1.1 Online Advertising Fraud

Nevena Vratonjic, Mohammad Hossein Manshaei and Jean-Pierre Hubaux

Abstract Over the last decade, online advertising has become a major component of the Web, leading to large annual revenues (e.g., $26.04 billion in US in 2010 [4]). Internet advertising is a very successful form of advertising as it provides an easy and effective way for advertisements to be targeted to individual users' interests. Unfortunately, fraudsters were able to exploit several vulnerabilities of the online advertising model and started abusing the system in order to make a profit out of it. These attacks are illegal only in a few countries and states (e.g., click fraud is a felony covered by Penal Code 502 in California and the computer misuse act in the UK). In most of the cases, the fraudsters instead violate terms of service of online advertising networks (e.g., in Nigeria where there is no law against this type of cybercrime and in India, where companies even advertised in national newspapers looking for people willing to use computers to click on ads, with no repercussions from authorities). In this chapter, we aim to provide a better understanding of vulnerabilities of online advertising systems, the well-known attacks, and possible countermeasures. We first address the online advertising system model and discuss different revenue models for it. We explain vulnerabilities of these models and classify the identified attacks into four main types: click fraud, malvertising, inflight modification of ad traffic, and adware. For each type of attack, we address how fraudsters can make profit from the existing advertising system. We discuss in more detail some of the attacks reported in practice. Finally, we address the possible countermeasures to each of the attacks.

1.1.1 Advertising on the Internet

Online advertising is a form of marketing that relies on the Internet to deliver marketing messages to the targeted users. Internet advertisement typically comprises a short text, an image, or an animation embedded into a Web page. The purpose of an ad is generally to capture a user's attention and persuade him to purchase or to consume a particular product or a service and, consequently, to increase the revenue of the advertiser. Advertisers pay for their ads to appear online, thus online advertising has become the major business model for monetizing online content. In contrast to other types of media (e.g., television or radio), online advertisements are not limited to an audience at a given time or a geographic location. An additional benefit is that online advertising allows for the customization of advertisements, thus increasing the probability that a user is interested in the advertised products and services. Hence, many advertisers, realizing the opportunities of online advertising, invest significant budgets into this form of advertising. Consequently, for many of the websites that users visit, a number of advertisements appear together with the content of a Web page.

Now let us look at how online advertisements are embedded into the content of Web pages, which techniques are used to tailor ads to users' interests and the main revenue models in online advertising.

Ad Serving Architecture

Ads are embedded into Web pages either through an ad serving system, or by websites themselves. Although it might be a straightforward task for major publishers with marketing units to sell the advertising space on their Web pages to advertisers, this is not the case for a large number of small publishers for which the overhead of doing so may surpass the benefits. Ad networks emerged as a solution to increase the reach of online advertising campaigns across these small publishers as well. Publishers offer their advertising space to ad networks that deal with advertisers and sell the advertising space on behalf of publishers.

The prevalent model of the Internet advertisement serving architecture is depicted in Figure 1.1 . In this model, an ad network plays the role of intermediary between advertisers and publishers, and its job is to automatically include ads into the appropriate online content. For this purpose, the ad network provides publishers with the HTML code that publishers should include (i.e., copy-paste) into the HTML code of their Web pages. When users browse these Web pages, relevant ads appear together with the publishers' content.

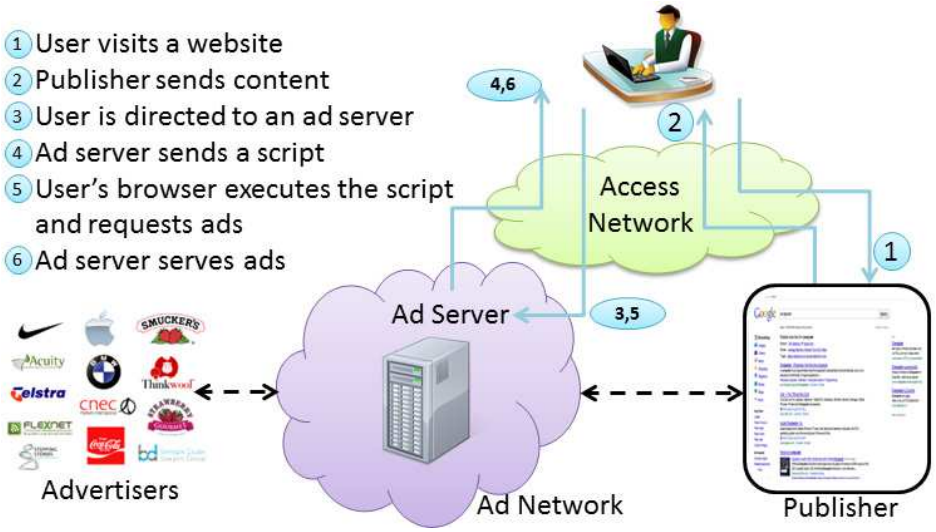

Figure 1.1 The ad serving architecture. *Advertisers* subscribe to an *Ad Network* whose role is to automatically embed ads into related Web pages. *Publishers* and ad networks have a contractual agreement (dashed arrow) that lets ad servers add advertisement to publishers' Web pages. Ads are stored at *Ad Servers*, which belong to the ad network. When a *User* visits a website of a publisher that hosts ads (step 1), the user's browser starts downloading the content of the Web page (step 2), and is then directed to one of the ad servers belonging to the ad network (step 3). During the first communication with the ad server, a script is served to the user (step 4) that executes on the user's machine and requests ads from the ad server (step 5). The ad server chooses and serves ads that match users' interest (step 6) in order to maximize the potential ad revenue.

The protocol, illustrated in Figure 1.1 , can be represented as follows:

1. A user's browser issues a request for the Web page corresponding to the *URL* the user types into browser's address bar.

2. The downloaded Web page contains the publisher's content and the block of the HTML code provided by the ad network. The HTML code redirects the browser to communicate with an ad server and download the ads that should accompany the publisher's content. This approach makes the ad serving system scalable, as the workload is distributed across users, rather than having a website communicate with an ad network on behalf of each user and deliver the ads together with the content. In addition, it allows ad servers and advertisers to keep the control, as ads are stored and maintained at their servers.
3. Typically, the user's browser first requests a script (e.g., Javascript) from the ad server.
4. When the script is fetched, it executes locally on the user's machine and collects certain parameters that influence the selection of ads by the ad server, including the HTTP cookies if they were deposited by the ad server during previous interactions. Cookies uniquely identify users and enable the profiling of their browsing preferences. This enables ad servers to track users across multiple websites. Besides collecting relevant information about the user before actually serving the ads, an additional benefit of having the HTML code that directs users to first download the script is that it is simple and easy to maintain, as only a few lines of a generic code (a reference to the Javascript) are added in the code of Web pages. Thus, if the ad network wants to modify the way ads are included in online content, it can simply modify the script that is hosted on their servers, rather than requesting each of the associated publishers to implement the corresponding modifications.
5. Information collected by the script is communicated back to the ad server with the request for ads.
6. The ad server chooses and serves the most appropriate ads for the given user. The browser merges the ads with previously downloaded elements of the Web page.

