated that globally, by the end of 2020, the total PPC (cost-per-click (CPC), also called pay-per-click (PPC), is a model in digital advertising in which the advertiser pays a fee to the publisher every time one of their ads is clicked on) invalid click loss would reach USD 23.786 billion. The prediction for the total invalid click loss for the USA alone is USD 9.06 billion. Motivated by these considerations, the primary purpose of this study is to bring awareness to the importance and challenges of combating click fraud in the online digital advertising ecosystem and provide a broad investigation of click fraud threats in online advertising. In particular, the main contributions of this work are as follows:

- First, we systematically look into the classification of different forms of click fraud as well as different types of click bots used to commit click fraud.
- We investigate the critical behavioral characteristics of bots versus humans in conducting click fraud within modern ad platforms.
- To date, several cutting-edge solutions have been proposed to address the problem of click fraud in online advertising. This article provides a thorough overview of the proposed methods and technologies in detecting and preventing click fraud discussed in the literature. We specifically categorize the surveyed techniques based on which particular actors within a digital advertising system are most likely to deploy them.
- We outline some of the most famous real-world click bots and their respective ad fraud campaigns observed to date.
- Finally, we conclude the article by highlighting some open challenges and future research directions in this field.

The remainder of this article is structured as follows: In Section 2, we review some essential background concepts, including the general characteristic of Web-bots, the different generations of Web-bots, typical online advertising services and the anatomy of ad clicks. In Section 3, we specifically discuss different types of click fraud conducted by the means of Web-bots. In Section 4, we explain several existing approaches to combating click fraud in the online advertising system. In Section 5, we present some of the most famous real-world click bots and ad fraud malware campaigns. Finally, in Section 6, we conclude the article and outline several research directions considering the emerging security requirement for protecting online advertising from click fraud threats.

2. Background

In order to understand the actual challenges of dealing with and defending against click bots, it is important for the reader to be familiarized with the history and state-of-the-art in this field. Therefore, in this chapter we provide a brief overview of progression in Web-bot development, and we discuss different known categories of Web-bots along with their malicious activities and associated threats.

2.1. Characteristics of Web-Bots

The history of Web-bots can be traced back to 1988, and to the advent of Internet Relay Chat (IRC) bots such as those used within the Hunt the Wumpus game platform or Bill Wisner's Bartender bot [4]. These early IRC bots provided automated services to users and sat in channels to prevent servers from shutting down due to user inactivity. It was not until 1994 that the first Web crawlers (i.e., Web-bots) were created. The first such bot (used to index webpages) was created by AOL (America Online) in 1995 and purchased by Excite in 1997. Soon after, several commercial web crawlers became available such as Lycos, Infoseek, Excite, AltaVista, and HotBot. [4,5].

According to Barracuda research conducted over the first six months of 2021, bad bot traffic accounts for nearly 40% of Internet traffic [6]. This trend is expected to continue, and as Cisco predicts [7], automated (i.e., bot generated) traffic will increase 37% year after year by 2022. However, these predictions not only suggest an increase in the overall volume, but also an increase in the diversity and enhancement of the automated Internet/Web traffic due to a number of factors, including: new paradigms for interacting with the Internet/Web, more sophisticated business models and dependencies on data sources, development of new shopping methods and habits, an increase in the complexity of criminal activity, and availability of cloud-based computing capacity, etc. As a result of the ever-growing volume and diversity, Internet/Web traffic is becoming increasingly difficult for website owners to manage and analyze [5].

There are several different definitions of what can be classified as an Internet/Web-bot. For example, Radware states that "bots are automated programs created to execute repetitive tasks”, Wikipedia says “bot is a software application that runs automated tasks (scripts) over the Internet”, and Netaces describes automated traffic as “any set of legitimate requests made to a website that is made by an automated process rather than triggered by a direct human action”.

It should be stressed that the term “Internet bots” refers to a broad family of malicious programs that target layer 3 (network layer), layer 4 (transport layer), or layer 7 (application layer) of the Open Systems Interconnect (OSI) model. However, the application layer bots are the only bots explicitly capable of mimicking human behavior, and as such are the main focus of this study. For that reason, in the remainder of this paper, we use the terms ‘application-layer bots’ and ‘Web-bots’ interchangeably.

Although Web-bots have been in existence for several decades, modern bots are a complex collection. Some are very useful; others are inherently dangerous. In the literature [8], Web-bots are generally grouped into good bots and bad bots, and are described as follows:

1. Good Web-bots are legitimate bots whose activities might be beneficial to businesses as well as individuals. They crawl websites to help with search engine optimization (SEO) aggregation and market analysis. Some specific sub-categories of good Web-bots and the functions they perform are listed below:
   - Web site monitoring bots monitor websites’ availability and system health. An example of a bot in this category is Pingdom.
   - Aggregator bots collect information from websites and notify users or subscribers about news or events. An example of this type of bot is Feedly.
   - Backlink checker bots confirm the inbound (referrer) URLs that a website receives so that marketers and SEOs can understand trends and optimize their pages accordingly. SEMRushBot is an example of this type of bot.
   - Partner bots execute tasks and functions on transactional websites. An example being PayPal IPN.
   - Social networking bots are deployed by social networking platforms, to add visibility to their webpages and drive the overall user engagement. Facebook bots are an example of this type of bot.
   - Search engine bots, which are also known as Web crawlers or spiders, crawl through websites in order to index their pages and make them available/accessible on the respective search engine. Without them, most online businesses would struggle to define their brand value and attract new customers. Bots in this category include: GoogleBot, Bingbot, and Baidu Spider.
2. Bad Web-bots are programmed to perform various malicious tasks on the WWW. They work evasively and are mainly used by scammers, cybercriminals, and other nefarious parties involved in a variety of illegal activities. Bad bots are automated programs that do not follow any rules. Mostly unregulated, they have a specific malicious objective which they are trying to accomplish. Some general sub-categories of bad Web-bots are [8,9]:
   - Scraper bots collect/steal large amounts of information from websites. They are scripted to look for specific data, including product reviews, breaking news, prices, customer names, product catalogues, or even user-generated content on community forums. By scraping the content off a website and then posting it somewhere else, bots can negatively affect the search engine’s ranking of this websites and/or the products it advertises. By scraping and posting content elsewhere, bots can also have a negative impact on the companies that invest budget and resources into creating original digital content.
   - Scalper bots are designed to automatically capture and purchase goods and have a high-speed checkout process. They make bulk purchases. For example, they buy hundreds of tickets immediately after opening of a booking and then sell them through reseller websites for a price considerably higher than the initial

ticket price. It is very common for scalper bots to mimic human behavior in order to avoid detection.

- Spam bots (also known as content spammers) inject messages into the user-controlled areas of a website, such as forums, guestbooks, bulletin boards, and reviews or comments sections associated with news articles. They arrive in the middle of users' conversation and insert messages with unwanted advertisements, links, and banners. This insertion frustrates real users who participate in forums and comment on blog posts. For example, spam bots may insert malicious links to direct users to phishing sites in order to trick them into revealing sensitive information such as bank account numbers and passwords.

Web-bots can regularly make requests from real browsers and execute JavaScript code intended to validate users as humans. Sophisticated Web-bots can bypass modern detection mechanisms such as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) either by using artificial intelligence, bruce-force systems, or with the help of human agent farms.

A botnet is a large network of bots (i.e., malicious programs), each running on a compromised computer while being controlled by a remote command-and-control center (the botnet operator). Commonly, the word botnet creates an image of a distributed denial-of-service (DDoS) attack. However, in reality, botnets can carry different payloads and can be used in different types of attacks. For example, they can be used to extract cryptocurrency from infected devices, or to cover up other attacks or an illegal activity. Some bots (i.e., botnets) can be utilized as email relays for massive spam campaigns. Ultimately, the threats emerging from botnets are only limited by the creativity of their creators [10].

It is important to emphasize that both types of Web-bots—good bots and bad bots—leverage the same HTTP protocol. This means that the techniques and approaches used to support Web-bot communication and service may be the same for both categories of bots.

2.2. An Overview of Different Generations of Web-Bots

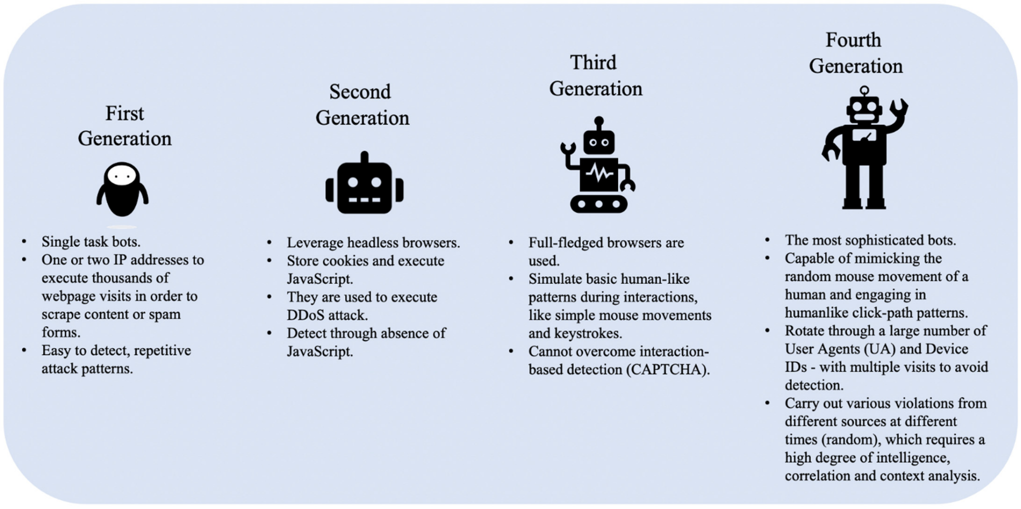
Web-bots have evolved rapidly from their origins as simple scripting tools with command-line interfaces to modern-day, complex programs that leverage full-fledged browsers and are able to mimic human behavior (e.g., navigate a website or application, move the mouse, touch and drag objects, etc.).

In this section, we provide a more detailed overview of four generations of Web-bots (as commonly classified by the cyber security community), all of which can still be found in use today [8].

- First-Generation Bots are basic scripts that send requests such as cURL (Client for URLs (or cURL) is a command-line tool for getting or sending files using URL syntax.) from a small number of IP addresses. These bots cannot store cookies or run JavaScript code (i.e., they do not have real Web browser functionality) and can be easily detected and mitigated by blacklisting their IP addresses and UAs, as well as combinations of IPs and UAs. They are mostly used for scraping, carding, and spamming.
- Second-Generation Bots leverage headless browsers (such as PhantomJS), and unlike first-generation bots can store cookies and execute JavaScript code to automate control of a website. These bots are used to conduct DDoS attacks, scraping and spamming campaigns, as well as to skew Web analytics or conduct ad fraud. However, they can be effectively detected using their browser and device characteristics including the presence of certain JavaScript variables, frame forgery, sessions, and cookies. Once identified, these bots can subsequently be blocked based on their fingerprints. Another way of detecting these bots is by analyzing their click-path through the target website as they often exhibit significant discrepancies relative to the click-path of ordinary (human) users/visitors.
- Third-Generation Bots can operate in full browser mode and are capable of executing human-like interactions such as simple mouse movements and keystrokes. However, they are typically unable to exhibit human randomness in their behavior. They are

commonly used to execute DDoS attacks, API abuse, ad fraud, and account takeover fraud. An interaction-based user behavioral analysis approach could help in identifying these bots as they generally follow a programmatic sequence of URL traversals.

- Fourth-Generation Bots, the most advanced category of bad bots, are capable of mimicking the mouse movements of a human and engaging in humanlike click-path patterns. These bots can change their UA as they cycle through thousands of IP addresses. The developers of this category of bots engage in behavioral hijacking (i.e., recording real users touching and swiping behaviors on hijacked mobile apps or websites) to fully simulate human behavior on websites or apps. This makes the process of detecting these bots challenging because their activity is not easily distinguished from real users, and they are usually distributed across tens of thousands of IP addresses. They are typically employed in ad fraud, account takeover, API abuse, and DDoS attacks. Figure 1 shows the key behaviors of bad bots by generation.



First Generation
- Single task bots.
- One or two IP addresses to execute thousands of webpage visits in order to scrape content or spam forms.
- Easy to detect, repetitive attack patterns.

Second Generation
- Leverage headless browsers.
- Store cookies and execute JavaScript.
- They are used to execute DDoS attack.
- Detect through absence of JavaScript.

Third Generation
- Full-fledged browsers are used.
- Simulate basic human-like patterns during interactions, like simple mouse movements and keystrokes.
- Cannot overcome interaction-based detection (CAPTCHA).

Fourth Generation
- The most sophisticated bots.
- Capable of mimicking the random mouse movement of a human and engaging in humanlike click-path patterns.
- Rotate through a large number of User Agents (UA) and Device IDs - with multiple visits to avoid detection.
- Carry out various violations from different sources at different times (random), which requires a high degree of intelligence, correlation and context analysis.

Figure 1. The evolution of bots.

2.3. *Impact of Bad Bots on Various Business Functions and Industries*

There are a wide variety of activities involving automated traffic that can be used to exploit businesses across all industries. NETACEA [9] believes that, regardless of the industry targeted with these attacks, the core of each bot attack is due to one of the following three motives: money, data, and stock.

2.3.1. Impact on Different Business Functions

We categorized the main types of bot attack techniques that are used to exploit business logic under seven broad categories [5,8,9]:

- Web scraping (Scraping of pricing, content and inventory information): This is a technique of extracting different types of information from websites, such as product prices and news content, which can be costly if extracted without consent. For example, nefarious competitors scrape prices and product lists to attract the other business' customers. They effortlessly steal whatever pieces of content they are programmed to find in order to sabotage the (victim) retailer's sources of income. Attackers also scrape unique content (and duplicate exclusive content) of an online business to negatively impact their search engine optimization (SEO) efforts.
- Cart Abandonment and Inventory Exhaustion: Merchants usually leave items in the shopping cart for about 10 to 15 min before concluding that the buyer has abandoned the purchase. After this period, the items are released and placed back into the available inventory. Competitors' bots put hundreds of items in shopping carts and abandon them

later to limit real consumers from buying products. That sets the grounds for a decline in sales, distorted conversion rates, and ultimately a damaged brand reputation.

- Application DDoS: These types of attacks look for functionality areas that are 'weak points' of the target application. This can be an area that involves high CPU usage, integration with third-party systems, or complex database activity such as search, registration, availability checking, or real-time booking requests. The bots automate their requests to those areas of the website until the website reaches its limit and fails or is unable to carry out normal transactions with legitimate customers. These attacks specialize in utilizing rotating IP addresses and legitimate user agents (to conceal the bots' identities) and are usually launched via large botnets.
- Scalping Products and Tickets: Malicious bots can be programmed to actively buy valuables goods such as consumer electronics and resell them for a considerably higher price. Bots can pick up tickets for popular events as soon as they go on sale.
- Card Cracking: Fraudsters use bots to test thousands of stolen credit card numbers against merchant payment processing. Since the stolen card owner can report a fraudulent transaction and request a repayment, the sites targeted with card cracking attacks will ultimately suffer financial losses (due to issued refunds), legal penalties, and lousy trading history. In extreme cases, frequent carding activities and too many refunds may force the merchant to disable credit card payments altogether.
- Fake Account Creation: Criminals use bots to create fake accounts and commit various forms of cybercrime. Some of the activities that can be carried out after creating such accounts include: misusing the 'first-time-buyer' bonus, using a free product trial awarded to a new account, using multiple accounts to attack the inventory of websites that only allow logged-in users to store items, content spamming, money laundering, malware distribution, and skewed research and SEO.
- Account takeover (ATO): Account takeover bots focus on gaining control over user accounts within a system and accessing people's personal data for use elsewhere. Credential stuffing and card cracking/credential cracking are amongst the commonly used ATO techniques, and each uses automated bots to gain brute force entry to an account. In the credential cracking attack model, multiple username and password combinations are attempted until a successful combination is discovered. Credential stuffing as an alternative approach involves taking known lists of email and password combinations and determining if they are further valid for alternative sites. After the credentials are authenticated, attackers can extract money or other financially valuable items (e.g., loyalty rewards) from within that account. They can also harvest personal data for use/sale elsewhere.

2.3.2. Impact on Different Industries

The bad bot problem affects every industry, and some particular types of bots are designed to attack a specific industry. The top 5 industries with the greatest amount of bad bots traffic include finance (47.7%), education (45.7%), IT and services (45.1%), e-commerce (39.8%), and government (37.5%) [7]. This section reviews some of the negative impacts that bad bots have on certain industries.

- Threat in Finance: Banks, financial service providers, and insurance companies are counted as high-value targets for fraudsters. In recent years, botnet attacks have progressively ramped up the rate and extent of fraud in these industries. The types of botnet attacks on financial institutions include: account takeover, DDoS attacks, and content scraping. However, credential stuffing and card cracking are the two most common techniques used by attackers in the financial services domain [8].
- Threat in Education: Malicious bots can be employed to look for research papers, class availability, and access user accounts in educational institutions. Recently, educational institutions have become a major target of DDoS attacks as more schools rely on distance learning to stop the spread of COVID-19. Researchers also reported an increase in phishing pages and emails, as well as threats that are posing as online

learning platforms and apps. According to the recent study in [11], the number of DdoS attacks on educational resources increased 550% in January 2020 compared with January 2019.

- Threat in IT and Services: Malicious bot attacks are capable of freezing inventory, crashing customer service, suspending orders, and crippling the operation of IT systems. They may not only stop businesses from generating revenue but also cause their complete closure.
- Threat in E-commerce (Web shops, Marketplaces, Classified portals, Aggregators): Web shop: A company has a product that it sells to customers through its own website. Marketplaces: They do not sell the product or service themselves but rather provide a platform for sellers and buyers to make online transactions. Classifieds portals: In this method, sellers list their products or services on a portal, and potential buyers contract the sellers through the portal. Aggregators: they are portals that crawl the Web and collect information on the same products and compare prices across Web shops; in fact, they aggregate information [12]. E-commerce companies receive a wide range of bad bot attacks. Malicious bots sent to third parties by competitors can crawl/collect information from these websites to post them elsewhere or even (re)sell them. Malicious bots can not only steal new listings, they can also fill Web forms with bogus details. In general, their activities include price and content scraping, account takeovers, credit card fraud, and gift card abuse [13].
- Threat in Government: When it comes to bad bot attacks, Governments are generally concerned with protecting company registration lists from being deleted by bots and eliminating election bots from tampering with voter registration accounts.
- Threat in Travel: Card cracking in the travel industry results in the theft of valuable and monetizable frequent flyer miles that are subsequently sold for a profit. Bad aggregators plague travel sites for travel lists, prices, and trends that can be used to inform and offer competitive package deals. Furthermore, inventory denials are frequently practiced at airlines, though this method is used throughout the tourism industry. In the airline sector, bots are employed to reserve seats on flights for up to 20 min (until they are paid for). During this time, genuine customers are shown that there is no availability on flights. The perpetrators then try to sell these seats for a profit.
- Threat in Gambling and Gaming: Account take over and credential stuffing are the two most common techniques that gambling and gaming companies suffer from because each account contains cash or loyalty points that can easily be transferred to other users and emptied if compromised.
- Threat in Digital Advertising: Ad fraud is known as a multi-billion dollar industry that uses very sophisticated methods to ensure that the maximum value is extracted. Fraudsters use botnets to generate fake clicks and obtain fraudulent digital ad impressions. Fake traffic artificially increases advertising costs. Malicious automated traffic also performs retargeting fraud to illegally generate revenue from invalid traffic to publishing sites. Such attacks sabotage the advertising network's efforts to connect them to quality inventory. It also prevents marketers from reaching a wider audience. Bad bots generate invalid traffic, which negatively affects the brand reputation of an advertising network and undermines its claim to provide reliable media for a media buying environment. Over and above that, skewing of analytics and other metrics by bad bots would result in invalid business decisions and a large amount of marketing and advertising expenditure being squandered, often in a matter of hours [5].

2.3.3. Terminology

This section reviews some essential terminology related to online advertising that we deploy in the remainder of this paper.

Online advertising ecosystem has three players: advertisers who desire to advertise a product or service, publishers who run websites, mobile apps and games that display the ads and ad networks such as Google AdSense, Bing Ads, and Yahoo that connect the advertisers with the publishers. A paid and displayed ad should ideally result in a click event. Cost-per-click (CPC), also called pay-per-click (PPC), is a model in digital advertising in which the advertiser pays a fee to the publisher every time one of their ads is clicked on.

As a type of advertising fraud, click-spam occurs when a fraudster executes clicks on behalf of users who have no interest/intention to click on an ad. It involves clicks through dedicated click-spam malware, accidental clicks, and clicks that the user was confused or tricked into clicking. Additionally, there are situations where the user simply intended to retrieve something else, but their click got hijacked into an ad click for something unrelated to their original query [14].

2.3.4. Typical Online Advertisement Services

Online advertising systems usually operate by placing a JavaScript snippet code on the publisher's website. In general, a publisher generates an "ad tag" (it may be an HTML or JavaScript code snippet code combined with a URL from which the browser will request an ad) via an ad server and places it on its website. Every time a user visits this website and downloads an ad from the ad server, the JavaScript code is executed (the ad tag sends a request to the ad server to show an ad in a given place). Each ad tag determines how the ad should be served on the website (including ad format, size, category, and other requirements). The snippet can also be put into the header or iframe (i.e., an inline frame used to embed another document within the current HTML document) wrapper to isolate it from the primary website script. This action (downloading the ad) causes the JavaScript code in the frame to be rewritten along with the HTML needed to display the ad; (shown in Figure 2, steps 1 and 2). Namely, the ad tag is sent after further to the advertiser's ad server. When the advertiser's ad server receives the ad tag, it ships a relevant ad creative to the publisher's ad server. Publishers rely on a pay-per-click model to count the number of times a user clicks on a link on the ad provider's server. They then charge the ad client/advertiser for the number of executed clicks (step 3). Finally, the user is directed to the advertiser's website (step 4) [2,15].

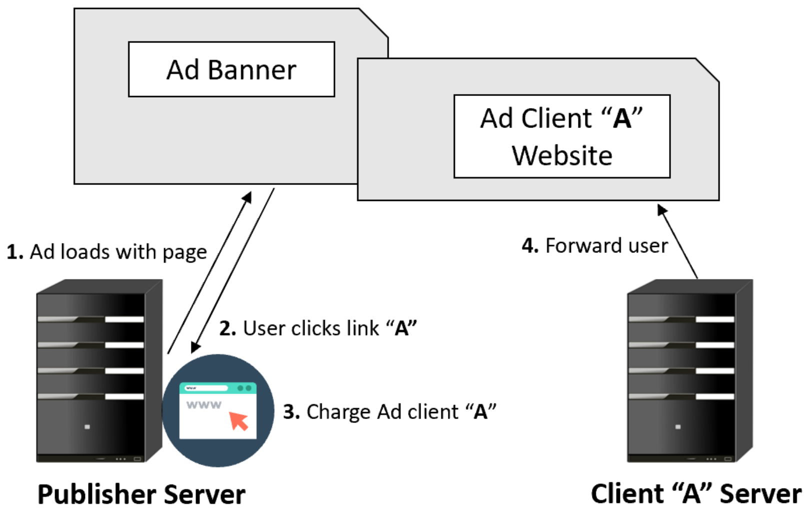


Figure 2. Typical online advertisement services.

2.3.5. The Anatomy of a Digital Ad Click

Figure 3 shows a more detailed anatomy of a digital ad click [14]. Ad networks provide libraries that enable publishers to host ads. (It is previously mentioned that these can be JavaScript snippets, which publishers embed in their website or application, or they can be server-side code, such as PHP or Java). As shown in step 3 of Figure 3, the JavaScript code (which runs in the user's browser) or the PHP/Java code (which runs on the publisher's Web server—not displayed) communicates with the ad network server in order to obtain a set of ads which get displayed on the publisher's website/application. Additionally, this code identifies the publisher when it connects to the ad network server, so that the given request can be correctly associated with the right publisher. This process is known as a creation/logging of an ad impression (step 4). Each ad returned (step 5) includes a unique identifier used to track clicks on the given ad. When a user clicks on the ad (step 6), the user's browser sends an HTTP request to the ad network, including a unique identifier for this ad impression (step 7), which gets registered in the ad click logs (step 8). In addition to the unique ad identifiers, the ad click logs also store the information about the advertiser being billed (and how much), the publisher who was paid, the user who initially retrieved the ad, etc. In most cases, the HTTP response to the above request redirects the browser to the advertiser's website (mostly using an HTTP response code 302; steps 9 to 11 of Figure 3). Beyond that point, ad networks are unable to track user activity on advertisers' websites. However, advertisers can embed JavaScript code provided by ad networks on certain pages (e.g., payment confirmation page) to notify the ad network if/when the user has taken an action (step 15). Such an event—in which the user has executed a desired action (e.g., a purchase) following an ad—is known as ad conversion. Ad networks use cookies to link conversion events to unique ad impressions and click IDs (step 16). The final conversion events can occur hours or days after the first click on the ad. Ad networks use conversion signals to provide bulk discounts to advertisers according to Smart Pricing Algorithms [16]. The assumption behind Smart Pricing is that traffic is useless to advertisers if the publisher sends traffic that does not lead to the desired action (e.g., purchase, email sign-up). The Smart Pricing Algorithm calculates penalties for publishers. The less traffic that is converted for this publisher, the higher the publisher penalty score and the more discounts all advertisers receive for clicks on their ads when displayed on that publisher's website, which ultimately implies that less money is paid to that publisher.

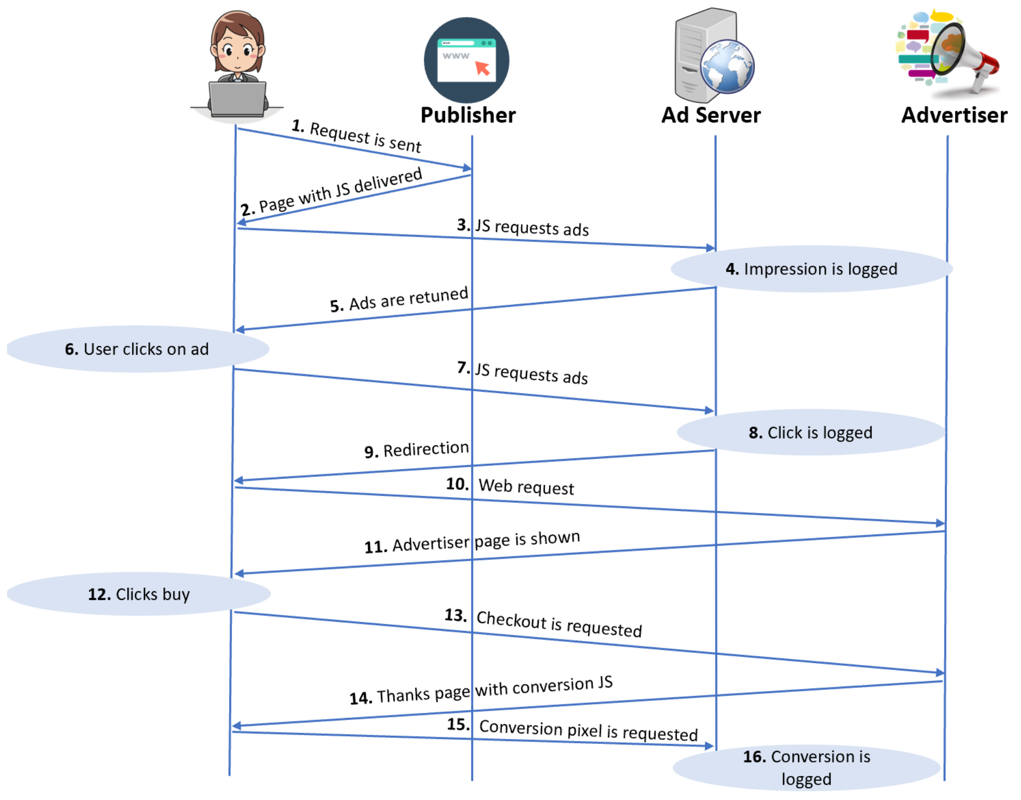


Figure 3. The anatomy of a digital ad clicks.