Due to its many advantages, this approach is widely used in practice. However, there are several drawbacks as well. Because users fetch ads from third-party servers (i.e., servers different from publishers' servers), the ad serving technology slows down the display of Web pages, consumes extra bandwidth, can be used as an attack vector to compromise the security of users' machines and affects the privacy of users [11].

In an alternative online advertisement serving architecture, a website can embed advertisements locally and serve them to users, together with the content of a Web page. This ad serving technique is not very popular because it puts more workload on the Web servers compared to the previous approach, thus it does not scale as well. Some of the ad networks deploy this model when serving mobile advertisements, i.e., advertisements that are displayed on users' mobile devices. In this particular case it might be justified to put the overhead on Web servers rather than on users, because with mobile devices the available bandwidth and computational power is limited, the latencies are higher and the communication is more expensive for the user. The previous approach is also preferred by ad networks because direct communication of users to ad servers allows for better profiling of users' online behavior, thus better matching of ads to users interests and consequently higher potential revenues.

Targeted Advertising

A notable difference between online and traditional advertising (e.g., television, radio) is that online ads can be *targeted* to individual user's interests. To maximize the potential revenue, ad networks use *ad targeting techniques* to serve the ads that match users' interests. The

most popular ad targeting techniques are *contextual*, *behavioral* and *location-based* targeting. With contextual targeting, ads are related to the content the user is currently viewing. Behavioral targeting customizes ads based on users' *digital footprints*, i.e., information about the observed behavior of the users in the digital world, including usage of the Internet, mobile phone, etc. With location-based targeting, users receive location-specific ads on their mobile devices.

Targeted advertising aims at providing each user with the ads that best suit his interests. At ad servers, users' interests can be expressed with *keywords*. The ad server associates ads with each keyword and runs auction algorithms to select the most relevant ads and the order in which they appear on the Web page, with the goal of maximizing the profit of both advertisers and websites hosting the ads. In particular, small businesses find that online advertising offers maximum exposure for a minimal cost.

Revenue Models

There are three main revenue models: Advertisers may pay the ad network on a per *impression*, per *click* or per *action* basis.

In the per impression model, advertisers pay the ad network for the exposure of their ads to end users, i.e., there is a *cost-per-mille* (CPM) (cost to expose one ad to one thousand users). This model is widely used for *brand advertising*, i.e., increasing customers' awareness and ability to recall and recognize the brand, typically by displaying banner ads. Brand awareness is of critical importance as customers will not consider a brand if they are not aware of it. The impression-based model is an online counterpart to the traditional mediums for conveying a brand image to customers, such as print (where impressions are created by the placement of ads in subway cars, billboards, etc.) and television (where impressions are created by the emission of commercials). Thus, many advertisers are choosing impression-based online advertising as a way to establish their brand as a trustworthy friend to the consumer.

In the pay-per-click (PPC) model, advertisers pay the ad network a *cost-per-click* (CPC) for each user-generated click on an ad that directs the user's browser to the advertised website. From an advertiser's point of view, a click on an ad represents a user's choice. The benefit of the PPC model is that it offers instant feedback and the opportunity to measure the effectiveness of an advertising campaign. The success of an advertising campaign can be expressed with *clickthrough rate* (CTR). A CTR is obtained by dividing the "number of users who clicked on an ad" on a Web page by the "number of times the ad was delivered" (impressions). As of 2006, PPC started gaining prevalence over other revenue models. The trend continued over the years to reach approximately 62% of the advertising revenues that are priced based on this model in 2010, according to the Interactive Advertising Bureau [4].

If a click on an ad is followed by a predefined action on the advertiser's website (e.g., online purchase or registration for a newsletter), advertisers pay a *cost-per-action* (CPA) to the ad network. This model is widely used by many organizations primarily in service based businesses, rather than by companies who sell tangible "mail order" types of products online. These service-based businesses (e.g., insurance companies, mortgage companies, real estate brokerage, etc.) are aware that customers generally do not buy these kind of services on a first

impression. Therefore, these organizations using CPA media are instead generally far more interested in collecting initial, focused, targeted leads (i.e., potential sales contacts) from their advertising. As these markets are very competitive, businesses know and appreciate the fact that if they can get someone to join their e-mail list or find some other method of encouraging people to complete their online form, they would instantly have a significant head start on their competition. Therefore, they are willing to pay for CPA ads knowing that they are paying only for leads that are focused, refined and targeted for their business.

The ad network gives a fraction of the ad-generated revenue to the publisher who hosted the ad that resulted in an impression, a click or an action. These models provide incentive to participate in the ad serving system: advertisers earn the revenue created by ads, ad networks earn money for storing the ads and finding proper publishers to display ads, and the publishers earn money for hosting ads and directing users towards advertised websites. Users benefit from obtaining advertisements that are tailored to their interests.

1.1.2 Exploits of Online Advertising Systems

Surprisingly, online advertising and Web browsing still rely on the HTTP protocol, which does not provide any guarantees on the integrity or the authenticity of online content. Given the lack of security protocols, an adversary may perform ad fraud attacks to exploit the online ad serving system for its own benefit. Considering the amount of money at stake, the security of online advertising is becoming a pressing concern for advertisers, ad networks and publishers. As online advertising has emerged as the main source of revenues for most online activities, the attacks on online advertising systems could undermine the business model of the participating stakeholders and thus may represent a concern for the future of the Internet.

Adversary

An adversary launching an attack on an advertising system can take various forms in practice. We consider a *selfish* adversary intending to take advantage of the ad serving system: A selfish adversary exploits the system with the goal of diverting part of the ad revenue for itself. In contrast, a *malicious* adversary may perform any types of attack on the ad system, typically for nefarious purposes (e.g., launching a Denial-of-Service Attack, spreading a malicious software or hurting a competitor).

The adversary can be part of the ad serving architecture or part of the access network that provides Internet connectivity to end users (Figure 1.1). As discussed, all entities of the ad serving architecture benefit from the delivery of ads to end users, however there are various ways in which they may try to increase their revenue. In contrast, the access network that carries all users' traffic does not receive any ad revenue. Thus, the access network may also be tempted to tamper with the transiting data to generate benefits for itself.

Depending on the amount of resources and know-how available to the adversary, it can either attempt simple attacks from a single computer or it may deploy automated mechanisms to perpetrate large scale attacks from a number of machines worldwide. Today, botnets are a very popular tool for perpetrating distributed attacks on the Internet and are used very often to commit ad fraud. A botnet is a collection of software robots, or bots, that run autonomously and automatically. Bots are typically compromised computers running software, usually installed via drive-by downloads (i.e., downloads that happen without users' knowledge or

consent) exploiting Web browser vulnerabilities, worms, Trojan horses or backdoors, under a common command-and-control infrastructure. A bot master controls the botnet remotely, usually through a covert channel (e.g., Internet Relay Chat) for the botnet to be stealth and to protect against detection or intrusion into the botnet network. An adversary wanting to use a botnet for ad fraud may build its own or rent an existing botnet from another botnet master.

Although botnets typically enslave PCs to act like zombies in a botnet, a (believed to be the first) botnet of compromised wireless routers was detected in 2009 [2]. The botnet was used to launch a Distributed Denial-of-Service attack on DroneBL, a distributed DNS Blacklist service. It was estimated that the botnet gained control of approximately one hundred thousand routers, targeting home routers that have Web interface and an SSH port directly accessible without requiring a password or with a weak username and password combinations. This problem was later solved with a firmware update. Once it gained access to the system, the botnet loaded a file that turned routers into bots. This example demonstrates that the botnet problem is not something that only affects PCs. An adversary may use *warkitting attacks* to subvert home wireless routers [23]. Warkitting refers to a drive-by subversion of wireless home routers through unauthorized access by mobile WiFi clients. It is shown that in practice an adversary can perform warkitting with low-cost equipment and that a large number of routers are susceptible to such attacks.

A botnet of wireless routers can perpetrate powerful man-in-the-middle attacks, as routers are in a position to *eavesdrop*, *alter*, *inject* and *delete* communications. It also has the advantage of having the bots almost always connected to the Internet (compared to the typical end-user machine that is connected to the Internet only from time to time). In addition, it is more difficult to detect that a device has been compromised, due to the lack of security software for such devices (e.g., no anti-virus software).

Ad Fraud

An online advertising system can be abused in many ways. We will focus on the ad fraud attacks that have been the most prevalent in practice and that yield monetary benefits for the adversary: click fraud, malvertising, inflight modification of ad traffic and adware. We also address possible countermeasures to these attacks.

1.1.3 Click Fraud

In each of the revenue models (i.e., impression-based, click-based and action-based) an advertiser who pays for his ads to be included in online content has a positive return on investment (ROI) only when genuine impressions, clicks and actions are generated by legitimate users. ROI is used to express the actual or perceived future value of a marketing campaign and is calculated as the ratio of the revenue gained or lost, relative to the initial investment. An adversary can simulate interest in ads (by creating illegitimate ad impressions, clicks or conversions in the corresponding revenue models) that provides advertisers with little or no ROI, because they are not a result of legitimate users being exposed to ads. We refer to this type of ad fraud as *click fraud*.

The two most occurring types of click fraud are: *publisher click inflation* and *advertiser competitor clicking*.

With a **publisher click inflation attack**, a publisher tries to over-report its contribution in exposing users to ads. As publishers are rewarded by ad networks proportionally to the number of impressions or user-generated clicks and actions on the ads included in the publisher's Web pages, they sometimes inflate the numbers in order to obtain more revenue from ad networks. To do so, they generate fraudulent impressions, clicks and actions for which advertisers are charged by ad networks, and the fraudulent publishers receive a share of that revenue.

With an **advertiser competitor clicking attack**, an advertiser tries to undermine the advertising campaigns of its competitors. In order to increase the visibility of its own advertisements, an advertiser may create artificial impressions, clicks or actions on advertisements of its competitors. If its competitor advertisers are charged for these, their daily budgets may be exhausted rapidly and the fraudulent advertiser's ads would have the advantage of being selected and served to legitimate users.

Depending on the revenue model, an adversary generates artificial interest in ads as follows:

- In the **impression-based model** an adversary generates fraudulent ad impressions by issuing HTTP requests for Web pages containing ads that users never see.
- In the **pay-per-click model** an adversary generates fraudulent clicks on ads by issuing HTTP requests for ad impression URLs, that were not generated by legitimate users.
- In the **conversion-based model** an adversary can produce fraudulent conversions by issuing HTTP requests that represent an advertiser-defined action, such as a subscription, in order to simulate the action of a legitimate user.

Fraudsters may generate ad fraud themselves and deploy a third-party or automated programs to do so. Automated ad fraud attacks very often rely on botnets. An example of a botnet click fraud in the PPC model is Clickbot.A, the botnet that executed a low-noise click fraud attack against syndicated search engines and was investigated in detail by Google [6]. The botnet consisted of over one hundred thousand compromised machines and it perpetrated a publisher click inflation ad fraud. The bot operator acted as a publisher and created several websites that contained links that eventually led to ads on which the clickbot would click.

Automated ad fraud attacks can also be executed without compromising the end users' machines. For example, in the PPC model an attacker can launch a stealthy, automated click-fraud attack called "badvertisement" where fraudulent clicks are generated on ads hosted by the attacker [7]. The goal is accomplished by corrupting the JavaScript required to properly include ads into Web pages and does not depend on any client-side vulnerability. The script causes an ad to be automatically clicked and processed by a client's Web browser. Consequently, the click is accounted for by the ad network, the advertiser is charged and part of the revenue is transferred to the fraudulent publisher. Badvertisement attack is also an example of a publisher click inflation ad fraud.

An attacker can also generate fraudulent clicks by tricking users with "clickjacking" attacks to click on ads. Clickjacking happens when the attacker uses multiple transparent layers of Web pages to trick a user into clicking on a button or a link on a hidden page when they were intending to click on the bottom visible page [19]. Therefore, the attacker can trick users into performing actions that the users never intended and thus "hijack" their clicks. The clicks can then be turned into fraudulent clicks on CPC ads. Figure 1.2 shows an example

of a clickjacking attack where a victim surfs the bottom page (a fraudulent site that launches the clickjacking attack, e.g., *myphotos.com*), while actually affecting the site in the top frame (e.g., Google search result page) that the victim does not see. In the example, we have made the top page partially transparent for the purpose of illustration, whereas in the actual attack the top page is invisible to the victim. When the victim clicks on the button ”Next“ to proceed to the following photo on the Web page of *myphotos.com*, the click is hijacked and turned into a click on one of the CPC ads positioned on the right side of the Google search result page. In order to generate profit from clickjacking attacks, the fraudster may load his own website in the top frame (instead of Google search results as in the example) and turn hijacked clicks into clicks on CPC ads that appear on the fraudster's Web pages (i.e., perform a publisher click inflation attack). Alternatively, the fraudster may load Web pages on which its competitors' ads appear and generate fraudulent clicks on these (i.e., perform an advertiser competitor clicking attack). Clickjacking is possible because of Web browser vulnerabilities and an interested reader can find details about countermeasures in [19].