3. Taxonomy of Click-Based Fraud

The ultimate goal of advertising is to deliver and display relevant advertisements to users who have a genuine interest in a product or service. Thus, one of the main questions in dealing with (i.e., preventing) click fraud is: what can distinguish genuine human click behavior from the behavior of click bots? It is challenging to derive a simple and deterministic rule for addressing this problem, especially as bots continue to evolve so as to better mimic the behavior of humans. Therefore, the need for an extensive study on this topic seems necessary, similar to the one conducted to analyze the behavior of bots and humans on Twitter [17].

Above all, it is worth mentioning that genuine users' clicks are generated if users are truly interested in the advertised products. Furthermore, a real user tends to read, consider, think, and surf the website to learn more about a product before actually purchasing it. Or, to put it another way, a real user first tends to focus on the 'context' surrounding the product before performing the final 'click' action. Consequently, we believe that paying enough attention on the overall behavior of a user 'prior' and 'post' a click event, while at the same time understanding the actual context pertaining to the click, is the most promising approach to distinguishing between genuine human clicks and those generated by click bots or click farms.

3.1. Click Fraud—What We Know

Click fraud (also known as malicious clicks or click spam) is the most common fraud in the digital advertising environment [2]. Click fraud is closely tied to the so-called cost-per-click pricing model, which is generally used to determine the costs of showing users ads on search engines, social media platforms, and other publishers. Click fraud is often performed to execute an attack against a competitor and ultimately increase its advertising expense. Xu et al. [18] explain that click fraud happens when a fraudster makes HTTP requests for the destination URLs found in shown ads. Click fraud can be conducted either by human clickers or click bots, with each group exhibiting unique characteristics. For example, Xu et al. [18] believe that human clickers have financial incentives to perform clicks on multiple ads in a short period of time; therefore, they can be distinguished from genuine users who typically tend to read, think, and surf the website before making a purchase. The research of Pooranian et al. [2] focuses on the idea there are many important differences between the behavior of automated click bots and human click fraudsters—the phenomenon of exploiting a group of real people to increase fraudulent traffic in online advertising is known as crowd fraud. For example, whereas human fraudsters may engage in crowd click fraud via their individual (and different) accounts and/or computers, automated fraudulent traffic can be generated from relatively few computers. Nonetheless, it is generally much more difficult to distinguish between normal and fraudulent traffic generated by real people than to distinguish between normal traffic generated by genuine users and noisy traffic generated by machines. Therefore, the methods used to detect automatic click fraud are not very effective in identifying man-made fraud. It is worth noting that this type of fraud is not a subject/focus of this study, since according to many research and industry reports the crowd click is not as prevalent as the click bots and does not scale well. It is much cheaper for hackers to use automated scripts to simulate a human clicking on ads than hiring human laborer to do so.

In the remainder of this section, we will explore the concept of 'automated click' in more depth.

3.2. Classification of Automated Click Fraud Attacks

In general, automated click fraud attacks can be classified into four groups [2,19]: Badvertising, Hit Shaving, Hit Inflation, and Botnet Click Campaigns (See Figure 4). Each type of attack is briefly discussed below:

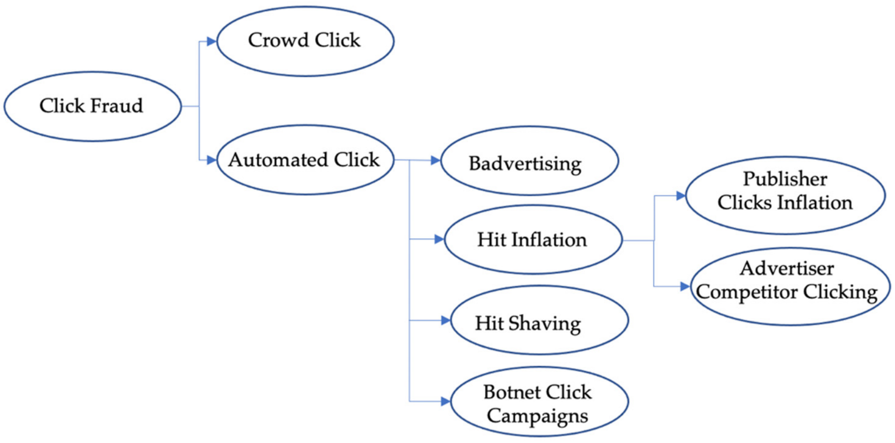


Figure 4. Categorization of click fraud attacks.

3.2.1. Badvertising

This type of attack works by synthetically—and stealthily—increasing click-through counts of ads hosted by a fraudster [20]. Badvertising is accomplished by corrupting the JavaScript file that is required to be downloaded and executed by a user's Web browser to publish sponsored ads. A part of its functionality, this JavaScript code traverses the source of the ad frame to find a list of legitimate ads and then simulate a fake click. Each click increases the number of registered visits for a sponsored ad, thereby accounting for higher revenue for the publisher. Recall from Section 2.3.4, in online ad platforms, a JavaScript snippet is embedded into a publisher's webpage. The JavaScript snippet gets executed whenever a user clicks on an ad and downloads the advertisement from the ad server. As previously indicated, badvertisements contain extra malicious scripts that simulate clicks automatically (See Figure 5). After the advertiser's JavaScript code runs and rewrites the page to include the ads, the badvertisement JavaScript parses the resulting HTML and compiles all links on the webpage. It then modifies the page to embed an HTML iframe (i.e., inline frame-designed to allow a page to seamlessly draw content from other pages). If the user decides to click the link, the iframe will be activated in the background and load its content to exploit the user.

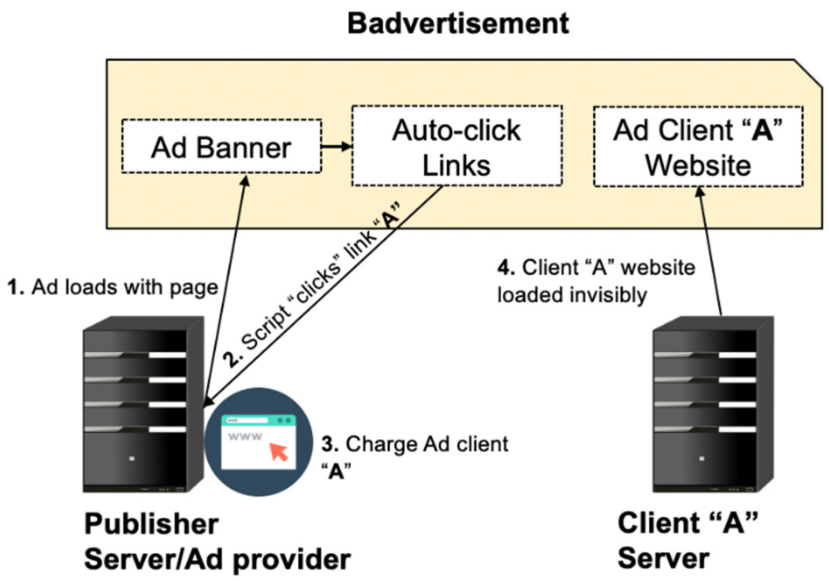


Figure 5. Automatic clicks on hidden badvertisment. Compared with Figure 2, the ad banner displayed here is hidden. JavaScript code extracts links from the hidden ad banner and makes it appear in another hidden iframe, creating the impression that the user has clicked on the links.

3.2.2. Hit Inflation

This type of fraudulent activity [21] is employed by an adversary as a way to increase the hit count with the intention of earning unjustified revenue from the traffic driven to the advertisers' websites, or (in some cases) with the intention of simply hurting the competitors. In particular, publishers may inherently have an interest in committing click fraud as they are rewarded based on the number of executed clicks on the ads they display to their audiences. In addition to malicious publishers, competing advertisers may also be interested in committing click fraud. Namely, most advertising campaigns have limited budgets, and each fake click consumes a small portion of that budget. Thus, an advertiser can financially hurt its competitors by generating a large number of fake clicks on the competitors' ads. These types of fraud are called publisher clicks inflation and advertiser competitor clicking, respectively [2].

One of the most basic attacks is conducted by a publisher who repeatedly clicks on the ad hosted by the publisher himself [22]. However, this type of attack can easily be blocked by removing duplicate clicks generated by the same visitor in a short period time. A dishonest publisher may also use a script to indirectly make every visitor automatically click ads. Mayer et al. [23] discuss different methods that a publisher can deploy to simulate clicks. A scenario of a sophisticated type of hit inflation attack consisting of a fraudulent publisher XYZ.com, a fraudulent website ABC.com, an advertiser AD, and a user U, is illustrated in Figure 6. The scenario begins when the user requests or clicks to fetch the ABC.html page from ABC, where the fraudulent website ABC has a script code to redirect the user to XYZ silently. Therefore, the user will be redirected to XYZ.com, where XYZ.com keeps two forms of webpages: a valid/original page and a manipulated version. XYZ.com will show the manipulated page to the user if the referrer field in the HTTP request points to ABC.com. Then, XYZ.com triggers to click the ads on its own (One possible implementation of simulated clicks [23,24] on a link can be done using JavaScript (see [25,26] for more details)) without the user's knowledge. Otherwise, XYZ.com will direct the user to the valid webpage and allow him/her to decide whether or not to click on the ads. Another method of attack involves publishers creating fake visitors and inflating the number of clicks by forging their IDs [27].

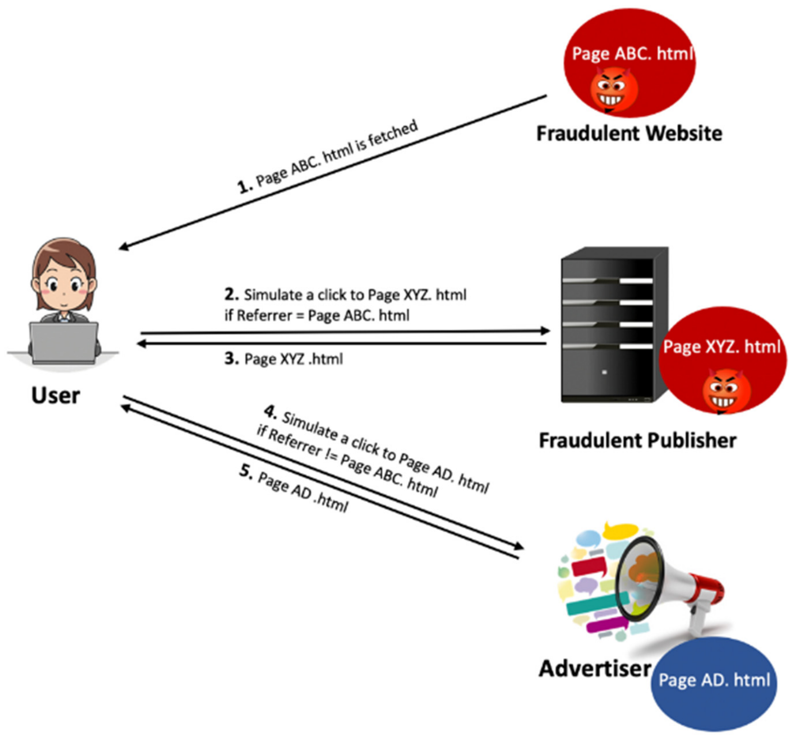
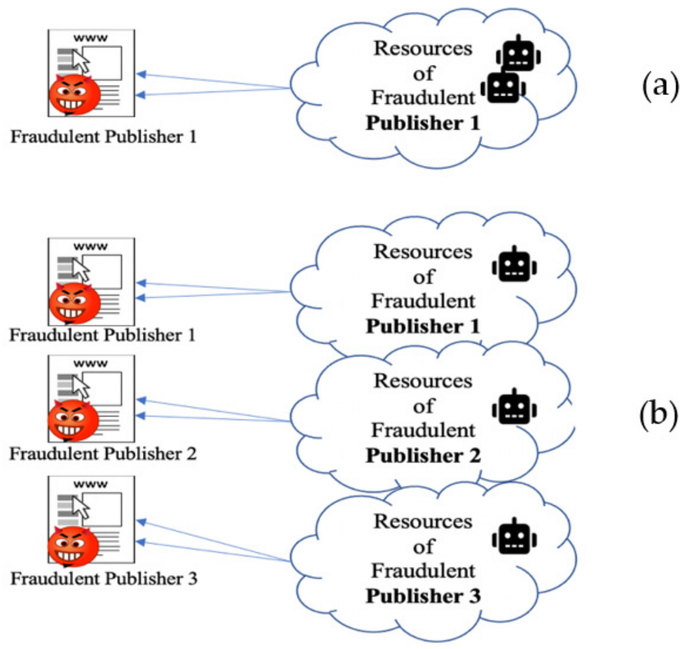

Page ABC. html
Fraudulent Website
1. Page ABC. html is fetched
User
2. Simulate a click to Page XYZ. html
if Referrer = Page ABC. html
3. Page XYZ .html
Page XYZ. html
Fraudulent Publisher
4. Simulate a click to Page AD. html
if Referrer != Page ABC. html
5. Page AD .html
Advertiser
Page AD. html

Figure 6. Hit inflation attack scenario.

The attack techniques mentioned above are operated by a single fraudster and are known as non-coalition attacks. However, fraudsters often form groups and launch the so-called coalition attacks by sharing their resources (i.e., machines or IP addresses). By joining a coalition, a fraudster expects more profit and less probability to be detected [2,21,22]. Namely, sharing resources has greater beneficial outcomes for fraudsters, as it diminishes the cost of launching the attack. Moreover, in case of coalition attacks, it becomes more challenging to identify the relationship between each fraudster and the attacking machines, such as IP addresses and cookie IDs of the fraudsters' sites and the resources generating traffic. Figure 7 illustrates coalition and non-coalition attacks.



Fraudulent Publisher 1
Resources of Fraudulent Publisher 1
(a)
Fraudulent Publisher 1
Resources of Fraudulent Publisher 1
Fraudulent Publisher 2
Resources of Fraudulent Publisher 2
(b)
Fraudulent Publisher 3
Resources of Fraudulent Publisher 3

Figure 7. Non-coalition and coalition attacks: (**a**) attack by a single publisher, (**b**) attack by a coalition.