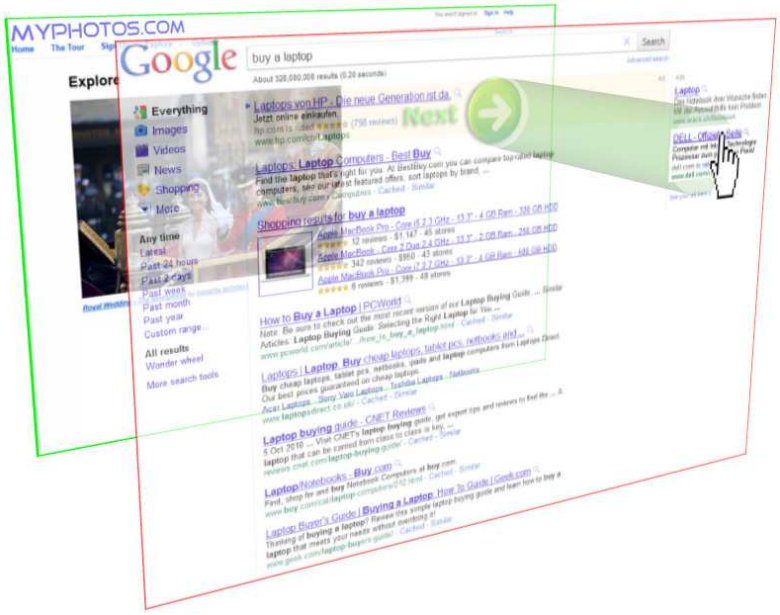

Figure 1.2 Clickjacking attack. In a clickjacking attack, a victim browses a Web page (in this example *myphotos.com* in the bottom frame) that loads an invisible top frame (in this case a Google search result page) and tricks the victim into clicking on the bottom frame while actually affecting the site in the top frame. We have made the top frame partially transparent for the purpose of illustration, whereas in the actual attack the top page is invisible to users. When the victim clicks on the button “Next” to proceed to the following photo on the *myphotos.com* page, the click is hijacked and turned into a click on a CPC ad on the invisible Google search result page.

Fraudulent clicks have a negative effect on advertisers' returns on investment and ideally, the ad network would detect all of the fraudulent clicks, mark them as *invalid* and not charge

advertisers for those clicks. To avoid the detection and ensure the revenue, the fraudulent clicks should be indistinguishable from the legitimate ones generated by users such that ad networks charge advertisers and share the revenue with publishers. That is why the fraudsters try to generate behavior patterns that resemble the behavior of legitimate users. Consequently, it is not possible to know with an absolute certainty whether a click is fraudulent or legitimate. Therefore, in order to preserve a good user experience even when a click is marked invalid, the user agent is still redirected to advertiser's website.

Estimates of the extent of the click fraud vary widely, and this is a subject of much discussion among advertisers and PPC search engines. According to Adometry (formerly, Click Forensics, Inc.), a company that performs ad traffic quality control, the click fraud rate declined in the fourth quarter of 2010 to 19.1% of the clicks being fraudulent, compared with 22.3% in the third quarter of 2010 [3]. Although this is an improvement, overall click fraud levels are still higher than the rate of 15.3% seen in 2009. However, Adometry's CEO says that this trend might not last: "While the overall click fraud rate dropped last quarter for PPC advertising, we saw the emergence of new schemes focused on display advertisements." [3].

We next present a case study of an ad fraud scheme that targets websites with display advertisements.

Case Study: Advertisers Scammed by Porn Sites

A case of click fraud reported in 2011 is a very good example of how a fraudster may orchestrate a large scale automated attack in a way that is difficult for ad networks to distinguish fraudulent clicks from legitimate clicks, thus producing high revenue for the fraudster.

The overview of the scheme is presented in Figure 1.3 . The fraudster hosts a website accessible at *www.mainsite.com* that contains links to pornographic video websites. In order to attract visitors to its website, the fraudster participates in the traffic exchange with a popular pornographic website (e.g., *www.pornsite.com*). In this traffic exchange, *www.pornsite.com* sends traffic of its legitimate visitors to *www.mainsite.com* for monetary remuneration. The traffic exchange is made possible by a man-in-the-middle website (*TrafficHolder.com*) that provides a catalogue of the traffic and corresponding prices that one can buy. The scheme then executes as follows:

1. To implement the traffic exchange, when legitimate users visit *www.pornsite.com*, the site opens a pop-under window and loads the fraudster's website *www.mainsite.com*, which generates traffic at the fraudster's site. According to the agreement, *www.pornsite.com* in return receives money from the fraudster. By cooperating in this way with popular websites, the fraudster is able to obtain millions of unique visitors for its own website.
2. When *www.mainsite.com* loads in the pop-under window, it generates a number of invisible zero-sized (i.e., 0x0 pixel) iFrames.
3. Each of the iFrames will load one of the *parked domain* websites registered by the fraudster. Parked domain websites are single page websites that typically do not have any content on these domains. These domains may be reserved for future development or to protect against the possibility of cybersquatting, i.e., registration of Internet domain names that contain trademarks with no intention of creating a legitimate website, but instead of selling the domain name to the trademark owner. Domain parked websites typically display advertisements and thus generate revenue for the registrant. In this scheme, the

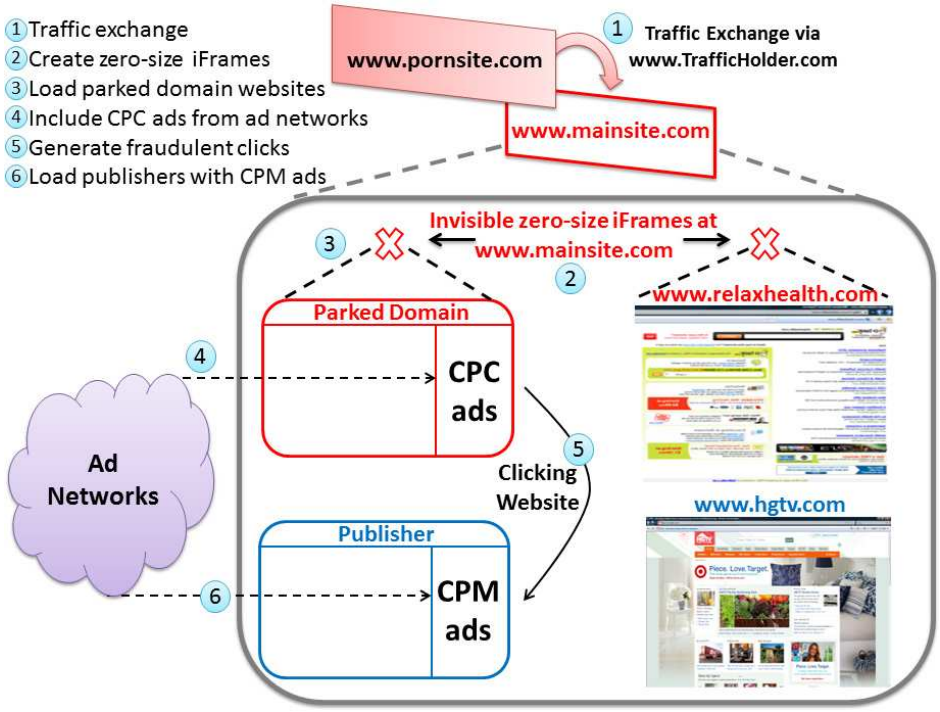

Figure 1.3 Schema of a click fraud attack. A fraudster buys legitimate traffic from a pornographic website in order to generate traffic at its own website (step 1) and produce legitimate-looking click traffic on ads. The fraudster's website creates a number of invisible iFrames (step 2) that load parked domain websites (step 3) owned by the fraudster and hosting CPC ads (step 4). In collaboration with clicking websites, the parked domain websites produce fraudulent clicks on their CPC ads (step 5). Fraudulent clicks result in loading of legitimate big-name-brand publishers with their own CPM ads (step 6). Reputable advertisers that pay for their ads to appear on quality publishers' websites have their ads appear within pornographic websites, which enables AdSafe to detect the fraud.

parked domains loaded in invisible iFrames are all registered by the fraudster and they all include advertisements. The domain names do not seem suspicious and are not related to pornographic websites (e.g., *www.relaxhealth.com* or *styleandmore.net*). This is important as most of the ad networks are not likely to include ads in pornographic websites.

4. Parked domain sites with corresponding advertisements are loaded in invisible iFrames.
5. A number of clicks on ads occur. To generate the clicks, the fraudster can simply deploy one of the "clicking websites" that already have such techniques.
6. The fraudulent clicks on the ads in the parked domains will eventually result in loading one of the big-brand-name publishers (e.g., HGTV) with its own CPM advertisements.

How does this scheme actually generate money for the fraudster? The monetization scheme is represented in Figure 1.4 . It is important to note that the big-brand-name publishers have a dual role, acting as (i) *CPC advertisers*, paying ad networks to include CPC ads with links to their websites into online content and (ii) *publishers*, collaborating with ad networks to host ads of CPM advertisers.

1. The fraudster generates fraudulent clicks on CPC ads of big-brand-name publishers that appear on his parked domain websites.
2. Ad networks charge the big-brand-name publishers (now playing a role of CPC advertisers) for the corresponding fraudulent clicks.
3. Ad networks pay a percentage of the CPC revenue to the registrant of the parked domains where the fraudulent clicks occur, i.e., to the fraudster.
4. By receiving traffic from the parked domains, ad impressions on big-brand-name publishers' Web pages are generated. For these ad impressions the publishers (now acting indeed as publishers hosting the ads) will obtain the revenue themselves. For this reason the fraudster cleverly targets only big-brand-name publishers that sell pay-per-impression and video ads and do not measure conversions, because for them only impressions count and any traffic is good. If the fraudster tries to load an e-commerce site that actually checks the quality of the traffic, this scheme would be detected. In addition, one more reason not to be suspicious is that the traffic towards publishers originates from legitimately-sounding domain names.
5. Ad networks charge the CPM advertisers for the impressions generated on big-brand-name publishers' websites.
6. Ad networks share the CPM revenue with the big-brand-name publishers.

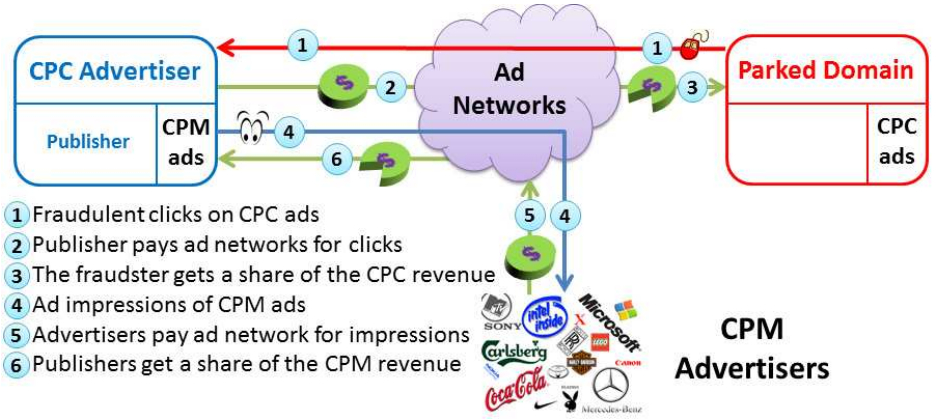

Figure 1.4 Monetization of the attack. The fraudster generates fraudulent clicks on CPC ads on his parked domain websites (step 1), for which the ad network charges the big-brand-name publishers (step 2) and shares part of the profit with the fraudster (step 3). The publishers are willing to pay as these clicks result in traffic on their websites that creates ad impressions (step 4) for which ad networks charge CPM advertisers (step 5) and share the profit with the publishers (step 6). In this scheme, the fraudster, the publishers and the ad networks make profit whereas the CPM advertisers lose revenue.

The ones that are hurt the most by this scheme are big-brand advertisers whose ads normally appear on reputable publishers' websites. The publishers and ad networks earn revenue by serving and displaying CPM ads, thus they are not concerned with the scheme. Therefore, big-brand advertisers are the ones who should fight the fraud. However, they do not want their reputation to be damaged by being associated with the fraud and in addition, the scheme does

not target a single party, it rather distributes the damage across a number of advertisers such that each individual does not have much incentive to fight the fraud itself.