It should be noted, though, that non-coalition attacks could be further classified according to the numbers of IPs and the system's cookie ID. Namely, when legitimate Internet customers visit a publisher's site, the generated traffic has specific characteristics different from fraudulent traffic and typically implies a unique relationship between the site and the customers IPs and cookie IDs. However, given that fraudsters do not control a large portion of surfers' machines, their attack traffic will leave a different fingerprint (i.e., correlation) between the IPs and cookie IDs. A sign of imbalance in these relationships is what fraud investigators look for to detect the fraud. Nevertheless, in order to avoid detection, a non-coalition fraudster may deploy different strategies to generate/induce a false correlation between the IPs and cookie IDs. Six such strategies have been identified in practice and are enlisted below [21]:

- Cookieless Attacks. Dishonest publishers employ two techniques to launch cookieless attacks. They either disable the cookies on the machine(s) they use to attack or utilize network anonymization services that are commercially available and generally used to protect users' privacy. Network anonymization is designed to protect users' privacy, and therefore, they block third-party cookies. As a result, such publishers will have a high percentage of cookieless traffic. This can be detected by tracking the percentage of cookieless traffic for each publisher and examining publishers that deviate from the norm.
- Single Cookie and Single IP Address Attacks. The fraudster (i.e., fraudulent publisher) can run a simple script to launch an attack from one machine with a fixed IP and one cookie ID.
- Single Cookie and Multiple IP Addresses Attacks. It is generally easier for a fraudulent publisher to manipulate the cookie than to change the IP address of the attacking

machine; therefore, this class of attack is not prevalent among attacking publishers but advertisers. Formally, in this attack, fraud investigators need to discover cookies that appear with more than p IPs in a specified period t to detect this type of attack.

- Multiple Cookies and Single IP Address Attacks. This category of attacks can be performed in many ways. For example, one simple although not the most economical way would be to utilize many scripts running on many machines connected to the Internet through one router. As such, this attack may resemble a normal traffic scenario, where many users with different cookie IDs are connected to the Internet with a single IP through a Network Address Translation (NAT) or ISP. In a more comprehensive and sophisticated way, the attacker can connect several machines to the Internet via an ISP with a similar IP. By combining fraudulent traffic with normal traffic, the fraudulent publisher can reduce the impact of the attack and confuse the fraud detection mechanisms.
- Multiple Cookies and Multiple IP Addresses Attacks. This attack is the most complicated type of attack in terms of execution and detection. Fraudsters must access several valid cookies and IPs to launch the attack. The simplest but not economically viable way is when fraudsters obtain access to various machines with multiple accounts residing on different ISPs. Another possibility is to acquire cookies and IP addresses of legitimate users via spyware and Trojans [28]. The traffic generated in this way is most likely to resemble normal/real traffic.

3.2.3. Hit Shaving

In online ad platforms, hit shaving [29,30] is a form of fraud that fraudulent advertisers use to undercount the real number of clicks received from a publisher in order to pay a lower commission fee. Before describing how the hit shaving attack is employed in an ad network, we need to provide an overview of the mechanisms used in click-through payment programs. Undoubtedly, electronic commerce has rapidly become recognized as an effective advertising tool. This allowed online advertising to become a key technology on the Internet, as confirmed by the growth of click-through payments. The main entities involved in click-through payment programs are (i) users who view the webpages and click on a link, (ii) the referrer who reveals the ads to the users, and (iii) the target site running the click-through payment process. When a user views webpage A (the referrer), for example, and clicks on a link that refers him/her to webpage B, then A should receive money from B for this reference (the user has clicked through A to reach B). Since this structure is based on the HTTP protocol (which is a stateless protocol), the structure inherits a number of vulnerabilities due to the lack of communication between the referrer and the target site after the user clicks on the link. For example, if the user retrieves page A. html from site A (the referrer site), and then clicks on the link and requests the page from site B (the target site) as a result, page B. html on the site B will be uploaded for the user. However, in this scenario, A cannot verify the number of times its webpage has referred users to B, whereas B can deliberately omit some of the click-through events coming from A, ultimately allowing 'hit shaving' fraud to take place. Even though in some cases, the referrer site may be able to identify such hit shaving, it cannot provide any evidence to the third party [2].

3.2.4. Botnet Click Campaigns

In this type of attack, a network of (typically compromised) machines connected to the Internet are infected by malware under the control of a botmaster/controller to visit various websites and click on their ads without the knowledge of the owners of those computers [19]. The botmaster orchestrates millions of click bots to perform automatic and large-scale click bot attacks.

Botnet click campaigns are the most effective way of hitting a high volume of pay-per-click ads that can be employed by several parties, such as competitors (advertiser competitor clicking) and webmasters (publisher clicks inflation—refer to Section 3.2.2) to commit click fraud. However, it is important to note that operating a botnet is not cheap, and operators must weigh the risks and costs of operating and maintaining a profitable botnet. This is generally done by a team of organized hackers/fraud rings [19]—groups of criminals who specifically target ad networks to exploit the maximum amount of money possible in a short term. Dozens of botnets have been discovered over the years, costing millions in wasted spend for marketers. In Section 5, we will describe some of the most famous real-world click bots and ad fraud malware campaigns and their respective techniques utilized to implement fake clicks.

In the remainder of this section, we will discuss—in detail—how a victim host conducts click fraud under the control of a botmaster.

Figure 8 illustrates the steps involved in a click fraud campaign conducted under the command of a botmaster. In the first step, the botmaster searches for vulnerable machines (they can be found via websites, servers and even Wi-Fi networks) in order to infect them with bot malware either by means of automated drive-by-download installation or by inciting them to download a Trojan horse program. Once compromised, these machines can be controlled by the botmaster through a command-and-control server (C&C) and eventually will get instructed to launch a click fraud attack. The attack instructions typically specify the type of ad to click on (on the website of the designated target publisher), the number of clicks to perform, the referrer to be used in the fabricated HTTP requests, and the duration and frequency of clicks [18]. After receiving the instruction, the compromised machine (i.e., the click bot) fetches the target website (step 2). The publisher's webserver returns the requested page and any ad tags (snippet of HTML or JavaScript code) that are within the page (step 3). For each ad tag, the click bot issues an HTTP request and sends it to the ad network to retrieve the ad content (step 4). The ad network responds to these requests by returning ads to the click bot (step 5). Of all the advertisements returned, click bot selects the ad that matches the specified search pattern and simulates a click on the ad, which triggers another HTTP request to the ad network (step 6). The ad network records this click traffic (which allows it to bill the advertiser and pay the publishers), and it also returns an HTTP 302 redirect response to the click bot (step 7). The click bot then follows the redirection URL and requests the advertised page (step 8). The advertiser returns the landing page (step 9), and at this point, the click bot performs click fraud. It should be noted that all the above-mentioned HTTP requests are made without the victim's knowledge, as the click bot works silently in the background to avoid being detected.

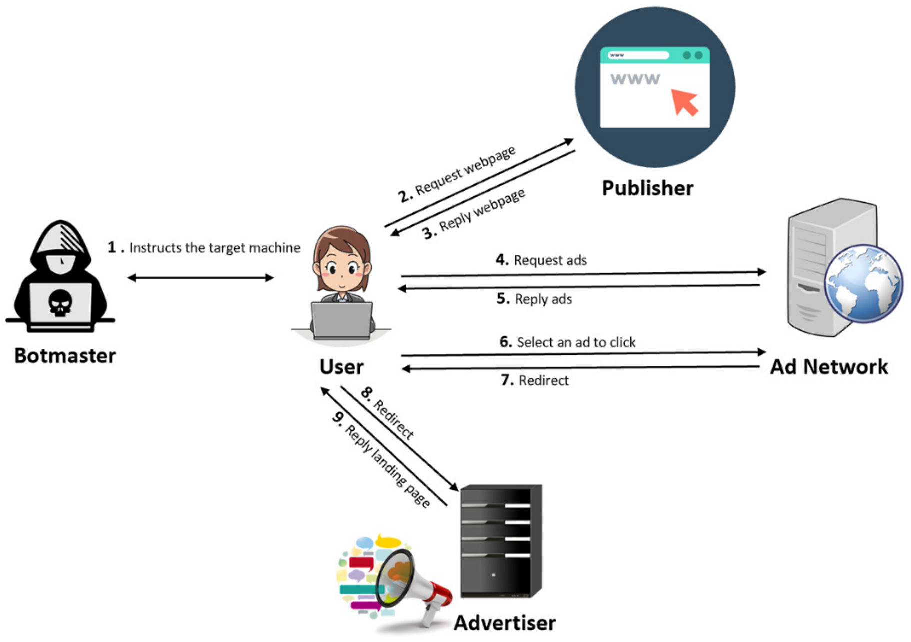


Figure 8. How a botmaster generates a click bot.

4. Comparison of Existing Detection Methods for Click Fraud

This section discusses the prominent existing solutions to detect, and combat click fraud in the online advertising system. A full list and summary of these techniques is also provided in Table 1.

Table 1. Summary of detection and prevention proposals against click fraud attacks.

Threat	Mitigation Strategy	Key Points	To Be Deployed By	Ref.
Competitors click fraud/Click farm/ Software click fraud/Background click fraud	CCFDP (Collaborative click fraud detection and prevention system), Log Analysis, Filtering-based Approach,	• A collaboration between server-side log and client-side log. • CCFDP architecture involves a Global Fraudulent Database (GFD) that stores the server-side log, client-side log, a fraud score report data, and monitored Web server site with filter program, such as ISAPI filter or CGI.	Publisher and Advertiser	[31] 2005
Click fraud	Badvertisment detection model using active and passive schemes	• Badvertisment attack was experimentally verified on several prominent advertisement schemes. • The proposed detection schemes fall into two basic categories—active schemes that attempt to seek out click fraud instances and passive schemes that watch for click fraud in progress.	Advertiser	[20] 2006
Click fraud /hit shaving	Cryptographic authentication	• It is an affirmative approach that only accepts legitimate clicks, namely those validated through client authentication. • Premium-click scheme relies on a foundation of cryptographic authentication to validate good clicks.	Advertiser	[32] 2007
Duplicate clicks pay-per-clickstreams	Decaying Window models including jumping Windows and sliding Windows	• Group Bloom Filter (GBF) algorithm was utilized to reduce memory operations when processing clickstreams. The GBF algorithm based on Group Bloom filters was effective and efficient over jumping windows. • Timing Bloom Filter (TBF) algorithm was introduced to detect duplicate clicks over sliding windows and jumping windows. It could process clickstreams over sliding windows using less memory space and processing time.	Advertisers	[33] 2008
Click bot	Bluff ads	• The Bluff ads are authentic ads of different advertisers, spread in the network and shown randomly, but never charged for. • Put some obstacles against the botnet's owner to train their software.	Publisher and Advertiser	[34] 2010
Click fraud	Multimodal Evidence Fusion by Dempster-Shafer Theory	• A multimodal collaborative click fraud detection and prevention (CCFDP) system that used three independent data mining modules to analyse click recording data.	Advertiser	[35] 2010

Table 1. *Cont.*

Threat	Mitigation Strategy	Key Points	To Be Deployed By	Ref.
Click fraud /deflation fraud	A hybrid of cryptography and probability tools	• The proposed model detects any z amount of deflation fraud with a success probability growing exponentially with z. • It allows the Web publisher to determine the expected transaction amount, which provides a sound basis to detect any massive magnitude fraud. • It is a user-friendly and straightforward model. • There is no need for a third party. • Constant ad communications, computation, and storage costs.	Publisher	[29] 2010
Click spam	Bayesian Approach	• A methodology designed for advertiser to independently measure and compare click-spams across ad network. • An automated framework for ad network to proactively fingerprint different simultaneous click-spam attacks.	Advertiser and Ad network	[36] 2012
Click bot	Bot signature	• A subset of methods for bot filtration includes: User Click Frequency, Presence of Cookie, IP Blacklist, Clusters of Users, User Ad Click Sequence Count, User Keyword Click Count.	Ad industry	[37] 2013
Click spam	ViceROI: Filtering-based approach	• ViceROI does not rely on security through obscurity and cannot easily be circumvented by click-spammers. • Click-spammers have higher ROI than ethical publishers. This higher ROI justifies the higher risk the click-spammer must carry, regardless of the specific mechanism the click-spammer is using to commit click-spam.	Ad network	[14] 2013
Crowd click and click bot	JavaScript support and mouse event test, Browser functionality test, Browser behavior examination	• Proactively tests if visiting clients are full-fledged modern browsers and passively scrutinize user engagement.	Advertiser	[18] 2014
Click bots	FCFraud—Random Forest classifier	• It is an operating system's anti-malware service. • It inspects and analyzes web requests and mouse events from all the user processes and applies Random Forest algorithm to automatically classify the ad requests. • It detects fraudulent ad clicks using a number of heuristics.	User	[38] 2016
Click bot	Rule-based model	• Online and offline detection rules. • The online process is responsible for what happen when an HTTP request is received in the server and passes the received request through various online rules. • The offline analysis process takes recorded requests from the database as input. The module takes the offline rules from the Rules module and runs the input through a series of tests from the rules.	Ad network	[39] 2018

Table 1. *Cont.*

Threat	Mitigation Strategy	Key Points	To Be Deployed By	Ref.
Click fraud	Multi-time-scale Time Series Analysis	• The proposed model forecasts click fraud behavior in terms of seconds. The proposed model consists of seven stages: pre-processing, data smoothing, fraudulent pattern identification, homogenizing variance, normalizing auto-correlation, developing the AR and MA models and fine tuning along with evaluation of the models.	Ad network	[40] 2019
Click fraud	Ensemble Learner	• Resampling using SMOTE and then performing the classification using Adaboost with Random Forest as base classifier produced the most improved results.	Advertiser	[41] 2019
Click fraud	Top-Rank-k frequent pattern mining	• A combination of frequent itemset mining algorithm and outlier detection algorithm was used to detect fraudulent clicks. SVM method was applied to make the detection result more accurate and has a wider detection range.	Ad industry	[42] 2019
Click fraud	LightGBM—a Gradient Boosting Decision Tree-type method	• LightGBM algorithm was applied over a public dataset including 200 million clicks over four days. • Feature engineering were used to improve detection performance.	Advertiser	[43] 2019
Click fraud	Traffic analysis	• Two click spam defenses were designed: the mimicry defence and the bait-click defence. • The Clicktok utilized Non-negative Matrix Factorization (NMF) algorithm to partition click traffic in order to identify fraudulent clicks.	Ad network	[44] 2019
Click fraud	CFXGB (Cascaded Forest and XGBoost), Feature transformation and classification.	• The model consists of three stages: pre-processing, feature transformation based on Cascaded Forest and XGBoost model classification. • The two parameters Max Layers and Early Stopping Rounds (ESR) were used to limit the number of cascades/layers added to the model. • The XGBoost model predicts whether a single click is fraudulent or not on all observations.	Ad industry	[19] 2020

There are limited research activities on click fraud detection and prevention methodologies in the scientific references before 2005. Some of the articles explain techniques which could be applied directly to the click fraud detection process, and the other are more general approaches introducing methods for Web log data analysis, which could be utilized in the click fraud systems under certain conditions [31].