This type of fraud has been detected by AdSafe, a company that ensures that brand advertisements appear with an appropriate content. Loading big-brand-name publishers' content and ads within the iFrames of the fraudster's *www.mainsite.com* has triggered an alarm at AdSafe as advertisements of reputable brands have appeared within frames of the pornographic website. As the fraud is based on the traffic of the legitimate users who visit *www.pornsite.com*, the click patterns appear as genuine (having different IP addresses, different Web browsers and at different times of the day) and are difficult to distinguish from legitimate clicks. Ad networks have a hard time detecting these clicks as fraudulent, because these clicks do not follow typical bot-generated click patterns.

A back-of-the-envelope calculation [9] shows that the fraudster may have earned between $50K to $700K per month with this scheme. Given that the scheme has been running for eight months, in total the fraudster may have earned $400K to $5M. This proves that a moderately sophisticated ad fraud scheme can result in substantial monetary gain. In addition, the fraudster does not violate law in most of the cases, but only terms of service of online advertising networks. Therefore, legal repercussions may not be sufficient to deter the fraudster from committing ad fraud, given the revenue at stake.

Countermeasures to Fight Click Fraud

As ad networks charge advertisers based on a number of impressions or clicks on advertisements, it might be counterintuitive for ad networks to have incentive to fight click fraud. In the short term, ad networks indeed earn more revenue by not filtering out fraudulent clicks. However, in the long run, the bad quality of the ad traffic may affect the reputation of the ad networks and result in poor performance of advertising campaigns, thus advertisers may stop investing in this form of advertising and publishers may not want to host the ad networks' ads within their content. Also, users may perceive ads as useless. Basically, if fraudulent clicks are not filtered out, an entire system may be ruined. An economic analysis [14], based on a game-theoretic model of the online advertising market, shows that ad networks that deploy effective countermeasures against click fraud gain a significant competitive advantage, as both publishers and advertisers will choose ad networks that offer the best return on investment.

The goal of the countermeasures deployed by ad networks is to make a successful attack more difficult or more costly for an attacker, rather than to absolutely eliminate click fraud. Most of the ad networks' techniques are kept confidential, otherwise it would be easy for the attacker to avoid detection. Typically, ad networks look for signals that indicate fraudulent click activity. Those signals may be different characteristics of HTTP traffic, anomalies in the ad click and conversion traffic, browser and user behavior that deviates from the expected behavior. Some techniques can be deployed to prevent click fraud as well, such as setting up a trust boundary between a publisher's content and ad slots on the publisher's Web pages. For example, assume that a publisher embeds a script into the content of his Web pages with a purpose of generating fraudulent clicks on ads that appear on these pages. If an ad network includes ads dynamically in the content in a way that the browser does not allow any script on other parts of the Web page to access the ads (e.g., by including ads in an iFrame), it may prevent a potential publisher click inflation attack.

In the case suspicious activities are noticed, ad networks may set an ad traffic monitoring team to investigate and potentially terminate collaboration with publishers on whose pages a lot of fraudulent clicks occur. Trusted third-party companies are employed to verify the practices of ad networks in examining ad traffic. Such companies are independent from ad networks and advertisers, and their job is to make sure that the clicks are properly labeled as legitimate or invalid, thus assuring advertisers that they are justifiably charged for the clicks.

1.1.4 Malvertising: Spreading Malware via Ads

Malvertising, one of the fastest growing security threats on the Web, is a class of online ads that attempt to infect an ad viewer's computer. It is particularly scary, because any site hosting ads and any operating system could be a potential target. Moreover, users do not even have to click on ads to trigger malware. For example, according to the report published by *Blue Coat's* research lab [13], an ad server can serve a JavaScript that, instead of fetching the legitimate ads, injects a hidden iFrame tag into the original Web page. The iFrame instructs the victim's browser to silently communicate with a malware server in the background, eventually resulting in the download of a PDF exploit file.

An advertiser can launch a malvertising attack, by adding its ad to a legitimate ad network. The ad network embeds the ad in publishers' websites and users click on it eventually. Publishers can also embed malvertisements into the content of their webpages to direct a user to the malicious website and install malwares.

Most of malvertisements are hosted by so-called *remnant advertising networks*. These networks sell empty advertising slots at the last opportunity. They aggregate advertisements and charge low rates. Consequently, there is less revenue and possibly less caution over the quality of advertisements.

Malvertisements can even appear at well-known websites, such as *New York Times* (reported on September 14, 2009 [10]), *Facebook* (reported on April 12th, 2010 [21]) and *London Stock Exchange* (reported on March 1, 2011 [8]). For example, visitors of the London Stock Exchange's website were exposed to malicious ads, that were designed to pop up fake security messages on their computers in order to sell anti-virus software.

Figure 1.5 shows a malvertisement that was embedded in Microsoft's search engine Bing (reported on July 3, 2010 by *StopMalvertising.com* [22]). The ad appears among the sponsored results and it refers to Macromedia Flash, while it points to *Flash.Player-Pro-Download.com* that does not belong to Adobe. Users who click on the ad go through *rc12.overture.com* and from there browsers are redirected to *player-pro-download.com*. This looks like a clear and nice website, but no mention of Adobe anymore. Instead there are promotions for online Flash games and professional Flash tutorials. If a user tries to download such software, browsers may issue security warnings because the content is not signed with a valid security certificate.

According to Dasient (an Internet security company that protects businesses from losses of traffic, reputation, and revenue caused by Web-based malware attacks) in the last three months of 2010 attackers managed to serve three million malvertisement impressions every day. In another study, Niels Provos (a researcher from *Google*) identified that about 2% of malicious web sites were distributing malware through advertisements, based on an analysis of about 2,000 known advertising networks [15].

Figure 1.5 Malvertisement promoting the latest version of Adobe Flash Player was embedded in Microsoft's search engine Bing. Bing included the malvertisement as one of the sponsored search results for the keyword search "adobe flash player". The malvertisement thus appeared in a colored box that marks sponsored links on the top of the results page. We single out the malvertisement with the red rectangle and a danger symbol. Web browsers cannot distinguish malvertisements from legitimate links and warn the users.

Countermeasures to Fight Malvertising

Appropriate and regular checks of advertisements are the best way to avoid malvertising. The publishers and ad networks should perform regular checks to verify the advertising content providers for any kind of active or malicious code. If they detect any unexpected or unwanted behaviour such as automated redirections, they should not publish the ads to the end users. In June 2009, *Google* launched a new search engine called *investigative research engine*, publicly available at *www.Anti-Malvertising.com*. This is to help ad network partners, identify potential providers of malvertising. The Internet users should also install and update appropriate anti-malware softwares on their machines to minimize the risk.