In [35], Kantardzic et al. have introduced a multi-modal real-time detection and prevention system to address the problem of click fraud in the online ad system. The proposed model is based on Collaborative Click Fraud Detection and Prevention (CCFDP) system [31], which includes collaboration between client protocols and server protocols. Originally, CCFDP was designed to collect data about each click, by fusing data from the client-side and the server-side logs, in order to enhance the description of each click and obtain a better estimation of the click traffic quality. Kantardzic et al. have analyzed these extended click records using three independent data mining modules: rule-based module, outlier detection module, and click map module. The output of each of these modules is a probabilistic measure of evidence that a particular click is fraudulent. An overall score is assigned to each click based on the individual scores estimated by independent modules. Scores are combined using the Dempster-Shafer proof theory. The authors have examined version 1.0 (The initial version (version 1.0) of CCFDP was invented using a rule-based system, which closely matches most of the click fraud detection systems available today) and version 2.0 (The new CCFDP (version2.0) has an outlier module and the click map module, and an improved rule-based system with additional click context information) CCFDP systems with data from a real ad campaign. The results have shown that even the most popular search engines such as Google experience a high rate of click fraud.

In contrast to the heuristic filtering methods that have been used to eliminate fraud clicks, Juels et al. [32] have considered an opposite (i.e., affirmative) approach that focuses only on accepting legitimate clicks—that is, clicks that have been verified via client authentication. Their solution supports a new advertising model in which validated premium clicks have a higher value than casual clicks with less certain authenticity. Click verification in the proposed system is based on websites that have proof of user validity (which differentiates them from bots, scripts, or scammers). The premium click model is transparent to the users, and the process of authentication ensures user anonymity (i.e., respects users' privacy). The premium clicks model provides new, cryptographically authenticated visibility into click traffic, which results in an innovative, robust platform for combating click fraud.

Haddadi [34] proposed 'Bluff ads' as an online click-fraud detection strategy based on a predetermined threshold (If a website receives multiple clicks from the same IP address in a short time, these clicks could be flagged as fraudulent) of bluff vs. real ad clicks. 'Bluff ads' are defined as a set of irrelevant ads that should never be clicked on by a regular/benign user, even though they get displayed/mixed together with the relevant ads appearing on a webpage. Bluff adds may either contain entirely irrelevant display text, or in some cases they may contain text that is relevant to the content of the underlying webpage but have no targeting value for the visiting user (i.e., they are unrelated to this user's profile). The defensive potential of bluff ads lies in the fact that it is extremely difficult for botnet owners to train their automated software (i.e., click bots) to specifically avoid clicking on these ads. Additionally, bluff ads can work as a litmus test for the overall legitimacy of a user's clicks on the underlying host webpage and its targeted ads. Bluff ads could also be deployed as a technique to decrease the user's negative perceptions about ads and user tracking by reducing the number of accurately targeted ads that appear on a webpage they are visiting. As noted in [34], bluff ads can be used to detect click fraud committed by a publisher against its respective ad broker or a particular advertiser. Clearly, if used by an advertiser as a defense against a malicious publisher, bluff ads will require a larger advertising budget. However, bluff ads can also be used to detect click fraud against a publisher committed by a malicious third party.

Motivated by Haddadi's model, Dave et al. [36] have proposed a methodology that enables advertisers to independently measure and compare click spam ratios on their ads through the use of fake ads. They have also developed an automated method for ad networks to differentiate between various click spam attacks proactively. The primary idea of this approach is that the users associated with click-spam are not interested in the ads; therefore, they would be less likely to make any extra effort to reach the target website than a legitimately interested user. Thus, advertisers can measure this difference and apply it to estimate the click-spam fraction. The authors have also considered that some legitimately interested users might not make the extra effort, or some uninterested users may do some extra work. Therefore, they have improved the model for both false positive and false negative errors using the Bayesian technique and have performed several experiments relative to the control experiments. Through these experiments, they have validated the correctness of the proposed model and have conducted a large-scale measurement study involving ten major ad networks (including Google, Bing, AdMob, and Facebook) and several different types of ads (in particular, search and contextual advertising) on mobile and non-mobile devices. The results have concluded that click spam is a serious problem even for the largest ad network, and it is rampant in the mobile advertising context.

To catch click-spam in search ad networks, Dave et al. [14] introduced a solution called Viceroi. This solution is based on the intuition that click spam, in order to be a profitable business that offset the risk of getting caught, should offer a higher return on investment (ROI) for click-spammers than for other ethical/honest business entities. Viceroi is intended to be used in an ad network, where all ad clicks are visible/observed. Viceroi consists of two components: (i) an offline module that analyzes (previous) click logs on multiple timescales to identify spammers and anomalous regions in their user distribution revenue, and (ii) an online module that decides if an incoming click should be flagged as fraudulent (and if so, this click would be discounted at billing time). Given that publishers are the most common perpetrator of click spam, Viceroi looks for and ultimately flags all publishers with anomalously high ROI. The model has been evaluated on a large real-world ad network dataset. The results showed that the model was able to differentiate among six different classes of click-spam, including conversion fraud, ad injection, search hijacking, malware, arbitrage, and parked domains.

In 2014, Oentaryo et al. [45] organized a Fraud Detection in Mobile Advertising (FDMA) Competition to open a unique opportunity for participants working on real-world fraud data. The task given to the participants was to identify fraudulent publishers who generate illegitimate clicks and distinguish them from ethical publishers. The competition organizers believed that the competition results would provide a comprehensive study on the usability of data mining-based fraud detection approaches in a practical setting. Interestingly, the participants quickly turned in good results, and the performances continually improved toward the end of the competition, which showed the potential of various feature engineering and data mining methods. Below, we summarize the common methods used by the winning teams:

1. Some participants used spatial features and click traffic grouped by country, referral URL, channel, etc. Simple normalization was also often used to increase the effectiveness of the deployed classifiers. Feature transformation and scaling techniques (such as principal component analysis, PCA) were rarely practiced as they could not bring performance improvements, according to the participants' reports.
2. Some participants applied feature selection methods and reported that the wrapper methods (A wrapper algorithm provides a subset of features intended to improve the results of the specific predictors.) performed better than the filter methods. (A filtering method is a feature selecting function that is independent of predictors and filters the features that may not be useful in data analysis [46].)
3. Over and above that, ensembles of decision trees were the most widely used classification algorithms. The algorithms provided reasonably fast learning, and they were well suited to highly skewed class distribution and noisy nonlinear patterns.

The conclusion, derived from the obtained results, showed that ensemble methods combined with wrapper methods were the most promising approach for click fraud detection. However, Oentaryo et al. have suggested that in practice, single models would still be preferred, due to the reasons of better interpretability and tractability.

It has been noted in [41] that there are two types of machine learning techniques for detecting click fraud—those based on an individual classifier and those based on an ensemble of many separate classifiers. In this work [41], Kar et al. have proposed a novel ensemble-based learning mechanism for click fraud detection. To train their classifiers, the authors have used the real-world dataset provided by Fraud Detection in Mobile Advertising (FDMA) 2012 competition. (This dataset comprises the click records corresponding to a number of different publishers). They have identified 108 features from FDMA click database to feed into the classifier(s) and have grouped/labeled the publishers in two categories: OK and Fraud. To deal with the problem of imbalanced data classes, different data balancing algorithms have been deployed, including SMOTE (Synthetic Minority Over-sampling Technique) [47] and ADASYN (Adaptive Synthetic Sampling Approach) [48]. The balanced data is then fed to various ensembles of classifiers (AdaBoost, LogitBoost, Bagging, etc.). Ultimately, the best results have been obtained using SMOTE and then performing the classification using Adaboost with Random Forest as the primary classifier.

Of the many forms of click fraud, the ones conducted by means of botnets comprising automated clickers are among the most severe. In this form of attack, attackers infect and use the computers of legitimate Internet users to deceive advertisers. In 2016, Iqbal et al. [38] introduced a novel technique named FCFraud, which can be built into the operating system (OS) to combat click fraud on the user side. Namely, the authors of [38] believe that adequate protection at the operating system level can save billions of dollars for advertisers. FCFraud significantly protects innocent users by detecting the fraudulent processes that perform click fraud silently. The model can be integrated with other known OS-level malware protection service. FCFraud inspects and analyzes Web mouse queries and events from all user processes and applies the RandomForest algorithm to classify ad queries automatically. It then uses heuristics to identify fraudulent ad clicks. In the final analysis presented in [38], the authors have examined their model using 25 popular websites (a total of 7708 HTTP requests), and the results have indicated that FCFraud could successfully detect all background processes involved in the execution of click fraud. Notably, FCFraud was 99.6% accurate in classifying ad requests from all user processes, and it showed 100% success in finding the fraudulent processes. It has also been claimed in this paper that FCFraud could also be useful against other large-scale botnet attacks such as the distributed denial of service (DDoS) and email spamming.

Thejas et al. [40] have presented a general time series model with multiple scales to predict click fraud behavior in minutes and hours. The proposed model consists of seven stages: pre-processing, data smoothing, fraudulent pattern identification, homogenizing variance, normalizing auto-correlation, developing the AR (Autoregressive) and MA (Moving Average) models and fine-tuning, along with an evaluation. The authors perform pre-analysis on the data and categorize the data into six datasets, each indexed with click time (based on the IP address of click and app id). The data smoothing stage, then, prepares the time series data representing ad clicks into two variants: time series data based on (i) learning approach and (ii) probabilistic modelling. The researchers merged various attributes to find a probability value, considered along with the timestamp, to form a time-probability pair using the two methods. In particular, they applied the logistic regression learning model to have a time series data in the form of a time-indexed label. They then calculated the individual probabilities of each attribute for the probability of being fraudulent. Next, the authors perform fraudulent pattern detection to set a prediction threshold value. It is conducted based on a high probability of positive click behavior, i.e., several download clicks observed for each attribute. Later, they check the homogeneity property of the data to verify the normality and homogeneity of variance. If this property is not met, they transform the time series to make its variance homogeneous and balance the normality. In the subsequent step, the stationarity of the data is checked using ACF (stands for Auto Co-relation Function), PACF (stands for Partial Auto Co-relation Function) [49] plots on different time charts—the minute and the hour. Finally, using the transformed time series data (based on ACF, PACF, ADF (Augmented Dickey–Fuller [49]), KPSS (Kwiatkowski–Phillips–Schmidt–Shin [50]), RM (Rolling Mean plots) and RSTD (Rolling Standard Deviation)), they model the AR and MR models. To select the best fit models from the set of evaluation parameters, AIC (Akaike Information Criteria), BIC (Bayesian Information Criteria) and forecasting errors are applied. Using AIC and BIC helps the authors measure the relative quality of the model for a given set of data and select a model from a finite set of models. Forecasting errors are calculated to find the differences between the observed and expected value, i.e., the value of unpredictability. Tuning the forecasting errors and minimizing the AIC/BIC, the best fit model for all time scale data was obtained. It is shown that the best model for forecasting fraudulent and non-fraudulent click behavior was the Probability-based model approach compared with the Learning-based probabilistic estimator model.

In 2019, Minastireanu and Mesnita [43] studied the precision of one of the more recent machine learning algorithms in detecting click fraud in the online environment. They examined click patterns in a public dataset, which included 200 million clicks generated over four days. They evaluated click-paths of individual users and then flagged the IP addresses (i.e., users) which produced lots of clicks but never ended up installing apps. The researchers particularly applied the experimental test for LightGBM (i.e., a Gradient Boosting Decision Tree-type method). They showed that the proposed model could remarkably outperform XGBoost (eXtreme Gradient Boosting) and SGB (Stochastic Gradient Boosting) in terms of computational speed and memory consumption. However, it should be noted that the model was limited in using 19 attributes/features, and insufficient resources prevented that the model be trained using the entire dataset.

Nagaraja et al. [44] introduced Clicktok, a statistical technique that detects click spam by identifying click traffic reuse. The underlying principle of Clicktok is based on modeling/measuring of timing properties of click traffic to support a technique capable of separating legitimate and fraudulent clicks. The proposed solution comprises two types of defenses: mimicry (passive defense), and bait-click (active defense). The mimicry defense technique is based on the assumption that 'organic click fraud' involves a reuse of legitimate click traffic, whereas 'non-organic click fraud' is based on the use of traffic synthesized using pseudo-random times. Accordingly, the organic click fraud can be detected by measuring it similarity to previously observed patterns of user behavior, whereas non-organic click fraud can be detected by measuring its traffic entropy which is generally lower than the entropy of legitimate user traffic. In the case of (active) bait-click defense, the ad network proactively injects bait clickstreams (with well-defined inter-click delay patterns) into legitimate user traffic. Any attempt (e.g., by a malicious publisher) to reuse these specific clickstream patterns will then be easily detected by the ad network. In this work, the non-negative matrix factorization (NMF) algorithm was the main approach used to partition click traffic in order to distinguish fraudulent clicks. According to the presented results, the proposed solution reached an accuracy of 99.6%.