1.1.5 Inflight Modification of Ad Traffic

A novel type of ad fraud has appeared, consisting in the inflight modification of the ad traffic itself. A prominent example is the *Bahama botnet*, in which malware causes infected systems to display to end users altered ads, as well as altered search results (e.g., Google) [1]. The difference, compared to the traditional click fraud where ad networks may even earn revenue from fraudulent clicks, is that the traffic and the revenue is diverted from ad networks.

In the case of the Bahama botnet, compromised machines take their users to a fake page that looks just like the real Google search page and even returns results for queries entered into its search box. The attacker redirect users' traffic to a fake website by corrupting the DNS

translation method on the infected machines. As a result, the domain name "Google.com" is translated to an IP address that is not owned by Google, but by an attacker. When a user enters a query into the search box on what they believe is a Google server, the traffic actually goes to the fraudulent server that pulls the search results for the given query from Google, meddles with them (notably, it turns organic search results into paid links) and sends the results back to the user. Consequently, the results displayed are different from what they would otherwise be. A click on an "organic" link (in this case is actually a masked CPC ad) will result in a paid click through several ad networks or parked domains. Advertisers will be charged and the click fraud has occurred. Essentially, the Bahama botnet diverts the traffic and the revenue from major ad networks (e.g., Google) and redirects it to smaller ad networks and publishers.

Instead of compromising the users' machines, an attacker may also rely on botnets of compromised wireless routers [2]. Once a wireless router is infected with a malware and turned into a bot, the botnet master can instruct the bot to perform inflight modifications of the traffic that passes through the router. Many public hot-spots rely on a similar business model: providing free Internet access to users and in return generate revenue by embedding ads into the users' traffic.

There are reports of similar behavior of some ISPs [16, 27]. TrendWatch, the malware research team of Web security company TrendMicro, has investigated the practices of an Estonian ISP that was replacing ads included in the Web pages users were browsing [18]. The ISP was in charge of a number of DNS servers and was redirecting ad traffic from legitimate ad servers (e.g., Google ad network) to the servers of its choice. Consequently, users received ads that websites did not intend to show to its visitors. The investigation shows that thousands of ads were replaced per day, which implies that a significant ad revenue was diverted from legitimate victim ad networks.

The consequence of inflight modification of the ad traffic is that when users click on altered ads they generate revenue for the attacker instead of the legitimate ad network. Thus, the modification of the ads undermines the business model of ad networks. In addition, such attacks may also negatively affect the security of end users (malvertisement may be included instead of legitimate ads), the reputation of websites and the revenue of legitimate advertisers.

Countermeasures to Fight Inflight Modification of Ad Traffic

In order to prevent inflight modifications (or in general, man-in-the-middle attacks) well-known solutions consist in deploying authentication and data integrity mechanisms to help guarantee the end-to-end security of communications, as done with HTTPS [17]. Nevertheless, such mechanisms have various drawbacks that hinder their large-scale deployment. First, the protection of the Web content relies on cryptographic operations that induce a large computation cost on servers [17]. Second, these authentication mechanisms make use of digital certificates to enable the authentication of Web servers. Digital certificates are inherently expensive because trusted certification authorities must manually verify and vouch for the identity of Web servers. If a website owns a certificate issued by a trusted certification authority, then a chain of trust can be established and Web browsers can authenticate the website. Alternatively, in order to avoid such costs, website administrators often choose to use self-signed certificates. This allows website administrators to produce certificates themselves instead of relying on third-party certification authorities. However, self-signed certificates could be tampered with and may not protect against man-in-the-middle attacks. A Web browser cannot

trust the identity of a website based on a self-signed certificate and it requires users to make a decision whether they trust the corresponding server or not. However, it is hard for end users to understand how certificates work and to validate a given certificate, which often results in users communicating with a malicious server. In order to help users properly verify self-signed certificates, the system of network notaries is built to monitor consistency of Web servers' public keys over time [26]. However, this solution also has several drawbacks [26].

For these reasons, various alternative approaches are proposed to protect Web content in an efficient fashion [12, 16, 24]. Previous work suggests encrypting all Web communications using opportunistic encryption [12]: a secure channel is set up without verifying the identity of the other host. This provides a method to detect tampering with Web pages, but only for expert users who know how to check certificates. But, it does not defeat man-in-the-middle attacks because an adversary can still replace the certificates used for authentication to impersonate websites. Another approach focuses on the protection of Web content integrity by detecting inflight changes to Web pages using a Web-based measurement tool called Web tripwire [16]. The Web tripwire hides javascript code into Web pages, which detects changes to an HTTP Web page and reports them to the user and to the Web server. Web tripwire offers a less expensive form of a page integrity check than HTTPS but, as acknowledged by the authors, is non-cryptographically secure. A secure scheme that relies on cooperation between Web servers and ad networks is also proposed as a solution to thwart inflight modification of ad traffic [24]. This solution relies on the fact that most of online advertising networks own digital authentication certificates and can become a source of trust, needed to provide authenticity and integrity of the traffic.

Implementing the proposed solutions to protect against inflight modification of ad traffic incur a cost for ad networks and publishers. However, an economic analysis [25] that uses a game-theoretic model to analyze the interactions of an ad network and an ISP that performs inflight modification of ad traffic shows that, under certain conditions, investing into security of advertising systems is the best strategy for ad networks.

1.1.6 Adware: Unsolicited Software Ads

The term adware refers to any software that displays advertisements without users' permission [5, 20]. They are often designed to present advertisements according to the websites users visit. Adwares are produced by advertisers or by publishers of free software. Accordingly, adwares can broadly be divided into two main groups.

The first group of adwares are published for users who do not wish to pay for certain software. Many programs, games or utilities are ad-supported and distributed as adware (or freeware). If users purchase a registration key, they can disable displays of ads. The ads should also disappear as soon as the user uninstalls the software. In this case, adware is usually seen by the developer as a way to recover development costs, and in some cases it may allow for the software to be provided to users free of charge or at a reduced price. The income derived from presenting advertisements to users may allow or motivate the developer to continue to develop, maintain and upgrade the software product. As an example, the *Eudora* mail client displays advertisements as an alternative to shareware registration fees.

The second group of adware can be described as a form of *spyware* that collects information about users in order to display advertisements in the Web browser. In other words, it

displays advertisements related to the data it collects by spying on users. When adware becomes intrusive like this, it can be categorized as *spyware* and users should avoid it for privacy and security reasons. In this case, adware can intercept all information that users enter via the Web, add unauthorized sites to desktops and Internet favorites, track and monitor browser activity or attach the unwanted toolbars and searchbars to browsers without users' knowledge or approval. Moreover, the personal information can be sold to other parties without users' knowledge or consent. Finally, adware can hijack the default homepage and settings so the user cannot change them.