To address the problem of click fraud, Thejas et al. [19] have proposed a model called CFXGB (Cascaded Forest and XGBoost). CFXGB is a combination of two learning models for feature transformation and classification. The model consists of three stages: pre-processing, feature transformation based on Cascaded Forest and XGBoost model classification. The authors completed similar pre-processing as previous works have done to prepare the data and feed it into the Cascaded Forest for feature transformation. Cascaded Forests is a part of the GcForest model [51], one of the ensemble-based models that use multiple learners to obtain a combined result. The GcForest model tries to mimic the functionality of deep learning models without the intense hyperparameter tuning through three features: (i) Cascade-by-Cascade/layer-by-layer processing, (ii) in-model feature transformation, and (iii) dataset flexibility. The Cascaded Forest in this study includes three ensemble models, i.e., Random Forests [52], Extremely Randomized trees (Extra Trees) [53], and XGBoost [54]. Each ensemble model is intended to help boost the model's performance. The XGBoost model is added to enhance diversity. The Cascaded Forest works in the form of layers/cascades, and each output is passed on to the next cascade for processing. The two parameters Max Layers and Early Stopping Rounds (ESR) are used to limit the number of cascades/layers added to the model, and if the number of cascades crosses Max Layers, the model stops adding more cascades. Once the model decided the last layer, the accuracy of the Cascaded Forest (as a classifier) is calculated. Following the training phase, the number of optimal cascades is obtained, and the data is transformed. The XGBoost model is then trained on the transformed data, and the parameters are tuned based on Maximum Depths and Learning Rate. Upon completing the training step, the XGBoost model predicts whether a single click is fraudulent or not on all observations. Later, to evaluate the performance of the proposed model, several experiments are conducted on different datasets to obtain a generalized set of parameter values. The performance results showed that XGBoost performed better as a classifier with feature transformation by Cascaded Forest. Additionally, CFXGB behaved exceptionally well in performing comparative analysis. The authors claim that the proposed model can be used as a generic model to solve other machine learning classification problems.

5. Case Study

According to [55], the history of ad click bots and PPC fraud dates back to 2003, when fraudulent publishers were starting to sign up their low-quality sites for Google AdSense and then proceed to click on the ads themselves. Competitive click fraud—advertiser competitor clicking—has also been a problem since the introduction of the pay per click model, and therefore the practice is commonplace today.

This section reviews some of the most famous real-world click bots and ad fraud malware campaigns, and also provides information about the primary techniques utilized in these campaigns to implement fake clicks. When deciding which particular real-world click bots and click campaigns to include in this paper, we were guided by a set of standardized criteria as well as the availability/accessibility of relevant information. The specific criteria applied included: (1) the scale of the attack caused by the click bot/malware (the number of the infected machines considered to be larger than one thousand), (2) the impact of the attack in the context of the cost (the amount of the inflicted damages), (3) the challenges faced by the industry/target to respond and recover quickly and effectively from the attack, and (4) the duration the attack (from weeks to years). A comparative analysis of click-bot campaigns is also provided in Table 2.

5.1. Clickbot.A—2006

In 2006, Google discovered Clickbot.A, a botnet that consisted of over 100,000 machines and conducted a low-noise click fraud attack against syndicated search engines [56] (search syndication is an alternative way of advertising on search engines, allowing advertisers to buy keyword-targeted traffic outside of search engine result sites. In search syndication, a search engine gives its services to an approved/authorized third party that requires search capabilities for their digital properties such as websites and apps). Similar to other botnets, Clickbot.A was composed of two distinct parts: bots (compromised machines) and a botmaster. It could issue HTTP requests to doorway sites (doorway sites refer to the websites set up by the attacker to function similar to a search engine), redirectors (a Web application that accepts an URL as part of its input is a redirector. It can issue an HTTP redirect to the client request for the URL specified in its input), and search engine result pages (a syndicated search engine offers pages that contain ad impression URLs and search results in which some of the ads were obtained from an ad network). Each bot/client was an Internet Explorer (IE) browser helper object (BHO) capable of accessing the entire DOM (document object model) of a webpage (i.e., a plugin to IE that was downloaded by users and operated using victims' machines to click on ads automatically). The botmaster ran an HTTP based Web application with a My-SQL database back-end. Most of the websites, doorway sites and/or redirectors that the attacker/bot operator used were supplied by ISP known to host compromised accounts.

The bots first contacted the botmaster to register and learn about a doorway site as well as to receive the instructions about the 'attack' keywords. As soon as the keywords were received, the bots queried the doorway site. Following the registration, the bot operated an infinite loop to request a doorway site and keywords and determined a candidate link to click on from the doorway site. The bots were configured to repeat the loop every 15 min; however, the number of clicks performed for each bot was limited, as Daswani et al. reported [56].

Table 2. Comparison on the most famous real-world click bots and ad fraud malware campaigns.

Type of Click Bot	Number of Infected Devices	Years Active	Target	Botmaster/Operator	Inflicted Damages
Clickbot.A—Botnet	100,000	2006	Advertisers on syndicated search engines	Unknown	USD 50,000
TDL-4—the fourth generation of TDSS botnet	4 million	2008–2012	Government agencies, 46 companies within the Fortune 500, and ISPs in the U.S., Germany and U.K.	Cybercrime group Known as GangstaBucks	USD 340,000 lost daily loss for advertisers
Bamital—A search hijacking and click fraud botnet	1 million	2009–2013	Advertisers (through major search engines and browsers, including those of Microsoft, Yahoo and Google	Unknown	USD 700,000 per year
Stantinko—A multi-use botnet	More than 500,000	2012–present	Joomla and WordPress administrative login pages in Russia and Ukraine	Unknown	Not known
Chameleon—humanlike botnet	More than 120,000	2013	Advertisers (through infecting Microsoft Windows machines in the U.S.)	Unknown	USD 6 million per month
ZeroAccess—botnet	1.9 million	2013	Advertisers (through major search engines)	Unknown	USD 100,000 per day
MIUREF—Trojan	Unknown	2013–2014	Advertisers in the United States, Japan, France and Australia.	Unknown	Not known
Kovter—botnet	700,000	2014–present	Advertisers (through major Windows Web browser in the US, Canada, the UK, and Australia)	KovCoreG group	Not known
Methbot—botnet	852,992 dedicated IPs, many falsely registered as US ISPs, 800–1200 dedicated servers operating from data centers in the U.S. and the Netherlands	2015–2017	Over 6000 premium domain names were targeted, then cloned, and made to serve up video ads.	A group of Russian criminals	USD 3 million to USD 5 million per day
3ve (Eve)—botnet	1.7 million	2017–2018	Advertisers (through 250,000 fake domains, spoofed from genuine websites including the Wall Street Journal, CNN, BBC.com and ESPN	A team of Russian and Kazakh nationals	USD 29 million
HyphBot)—botnet	500,000 computers in the US, UK, Netherlands and Canada	2017	Advertisers and a huge selection of premium inventory websites, including some of the most visited sites on the web	Unknown	USD 1.2 million per day
404Bot—botnet	Not known	2018–present	Advertisers and sites with a large inventory of ads.txt vendors	Unknown	USD 15 million

5.2. TDL-4—2008–2012

TDL-4 [57,58] was the fourth generation of TDSS (also known as Alureon and TDL-4) botnet, and the most sophisticated threat in 2008. This botnet utilized a “rootkit” to install itself deep within the infected PCs, thereby ensuring that it loaded before the Microsoft Windows operating system, according to experts at the Russian security firm Kaspersky Lab. TDSS was capable of removing approximately 20 other malicious programs from the host PC and preventing the system from communicating with other bot families. Interestingly, the activities of new TDL-4 variants were involved in massive click fraud operations. In particular, the machines infected with TDL-4 were periodically instructed to download “campaign” files, visit a list of designated websites, and generate ad revenue for the targeted sites.

5.3. Bamital—2009–2013

Bamital, discovered by Microsoft in early 2013, is a search hijacking and click fraud botnet that infected more than eight million computers worldwide, and affected many major search engines and browsers, including those of Microsoft, Yahoo and Google. Bamital was installed through drive-by downloads, which exploited kits implanted into hacked and malicious websites. The botnet exclusively applied the Phoenix Exploit Kit, i.e., a malware tool that used vulnerabilities in Web browsers, to silently install malware. The botnet was able to modify the organic search results on the host machines and redirect the victims toward webpages that presented advertising and referral commissions to affiliate marketers. Bamital then instructed infected systems to participate in click fraud or generate automated Internet traffic, as Microsoft reported in [59].

5.4. Fiesta Click Bot—2011

In 2011, Miller et al. [60] presented an analysis of two click bot techniques named Fiesta and 7cy. They examined the Fiesta and 7cy samples by infiltrating multiple malware Pay-Per-Install (PPI) services, in their study. They discovered PPI services were utilized to compromise machines and distribute malware to the compromised hosts in exchange for payment on the gray market. The Fiesta click bot was a click fraud model that utilized an intermediary to act as a layer of abstraction between ad syndicates and the click bots generating fraudulent traffic. The intermediary was responsible for laundering clicks generated by bots and delivering them to ad networks. The traffic generated by Fiesta click bots moved toward the intermediary and was then laundered within a series of ads syndicated in order to prevent fraud detection.

Two main players in Fiesta were a botmaster running a C&C server and Fiesta PPC Profitable Traffic Solution. The process of generating fraudulent clicks began with requesting a list of search query terms by Fiesta from C&C. In response, the bot received several terms that contained typographical errors, appeared arbitrary in nature and changed frequently. The bot then randomly selected one term that it would use for the remainder of this click. The bot started to communicate with the Fiesta PPC service and requested to receive ads that corresponded with the selected search query in the next step. In responding to the bot request, the PPC Ad Server delivered approximately 25 ads in XML format where some ads were related to the search, and the others appeared random. The bot selected an ad and informed the search engine on which ad it was about to click. At last, the bot contacted the PPC Click Server and completed the ad click.

5.5. 7cy Click Bot—2011

The behavior of the 7cy click bot was significantly different from Clickbot.A and Fiesta in the sense that it could emulate human Web browsing behavior by randomizing the click targets and introducing human-scale jitter to the interval between clicks [60]. The 7cy control servers were able to distribute fraud in accordance with the geographic region of the bot. The 7cy click bot utilized significant specifications, including random clicks, timing, and location-specific behavior, to present more realistic browsing behavior. It was also made of a substantially different C&C language and an extensive botmaster infrastructure. In particular, to control the bot’s click behavior, the 7cy C&C designated an initial site for surfing, i.e., a set of page content patterns to identify desirable links to click on and provide an inter-click delay time. Afterwards, the bot was able to leverage timing by entering a random amount of jitter into the delay between each click. Notably, each bot was instructed to generate more browsing traffic during the evening and workday (popular times).

5.6. Stantinko—2012

In 2012, Stantinko botnet infected half a million computers worldwide, and it went undetected for 5 years [61]. The botnet was mainly used to install browser extensions on the infected systems, inject ads, and then perform click fraud with a script (click on those injected ads). The two malicious browser extensions used by Stantinko were named "The Safe Surfing" and "Teddy Protection". The extensions were distributed through the Chrome Web Store to be employed in blocking unwanted URLs. With Stantinko malware, attackers could conduct various malicious activities involving brute-force attacks on Joomla and massive searches on Google and other backdoor activities [62,63].

5.7. ZeroAccess—2013

ZeroAccess is one of the most massive botnets in operation; it commanded 1.9 million infected computers in August 2013. It was a peer-to-peer (P2P) botnet for perpetrating advertising click fraud. The most significant part of ZeroAccess was the auto-clicking module capable of performing click fraud, which has caused losses to advertisers estimated at USD 2.7 million per month [64]. The ZeroAccess auto-clicking module conducted click fraud via simulating real/normal users' clicks. The module was able to download online ads onto the computer and then generate fake clicks on the ads pretending they were generated by normal users.

Structurally in this module, a real client web browser is invoked to enable realistic Web browsing behavior described by the HTML, JavaScript, and CSS specifications, as well as browser-specific quirks. It then periodically contacts the auto-clicking C&C to retrieve a list of publisher websites to visit. Subsequently, the module navigates to the URL provided by C&C in a hidden window, i.e., detecting the ad on the page, and imitates the user clicking on the ad by requesting the advertiser's URL. At the end of the session, the hidden window is closed and a new one is started with the next publisher on the list. It is worth noting that the auto clicking module did not simulate any user actions (such as purchase a product) on the advertiser webpage and therefore did not lead to any conversion.

5.8. MIUREF—2013

One of the most well-known click fraud malwares, MIUREF, was discovered in November 2013. MIUREF was a Trojan that used click fraud in its attack campaign. It was able to install itself as a browser plugin and then load itself every time the system's Internet browser was executed. Furthermore, MIUREF was able to install the TSPY_FAREIT malware family. Based on the reported infections, hackers utilized the most common method of spread MIUREF through spammed mails. The countries most affected by MIUREF were the United States, Japan, France, and Australia. It is worth noting that the click fraud was one of many MIUREF's possible malicious purposes [65].

5.9. Ramdo—2013

Ramdo [66] is another click fraud malware that helped cybercriminals make profits by silently clicking on online ads from infected systems discovered in late 2013. It was also capable of downloading and installing different other malicious software on the compromised machines. Ramdo was spread by leveraging exploit kits such as Angler, RIG and Magnitude and spam email containing URLs that redirected users to malicious Adobe Flash Player. Ramdo [67] was designed to inject malicious DLL code into process running on the compromised machine, download a CEF from its C2 server and navigate to ads on a fake browser. Then, the malware was able to access target ads via a specially designed search portal instead of directly navigating them. By accessing a website via a search portal with links to sponsored ads, website analysis tools were misled, and advertisers were made to believe that their ads were viewed by users performing Web searches, not bots.

5.10. Boaxxe—2013

Coming from the malware family Win32/Boaxxe.BE (Win32/Boaxxe was a family of trojans that were able to install themselves as Browser Helper Objects (BHO). It could contact remote websites to download and execute arbitrary files), Boaxxe was initially designed to drive traffic to advertisement websites by using different click fraud techniques and earning money from these websites [68].