As an example, *YapBrowser* is an adware (spyware) that served unsolicited, aggressive advertisements, redirected users to undesirable websites, and modified essential system settings. This product was designed to be illegally installed on users' computers in order to make profit for spyware and adware creators. It must be noted that YapBrowser was bundled with the *Zango* software, a software company that provided users access to its partners' videos, games, tools and utilities in exchange for viewing targeted advertisements on their computers. In June 2006, YapBrowser was acquired by UK's *SearchWebMe*. *SearchWebMe* assures that the new *YapBrowser* download does not contain any adware or harmful applications. *Gator Software* from *Claria*Corporation and *Exact Advertising's BargainBuddy* are two other famous adwares in this category.

Countermeasures to Fight Adware

Users should avoid visiting untrusted websites because they are mainly delivering adware and spyware to unsuspecting users. Moreover, they can also install and update regularly anti-adware softwares. Finally, they should carefully read the terms of use for free software as they potentially install the adware as well. Note that it is required by law to state whether or not software has adware bundled with it.

1.1.7 Conclusion

Internet economy relies on online advertising as the main business model for monetizing online content. Given the ad revenue at stake and the lack of legislation against ad fraud in many countries, fraudsters have economic incentive to engage in fraudulent activities and exploit online advertising systems.

This chapter provides a detailed description of existing online advertising systems and their vulnerabilities. We explained how fraudsters can exploit these vulnerabilities and launch ad fraud attacks that we broadly divide into four main categories: *click fraud*, *malvertising*, *inflight modification of ad traffic*, and *adware*. For each type of attack, we presented techniques that fraudsters deploy to make profit from the advertising systems. In particular, we presented a case study of a click fraud attack that made substantial amount of money (up to $5M) for the fraudster while stealthily running for eight months. We discussed challenges of ad fraud detection and mitigation as well as several deployed countermeasures. However, much research is still needed in order to design more robust countermeasures and protect the Internet business model.

References

1. Botnet caught red handed stealing from Google. http://www.theregister.co.uk/2009/10/09/bahama_botnet_steals_from_google.
2. Network stealth router-based botnet. http://dronebl.org/blog/8.
3. Adometry. Adometry click fraud index. http://www.adometry.com/media/press/release.php?id=1, 2011.
4. I. A. Bureau. IAB Internet Advertising Revenue Report, 2010 Full Year Results. http://www.iab.net/media/file/IAB_Full_year_2010_0413_Final.pdf, 2011.
5. E. Chien. Techniques of adware and spyware. In *the Proceedings of the Fifteenth Virus Bulletin Conference, Dublin Ireland*, volume 47, 2005.
6. N. Daswani and M. Stoppelman. The Anatomy of Clickbot.A. In *In USENIX Hotbots07*, 2007.
7. M. Gandhi, M. Jakobsson, and J. Ratkiewicz. Badvertisements: Stealthy click-fraud with unwitting accessories. *Journal of Digital Forensics Practice*, 1(2), 2006.
8. G. V. Hulme. Malvertising Continues to Pound Legitimate Web Sites. http://www.csoonline.com/article/675064/malvertising-continues-to-pound-legitimate-web-sites, 2011.
9. P. Ipeirotis. Uncovering an advertising fraud scheme. or "The Internet is for porn". http://behind-the-enemy-lines.blogspot.com/2011/03/uncovering-advertising-fraud-scheme.html, 2011.
10. B. Johnson. Internet Companies Face Up to Malvertising Threat. http://www.guardian.co.uk/technology/2009/sep/25/malvertising, 2009.
11. B. Krishnamurthy, D. Malandrino, and C. E. Wills. Measuring privacy loss and the impact of privacy protection in Web browsing. page 52. ACM Press, 2007.
12. A. Langley. Opportunistic encryption everywhere. In *W2SP*, 2009.
13. C. Larsen. Exploiting Trust in Advertising Networks. http://rocket.bluecoat.com/blog/exploiting-trust-advertising-networks, 2010.
14. B. Mungamuru, S. Weis, and H. Garcia-Molina. Should ad networks bother fighting click fraud? (Yes, they should.). Technical Report 2008-24, Stanford InfoLab, July 2008.
15. N. Provos, P. Mavrommatis, M. Rajab, and F. Monrose. All your iFrames point to us. In *Proceedings of the 17th conference on Security symposium*, pages 1–15. USENIX Association, 2008.
16. C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver. Detecting in-flight page changes with Web tripwires. In *Proceedings of the 5th symposium on Networked Systems Design and Implementation NSDI 08*, 2008.
17. E. Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2000.
18. T. M. T. Research. A cybercrime hub. http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/a_cybercrime_hub.pdf, 2009.
19. G. Rydstedt, E. Bursztein, D. Boneh, and C. Jackson. Busting frame busting: a study of clickjacking vulnerabilities at popular sites. In *in IEEE Oakland Web 2.0 Security and Privacy (W2SP 2010)*, 2010.
20. G. Shaw. Spyware & Adware: the Risks Facing Businesses. *Network security*, 2003(9):12–14, 2003.
21. SPAMfighter. Malvertising attacks on facebook farm town players. http://www.spamfighter.com/News-14247-Malvertising-Attacks-on-Facebook-Farm-Town-Players.htm, 2010.
22. Stopmalvertising. Sponsored malvertisement for adobe flash player. http://stopmalvertising.com/malvertisements/sponsored-malvertisement-for-adobe-flash-player.html, 2010.
23. A. Tsow, M. Jakobsson, L. Yang, and S. Wetzel. Warkitting: the Drive-by Subversion of Wireless Home Routers. *Journal of Digital Forensics Practice*, 1(3):179–192, 2006.
24. N. Vratonjic, J. Freudiger, and J.-P. Hubaux. Integrity of the Web Content: The Case of Online Advertising. In *Usenix CollSec'10*, 2010.
25. N. Vratonjic, M. Raya, J.-P. Hubaux, and D. C. Parkes. Security Games in Online Advertising: Can Ads Help Secure the Web? In *WEIS 2010*, 2010.
26. D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *USENIX Annual Technical Conference*, 2008.
27. C. Zhang, C. Huang, K. Ross, D. Maltz, and J. Li. Inflight modifications of content: Who are the culprits? In *Workshop of Large-Scale Exploits and Emerging Threats LEET 11*, 2011.