The story began in September 2013 with partnerka.me, a website where people-called affiliates—were monetarily compensated to infect users with Win32/Boaxxe.BE. When an affiliate registered himself, he gained access to the site. They then could find the latest information on the state of the business as well as the affiliate with the program he had to distribute. The binary was provided either by direct download or through a download URL, in the form "web5.asia/promos/download?token=TOKEN&sub_id=SUB-ID", where TOKEN was a 20-byte long value identifying the affiliate. The affiliate could also distinguish between different groups of binaries in his statistics if the SUB-ID value was set. It should be stated that each binary contained an affiliate ID, allowing the C&C to credit the correct affiliate's account whenever a new machine became infected. There were two kinds of click fraud implemented by Win32/Boaxxe.BE as follows [55]:

1. User-initiated click fraud: users who entered keywords in search engines could be rerouted to related ad websites. This form of click fraud had already been seen in various malware families, such as Win32/TrojanDownloader.Tracur or Win32/Goblin.
2. Automated click fraud: Boaxxe could browse ad websites silently, without the user's knowledge.

After binaries were distributed by partnerka.me to its affiliates, who were then in charge of infecting users, these users were forced to browse various advertisement websites, either automatically or during their search engines' browsing. Notably, infected machines reached some doorway search engines that returned a list of related advertisement websites for each keyword. The provided URLs then launched a chain of redirection through websites interconnected in an advertiser-publisher relationship. The advertised websites—which could be legitimate—needed to remunerate the ad networks for the traffic they brought. These networks received a commission and paid back the doorway search engines. Eventually, partnerka.me received the rest of the money, took its portion, and paid its affiliates.

5.11. Chameleon—2013

Chameleon click botnet [69], found by spider.io in 2013, was sophisticatedly designed to impersonate real/human browsing behavior, including running a mouse over display ads. Chameleon infected more than 120,000 Microsoft Windows machines in the U.S. and perpetrated click fraud on 202 websites, which were serving 14 billion ad impressions per month. Each bot/infected machine operated Flash and executed JavaScript. The bots created click traces that indicated legitimate users. They also generated client-side events intended to simulate real user engagement, and they were able to issue clicks on ad impressions with an average CTR of 0.02% and mouse traces across 11% of ad impressions. In order to evade detection, the bots subjected the host machine to heavy load, causing them to crash and repeatedly restart.

Despite the high level of sophistication in the bot operations, the entire botnet's traffic was very homogeneous since the bot browsers were engineered as Internet Explorer 9.0 running on Windows 7. They visited the same set of websites with slight variation and clicked on ads the way a real person would. They generated uniformly random clicks that were coordinated across ad impressions; similarly, mouse traces were generated randomly. The estimated cost of Chameleon botnet in click fraud was USD 6 m per month.

5.12. Kovter—2014–Present

Trojan Kovter [66] was a click fraud malware that utilized fileless design to bypass detection after infection. Kovte was spread via email attachments containing malicious

JavaScript files. The primary capability of Kovter was to download additional malware, steal sensitive information, and give attackers access to the infected machine. The malware was stored in the system registry rather than a physical file on the computer's hard drive, making it harder to detect and even harder to remove. Kovter operated on the base of a hidden Chromium embedded framework (CEF) browser on the compromised system. The command-and-control server would periodically send ads to the infected machine to display ads in the CEF browser. Kovter Trojan has been in existence since 2013, and it is one example of a constantly evolving malware. Starting November 2018, most of the key players behind the Kovter threat had been prosecuted, and the infrastructure of the click fraud malware was broken down.

5.13. Methbot—2015–2017

Methbot, discovered in 2016, is one of the largest and most lucrative ad fraud operations that caused financial damages between USD 3 million and 5 million per day, as reported by WhiteOps [70]. The Russian Methbot fraud is also known as the biggest digital ad fraud perpetrated by fake clicks on video ads. The bot inventors understood that video advertising on premium websites brings the highest revenue in the world of digital advertising. Therefore, they designed Methbot to hijack premium publishers by forging URLs in their video ads in order to attract advertising money. To accomplish this, Methbot follows the below steps:

First, it selects a domain or URL from a list of premium publishers (premium publishers are websites with optimized Web designs, compelling content, loyal audience that delivers better results than other websites for the ads they display) and creates a fake page. Subsequently, using the industry-standard VAST (a Video Ad Serving Template, is a universal XML schema for structuring ad tags that serve ads to digital video players [71]) protocol, Methbot requests an online video ad with one of the Methbot identifiers to receive its credits. Ultimately, to create fake views and clicks, Methbot loads the video ad through a proxy and plays it within the simulated browser. More importantly, specified anti-fraud and viewability verification codes are also loaded and issue false alerts to make the activity appear legitimate. It is worth noting that, even though compromised systems were used to generate the views in the Methbot program, much of the activity was controlled from centralized servers located in the USA and Netherlands [72].

5.14. 3ve—2017–2018

In February 2017, a mysterious botnet, "3ve," appeared with the capability of creating more than 10,000 forged websites and 40,000 new IP addresses each day to generate fake traffic and reap the ad revenue. Tamer Hassan, the founder and CTO of WhiteOps, was the first person who found this botnet and named it 3ve due to its three primary characteristics; the super-fast pace of spreading, large-scale expansion and the sophisticated strategies implemented to create it. The main distinction of 3ve was its accelerating and growing power that allowed the botnet to use someone's computer (hack it to perform malicious action) on one day and function normally on the next day. More than 1.7 million devices were silently infected with 3ve at that time. Moreover, 3ve was able to simultaneously bypass different security software and examine companies' defensive maneuvers to learn how different prevention and detection methods work in order to mask their moves [73]. 3ve was based on a complicated system consisting of three separate but interconnected botnet systems all doing more or less the same thing. Similar to Methbot, malicious operators had access to a large number of infected systems that they could remotely control to open browsers and visit their fake websites.

5.15. HyphBot—2017

In 2017 Adform identified HyphBot, potentially the largest bot network since the Methbot and 3ve takedown [74]. It was a botnet that displayed ads on unauthorized websites. Specifically, it targeted a massive selection of premium inventory websites (i.e., the most visited sites on the Web). In 2017, IAB Tech Lab introduced ads.txt to combat the growing problem of domain spoofing and illegitimate inventory arbitrage in programmatic advertising. ads.txt is a text file that holds a list of authorised digital sellers allowed to resell a website's ad inventory [75]. By using the ads.txt lists of websites, HyphBot created composite domain names that were used for fake impressions on video ads. The fake websites included just video players showing ads and a few links [76]. The attackers employed a network of existing botnets as their ad clickers. They appended a genuine URL with a randomized set of letters and numbers. For advertisers, at a glance, it seemed their ad had been displayed on sites such as Forbes or the Economist. However, they had paid for an impression on a fake website. HyphBot operated for a relatively short time before expiring, but nevertheless managed to generate millions of fake ad revenue.

6. Discussion and Conclusions

Click fraud is a serious problem for the cyber-security community with a significant revenue potential for their perpetrators. In this study, we have surveyed different types of Internet bots and showed how different click fraud campaigns are performed. We have also provided a summary of the current state of click advertising fraud, including representative forms of attack and representative countermeasures. We have, then, discussed several prevention mechanisms and detection techniques in this regard. Unfortunately, it should be noted that due to very limited availability of information, we were able to provide only partial coverage of commercial solutions in this domain.

As part of this research, we have made the following observations:

Among the various forms of click fraud discussed in Section 3.2, the fraud conducted by means of click bots (i.e., click-botnet) is not only the most prevalent type, but also the most difficult one to defend against due to the large-scale and automated nature of click-botnet actions. Furthermore, modern day click-botnets exhibit advanced human-like interaction characteristics, such as mimicking the random mouse movement in click-path patterns, and the use of traditional interaction-based user behavior (the "mouse pointer moving in a straight line" is an example of an interaction [77]) analysis in detecting these bots is likely to result in a high number of false positives.

Even a simple cross comparison between the techniques proposed in the research literature (as presented in Section 4) and the real-world click bot examples (as presented in Section 5) points to potential inefficacy of the existing state-of-the-art solutions when dealing with complex real-world click bots that are rapidly evolving and mutating. Namely, the majority of the proposed models rely on supervised "classification" based machine learning techniques. However, as Radware stated in [77], the supervised learning models trained on past data will have difficulty detecting any change/mutations in the bot behavior. Unsupervised learning models are not a bullet-proof solution either, as click bots with anomalous characteristics may also result in false positives, due to the fact that certain human visitors can exhibit anomalous characteristics as well.

The other main root challenge in defending against click fraud in the ad industry is the lack of transparency and information accessibility in the ad ecosystem. As pointed in [36], ad networks are unable to track user activity on advertisers' websites; instead, they can only track limited user engagement for multiple ads shown across multiple publishers, ultimately resulting in a broad-but-shallow view into user engagement. On the other side, advertisers are able to track detailed user actions on their websites (i.e., their ads) but not the interactions with other ads of the same publisher, ultimately providing a narrow-but-deep view into user engagement.

Regretfully, many research studies that have looked into the problem of click fraud still assume (i.e., rely on) basic fraudulent click patterns and detection parameters—e.g., too many download clicks observed with respect to an attribute such as IP address. We have noted that (recall from Section 5) the sophisticated and stealthy botnets, such as TDL4 botnet, can evade threshold-based filters by performing only one IP address click per day but by using millions of different bots.

Therefore, going forward, the only way to stay ahead of the continually evolving threat landscape involving click bots will require the development and utilization of more sophisticated defensive capabilities, as well as a more readily available access to datasets from real-world click-fraud incidents. Despite the weakness or negligence in the range of proposed solutions in Section 4, we believe that the most successful and widely used ML approaches in detecting click fraud are those deploying ensembles of decision trees. They provide reasonably fast learning, and they are well suited to highly skewed class distributions, noisy nonlinear patterns, and mixed variable types. In particular, the combination of tree-based ensemble classifiers and backward feature elimination can lead to a promising approach to tackle highly imbalanced datasets. Additionally, the literature states that the gradient boosting decision tree (GBDT) proved to be an effective approach for click fraud detection due to its efficiency, accuracy, and interpretability. It could achieve state-of-the-art performances in click prediction.

Additionally, effective categorization of traffic is an approach that should be considered alongside. An accurate separation of click bots from crowd click/click farms or legitimate Web users will improve the quality of mitigation and detection strategies. Namely, accurate traffic categorization (e.g., in training data) can undoubtedly prevent analytic pollution problems and improve the overall performance/detection quality.

Furthermore, to the best of our knowledge, there is no substantial previous research that has looked into the problem of click fraud detection with the use of a combined model of "user behavior observed on a single page" vs. "user's click-stream behavior observed over multiple pages". It is our conviction that integrating both of these types of information will help capture the more subtle differences in traffic patterns of click bots and legitimate web users, and thus greatly assist in the design of more effective click-bot detection systems.

Finally, it should be noted that, going forward, successful click bot detection will also require continuous efforts to acquire better understanding of the functional capabilities of real-world click bots (i.e., botnets) as well as the ultimate objectives of their (botnet) operators.

Much art and science are still being developed on various aspects of click fraud containment as the online advertising market is booming while working to support the needs of online advertisers and retailers. Successful fraud management gives advertising networks a competitive edge and helps them obtain the highest possible return on advertiser investment.

Author Contributions: Conceptualization, S.S., and N.V.; methodology, S.S.; validation, S.S. and N.V.; writing—original draft preparation, S.S.; writing—review and editing, S.S. and N.V.; supervision, N.V.; project administration, N.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fourberg, N.; Serpil, T.; Wiewiorra, L.; Godlovitch, I.; De Streel, A.; Jacquemin, H.; Jordan, H.; Madalina, N.; Jacques, F.; Ledger, M. Online Advertising: The Impact of Targeted Advertising on Advertisers, Market Access and Consumer Choice. 2021. Available online: https://www.europarl.europa.eu/thinktank/en/document.html?reference=IPOL_STU%282021%29662913 (accessed on 27 October 2021).
2. Pooranian, Z.; Conti, M.; Haddadi, H.; Tafazolli, R. Online Advertising Security: Issues, Taxonomy, and Future Directions. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2494–2524. [CrossRef]

3. Cavazos, P.R. The Economic Cost of Invalid Clicks in Paid Search and Paid Social Campaigns. 9. 2020. Available online: https://cdn2.hubspot.net/hubfs/5228455/UniBaltimore%20PPC%20Fraud-1.pdf (accessed on 27 October 2021).
4. Mirtaheri, S.M.; Dinçktürk, M.E.; Hooshmand, S.; Bochmann, G.V.; Jourdan, G.-V.; Onut, I.V. A Brief History of Web Crawlers. *arXiv* **2014**, arXiv:1405.0749.
5. Managing and Mitigating Bots: The Automated Threat Guide—Netacea. 2018. Available online: https://www.netacea.com/managing-and-mitigating-bots-guide/ (accessed on 27 October 2021).
6. Bot Attacks: Top Threats and Trends. Barracuda. 2021. Available online: https://assets.barracuda.com/assets/docs/dms/Bot_Attacks_report_vol1_EN.pdf (accessed on 27 October 2021).
7. Bad Bot Report 2020: Bad Bots Strike Back | Imperva. 2020. Available online: https://www.imperva.com/blog/bad-bot-report-2020-bad-bots-strike-back/ (accessed on 27 October 2021).
8. Ultimate Guide to Bot Management. [E-book] Radware. 2019. Available online: https://blog.radware.com/wp-content/uploads/2019/09/Radware_UltimateGuideBotManagement_Final.pdf (accessed on 27 October 2021).
9. Everything You Need to Know about Bots in 2020. Netacea. 2020. Available online: https://www.netacea.com/evolving-threat-guide-2020/ (accessed on 27 October 2021).
10. What's the Difference between a Bot and a Botnet? 2020. Available online: https://www.radwarebotmanager.com/bots-vs-botnets/ (accessed on 27 October 2021).
11. Digital Education: The Cyberrisks of the Online Classroom. 2020. Available online: https://securelist.com/digital-education-the-cyberrisks-of-the-online-classroom/98380/ (accessed on 27 October 2021).
12. Differences between Marketplaces and Classifieds Portals. MarketplaceGuru. 2017. Available online: http://marketplaceguru.de/differences-marketplaces-classifieds/ (accessed on 27 October 2021).
13. Industry Report: Bad Bot Landscape 2019—The Bot Arms Race Continues. 2019. Available online: https://www.globaldots.com/resources/blog/industry-report-bad-bot-landscape-2019-the-bot-arms-race-continues/ (accessed on 27 October 2021).
14. Dave, V.; Guha, S.; Zhang, Y. Viceroi: Catching Click-Spam in Search Ad Networks. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 765–776.
15. What Is an Ad Tag and How to Generate It [Examples Inside]. 2020. Available online: https://epom.com/blog/ad-server/what-is-an-ad-tag (accessed on 27 October 2021).
16. About Bidding Features in Display Campaigns–Google Ads Help. Available online: https://support.google.com/google-ads/answer/2947304?visit_id=637427097783271992-4033346762&rd=1 (accessed on 27 October 2021).
17. Gilani, Z.; Farahbakhsh, R.; Tyson, G.; Crowcroft, J. A Large-Scale Behavioural Analysis of Bots and Humans on Twitter. *ACM Trans. Web* **2019**, *13*, 1–23. [CrossRef]
18. Xu, H.; Liu, D.; Koehl, A.; Wang, H.; Stavrou, A. Click Fraud Detection on the Advertiser Side. In *European Symposium on Research in Computer Security*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 419–438.
19. Thejas, G.S.; Dheeshjith, S.; Iyengar, S.S.; Sunitha, N.R.; Badrinath, P. A Hybrid and Effective Learning Approach for Click Fraud Detection. *Mach. Learn. Appl.* **2021**, *3*, 100016.
20. Gandhi, M.; Jakobsson, M.; Ratkiewicz, J. Badvertisements: Stealthy Click-Fraud with Unwitting Accessories. *J. Digit. Forensic Pract.* **2006**, *1*, 131–142. [CrossRef]
21. Metwally, A.; Agrawal, D.; El Abbad, A.; Zheng, Q. 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), Toronto, ON, Canada, 22–29 June 2007; p. 52.
22. Kim, C.; Miao, H.; Shim, K. CATCH: A Detecting Algorithm for Coalition Attacks of Hit Inflation in Internet Advertising. *Inf. Syst.* **2011**, *36*, 1105–1123. [CrossRef]
23. Mayer, V.A.A.; Pinkas, K.N.B.; Reiter, M.K. On the Security of Pay-Per-Click and Other Web Advertising Schemes. *Comput. Netw.* **1999**, *31*, 1091–1100.
24. Neal, A.; Kouwenhoven, S.; Sa, O. Quantifying Online Advertising Fraud: Ad-Click Bots vs Humans. In *Techincal Report*; Oxford Bio Chronometrics: London, UK, 2015.
25. Chellapilla, K.; Maykov, A. A Taxonomy of JavaScript Redirection Spam. In Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web, Banff, AB, Canada, 8 May 2007; pp. 81–88.
26. How to Simulate a Click by Using x, y Coordinates in JavaScript? *The Web Dev.* 2021. Available online: https://thewebdev.info/2021/05/02/how-to-simulate-a-mouse-click-using-javascript/ (accessed on 27 October 2021).
27. Metwally, A.; Emekçi, F.; Agrawal, D.; El Abbadi, A. Sleuth: Single-Publisher Attack Detection Using Correlation Hunting. *Proc. VLDB Endow.* **2008**, *1*, 1217–1228. [CrossRef]
28. Shaw, G. Spyware & Adware: The Risks Facing Businesses. *Netw. Secur.* **2003**, *2003*, 12–14. [CrossRef]
29. Ding, X. A Hybrid Method to Detect Deflation Fraud in Cost-per-Action Online Advertising. In *International Conference on Applied Cryptography and Network Security*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 545–562.
30. Reiter, M.K.; Anupam, V.; Mayer, A.J. Detecting Hit Shaving in Click-Through Payment Schemes. In Proceedings of the USENIX Workshop on Electronic Commerce, Boston, MA, USA, 31 August–3 September 1998.
31. Ge, L.; King, D.; Kantardzic, M. Collaborative Click Fraud Detection and Prevention System (Ccfdp) Improves Monitoring of Software-Based Click Fraud. *E-COMMERCE* **2005**, *2005*, 34.

32. Juels, A.; Stamm, S.; Jakobsson, M. Combating Click Fraud via Premium Clicks. In Proceedings of the USENIX Security Symposium, Boston, MA, USA, 6–10 August 2007.
33. Zhang, L.; Guan, Y. Detecting Click Fraud in Pay-per-Click Streams of Online Advertising Networks. In Proceedings of the 2008 the 28th International Conference on Distributed Computing Systems, Beijing, China, 17–20 June 2008; pp. 77–84.
34. Haddadi, H. Fighting Online Click-Fraud Using Bluff Ads. *ACM SIGCOMM Comput. Commun. Rev.* **2010**, *40*, 21–25. [CrossRef]
35. Kantardzic, M.; Walgampaya, C.; Yampolskiy, R.; Woo, R.J. Click Fraud Prevention via Multimodal Evidence Fusion by Dempster-Shafer Theory. In Proceedings of the 2010 IEEE Conference on Multisensor Fusion and Integration, Salt Lake City, UT, USA, 5–7 September 2010; pp. 26–31.
36. Dave, V.; Guha, S.; Zhang, Y. 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, Helsinki, Finland, 13–17 August 2012; pp. 175–186.
37. Kitts, B.; Zhang, J.Y.; Roux, A.; Mills, R. Click Fraud Detection with Bot Signatures. In Proceedings of the 2013 IEEE International Conference on Intelligence and Security Informatics, Seattle, WA, USA, 4–7 June 2013; pp. 146–150.
38. Iqbal, M.S.; Zulkernine, M.; Jaafar, F.; Gu, Y. Fcfraud: Fighting Click-Fraud from the User Side. In Proceedings of the 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE), Washington, DC, USA, 7–9 January 2016; pp. 157–164.
39. Almeida, P.S.; Gondim, J.J. Click Fraud Detection and Prevention System for Ad Networks. *J. Inf. Secur. Cryptogr.* **2018**, *5*, 27–39. [CrossRef]
40. Thejas, G.S.; Soni, J.; Boroojeni, K.G.; Iyengar, S.S.; Srivastava, K.; Badrinath, P.; Sunitha, N.R.; Prabakar, N.; Upadhyay, H.A. Multi-Time-Scale Time Series Analysis for Click Fraud Forecasting Using Binary Labeled Imbalanced Dataset. In Proceedings of the 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Bengaluru, India, 20–21 December 2019; Volume 4, pp. 1–8.
41. Kar, K. A Study on Machine Learning Approach using Ensemble Learners for Click Fraud Detection in Online Advertisement. *J. Gujarat Res. Soc.* **2019**, *21*, 693–705.
42. Pan, L.; Mu, S.; Wang, Y. User Click Fraud Detection Method Based on Top-Rank-k Frequent Pattern Mining. *Int. J. Mod. Phys. B* **2019**, *33*, 1950150. [CrossRef]
43. Minastireanu, E.-A.; Mesnita, G. Light Gbm Machine Learning Algorithm to Online Click Fraud Detection. *J. Inform. Assur. Cybersecur.* **2019**. Available online: https://www.researchgate.net/profile/Gabriela-Mesnita/publication/332268924_Light_GBM_Machine_Learning_Algorithm_to_Online_Click_Fraud_Detection/links/5cab2156299bf118c4ba99c4/Light-GBM-Machine-Learning-Algorithm-to-Online-Click-Fraud-Detection.pdf (accessed on 27 October 2021).
44. Nagaraja, S.; Shah, R. Clicktok: Click Fraud Detection Using Traffic Analysis. In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, Miami, FL, USA, 17–19 May 2019; pp. 105–116.
45. Oentaryo, R.; Lim, E.-P.; Finegold, M.; Lo, D.; Zhu, F.; Phua, C.; Cheu, E.-Y.; Yap, G.-E.; Sim, K.; Nguyen, M.N. Detecting Click Fraud in Online Advertising: A Data Mining Approach. *J. Mach. Learn. Res.* **2014**, *15*, 99–140.
46. Gabryel, M. Data Analysis Algorithm for Click Fraud Recognition. In *Information and Software Technologies*; Communications in Computer and Information Science; Damaševičius, R., Vasiljevienė, G., Eds.; Springer International Publishing: Cham, Switzerland, 2018; Volume 920, pp. 437–446. [CrossRef]
47. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority over-Sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
48. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 1–6 June 2006; pp. 1322–1328.
49. Fuller, W.A. *Introduction to Statistical Time Series*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 428.
50. Kwiatkowski, D.; Phillips, P.C.; Schmidt, P.; Shin, Y. Testing the Null Hypothesis of Stationarity against the Alternative of a Unit Root: How Sure Are We That Economic Time Series Have a Unit Root? *J. Econom.* **1992**, *54*, 159–178. [CrossRef]
51. Freund, Y.; Schapire, R.E. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
52. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
53. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely Randomized Trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
54. Chen, T.; Guestrin, C. Xgboost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
55. Oli. A Short History of Ad Click Bots & PPC Fraud. 2020. Available online: https://www.clickcease.com/blog/a-short-history-of-ad-click-bots-ppc-fraud/ (accessed on 27 October 2021).
56. Daswani, N.; Stoppelman, M. The Anatomy of Clickbot. A. In Proceedings of the First Workshop on Hot Topics in Understanding Botnets, Cambridge, MA, USA, 10 April 2007.
57. Staff, S.C. TDL-4 Variant Spreads Click-Fraud Campaign. 2021. Available online: https://www.scmagazine.com/news/security-news/tdl-4-variant-spreads-click-fraud-campaign (accessed on 27 October 2021).
58. Rent-a-Bot Networks Tied to TDSS Botnet—Krebs on Security. 2011. Available online: https://krebsonsecurity.com/2011/09/rent-a-bot-networks-tied-to-tdss-botnet/ (accessed on 27 October 2021).

59. Microsoft, Symantec Hijack 'Bamital' Botnet—Krebs on Security. 2013. Available online: https://krebsonsecurity.com/2013/02/microsoft-symantec-hijack-bamital-botnet/ (accessed on 27 October 2021).
60. Miller, B.; Pearce, P.; Grier, C.; Kreibich, C.; Paxson, V. What's Clicking What? Techniques and Innovations of Today's Clickbots. In *Detection of Intrusions and Malware, and Vulnerability Assessment*; Lecture Notes in Computer Science; Holz, T., Bos, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 164–183. [CrossRef]
61. Stantinko Botnet Was Undetected for at Least 5 Years While Infecting Half a Million Systems. 2017. Available online: https://securityaffairs.co/wordpress/61250/malware/stantinko-botnet.html (accessed on 27 October 2021).
62. StantinkoTeddy Bear Surfing Out of Sight. 2017. Available online: https://www.welivesecurity.com/wp-content/uploads/2017/07/Stantinko.pdf (accessed on 27 October 2021).
63. How Stantiko Click-Fraud Bot Butchers CPA's. C3 Metrics. 2017. Available online: https://c3metrics.com/stantiko-click-fraud-bot-butchers-cpas/ (accessed on 27 October 2021).
64. Pearce, P.; Dave, V.; Grier, C.; Levchenko, K.; Guha, S.; McCoy, D.; Paxson, V.; Savage, S.; Voelker, G.M. Characterizing Large-Scale Click Fraud in ZeroAccess. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 141–152. [CrossRef]
65. The Click Fraud Malware: How MIUREF Turns Users into Cybercriminal Accomplices–Threat Encyclopedia. 2014. Available online: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/133/the-click-fraud-malware-how-miuref-turns-users-into-cybercriminal-accomplices (accessed on 27 October 2021).
66. Malware Spotlight: What Is Click Fraud?-Infosec Resources. 2019. Available online: https://resources.infosecinstitute.com/topic/malware-spotlight-what-is-click-fraud/#gref (accessed on 27 October 2021).
67. Ramdo Click-Fraud Malware Continues to Evolve | SecurityWeek.Com. 2016. Available online: https://www.securityweek.com/ramdo-click-fraud-malware-continues-evolve (accessed on 27 October 2021).
68. Boaxxe Adware: 'A Good ad Sells the Product without Drawing Attention to Itself'—Pt 1. 2014. Available online: https://www.welivesecurity.com/2014/01/14/boaxxe-adware-a-good-ad-sells-the-product-without-drawing-attention-to-itself-pt-1/ (accessed on 27 October 2021).
69. spider.io—Discovered: Botnet Costing Display Advertisers over Six Million Dollars per Month. 2013. Available online: http://www.spider.io/blog/2013/03/chameleon-botnet/ (accessed on 27 October 2021).
70. White Ops. The Methbot Operation. *White Ops*. 2016. Available online: https://cdn2.hubspot.net/hubfs/3400937/WO_Methbot_Operation_WP_01.pdf (accessed on 27 October 2021).
71. Digital Video Ad Serving Template (VAST). Available online: https://www.iab.com/guidelines/vast/ (accessed on 27 October 2021).
72. "Biggest Ad Fraud Ever": Hackers Make $5M A Day by Faking 300M Video Views. 2016. Available online: https://www.forbes.com/sites/thomasbrewster/2016/12/20/methbot-biggest-ad-fraud-busted/?sh=50a126324899 (accessed on 27 October 2021).
73. Eight People Are Facing Charges as A Result of The FBI's Biggest-Ever Ad Fraud Investigation. 2018. Available online: https://www.buzzfeednews.com/article/craigsilverman/3ve-botnet-ad-fraud-fbi-takedown (accessed on 27 October 2021).
74. How AdformDiscovered HyphBot. 2017. Available online: https://site.adform.com/media/85132/hyphbot_whitepaper_.pdf (accessed on 27 October 2021).
75. What Is Ads.txt and How Does It Work? 2020. Available online: https://clearcode.cc/blog/ads-txt/ (accessed on 27 October 2021).
76. Oli. Click Fraud Hall of Fame: HyphBot. 2020. Available online: https://www.clickcease.com/blog/click-fraud-hall-of-fame-hyphbot/ (accessed on 27 October 2021).
77. Intent-Based Deep Behavioral Analysis: A Proprietary Bot Detection Technology. 2019. Available online: https://www.radwarebotmanager.com/web/wp-content/uploads/IDBA_WP.pdf (accessed on 27 October 2021).