Term Research Project: Final Draft
SRA 221 Section 2, Larry Garvin, Professor
Project Title: The Malvertising Problem: A Survey and Modest Proposal

Team Members' Names: Matthew Adams,
Sarah Hoover,
J Randall Miller

a. Abstract

This is a survey of research upon the growing threat of *malvertising*, or the use of online advertising for the distribution of malicious software and the commission of fraud. After examining the structure of the online advertising industry and its prototypical transaction, the manner in which its vulnerabilities are presently being exploited by attackers, and the technical countermeasures that are currently available to mitigate the threat, this survey concludes that although the technical obstacles to containing this threat are significant, web publishers and advertising distributors need to redouble their present efforts. A framework of public policy to encourage these efforts is also presented.

b. Introduction: Web Ads Bring You Instant Karma

Advertising is the economic lifeblood of the internet. For online giants like Google, Facebook and Yahoo!, advertising revenues are the mainstay of corporate cash flow, and without advertising revenue many of the popular online websites that now supply news and other content to users "for free" would probably soon disappear. (Zarras *et al*., 2014). Continuing its persistent trend of strong annual growth, internet advertising generated $49.5 billion of revenues during 2014, increasing by 15.6% from its total in 2013. (Pricewaterhousecoopers, 2015). The industry is concentrated, the ten largest ad-selling companies having accounted, over the last decade, for 69% to 74% of annual revenues. (Pricewaterhousecoopers, 2015).

Industry leaders are now fiercely competing for a lion's share of business in the fast-growing mobile advertising sector, whose revenues are generated by content, often location-sensitive, that is delivered to consumer smartphones and tablets. The $12.5 billion in 2014 revenues from online mobile advertising, constituting 25% of the industry total, represents a remarkable increase of 76% from their level in 2013. (Pricewaterhousecoopers, 2015). Mobile advertising creates a unique and variable environment for the problem that is the subject of this paper: because Apple restricts advertising that appears upon the devices that it manufactures to content delivered through its wholly-owned advertising network, for example, this problem is hardly experienced by the owners of iPhones and iPads at all (Mansfield-Devine, 2014), while Moonsamy and Batten (2014) report, by contrast, that in the freewheeling uncontrolled Android app market third-party advertising libraries incorporated into their Java packages by unsophisticated developers caused 10% of the apps that the authors studied to expose private data from users' smartphones without the users' prior consent. Phenomena peculiar to the mobile advertising ecosystem are, however, beyond the scope of this survey.

Another fast-growing segment of the online advertising industry that like mobile is distinguished by the characteristics of its end-user platform is the $4.1 billion advertising business in social media, which the Internet Advertising Bureau defines for statistical purposes as comprising social networking and gaming websites and apps, whether on desktop, laptop or strictly mobile devices. Social technologies may similarly present their own particularized mutations of the problem discussed in this paper, as certainly do the software "widgets" that the authors of blogs and similar Web 2.0 websites frequently install into their content-management systems to permit communication of ads along with special-interest news stories and other third-party content; but these latter special environments, which are often vulnerable to techniques like those of cross-site scripting (Sood & Enbody, 2011; Provos, McNamee, Mavrommatis, Wang, & Modadugu, 2007), will not be examined in this paper either.

This paper will concern itself more generally with internet display advertising, in which advertising banners, sidebars or inline images and text, often containing clickable HTML links or more complex interface components inviting user interaction, and sometimes incorporating Flash or other plugin-rendered video or audio presentations as well, are visually integrated with the principal subject matter of the web page upon which they appear. Web display advertising is distinguishable from online classified advertising, whose well-known exemplars include Craigslist and Yahoo! Classifieds, as well as from online search advertising of the kind made familiar by Google, which delivers sponsored advertisements logically related to the topic of a user-generated web search in the form of text and navigable links, rendered above and alongside the results generated from its cached data store by its proprietary search algorithm. Search advertising still comprises the largest share of online advertising, accounting for 38% of total revenues in 2014, in comparison to 28% for non-mobile display advertising and just 5% for web classifieds. (Pricewaterhousecoopers, 2015).

The logically necessary participants in any web display advertising transaction or "impression" are the audience, the publisher and the advertiser, but a complex structure of other entities has arisen to bring these first three into a dynamically created, ephemeral online relationship, allowing an advertiser to target its message to the instantaneous combination of an audience having specific demographic characteristics[1] with a publisher of specific subject-matter context. Historically there first arose advertising networks to broker the relationship between publishers and advertisers, aggregating an inventory of available space on existing web pages for which the networks solicit offers from advertisers at large; any one of these spaces that it sells a network will afterwards fill by mediating a series of HTTP-protocol requests, redirects and responses between a user's web browser, the publisher's web server, one of its own advertising servers, and a content server (today often part of a cluster operated by a major content distribution network like Akamai) at a URL that the advertiser or its creative agency may designate itself. (OpenX, 2013). To retrieve the demographic characteristics of the end user that the advertising network must know in order to match him with an interested display

[1] Here and in the sequel the term "demographic characteristics" must be broadly understood to include personal lifestyle, leisure interests and sociopolitical attitudes no less than "census" attributes like age, gender, place of residence, education level, marital status, family size, occupation and household income, all as revealed by or deduced from the application of sophisticated statistical data-analytics algorithms to a database consisting chiefly of web-browsing histories.

advertiser, the advertising network first queries a data warehouse, prototypically using for this purpose a cookie that uniquely identifies the user and that it stored on his computer during an earlier HTTP transaction.[2] (Angelia & Pishvar, 2013). This entire complex process that precedes the display of the advertisement on the web page typically requires well under one second to execute.

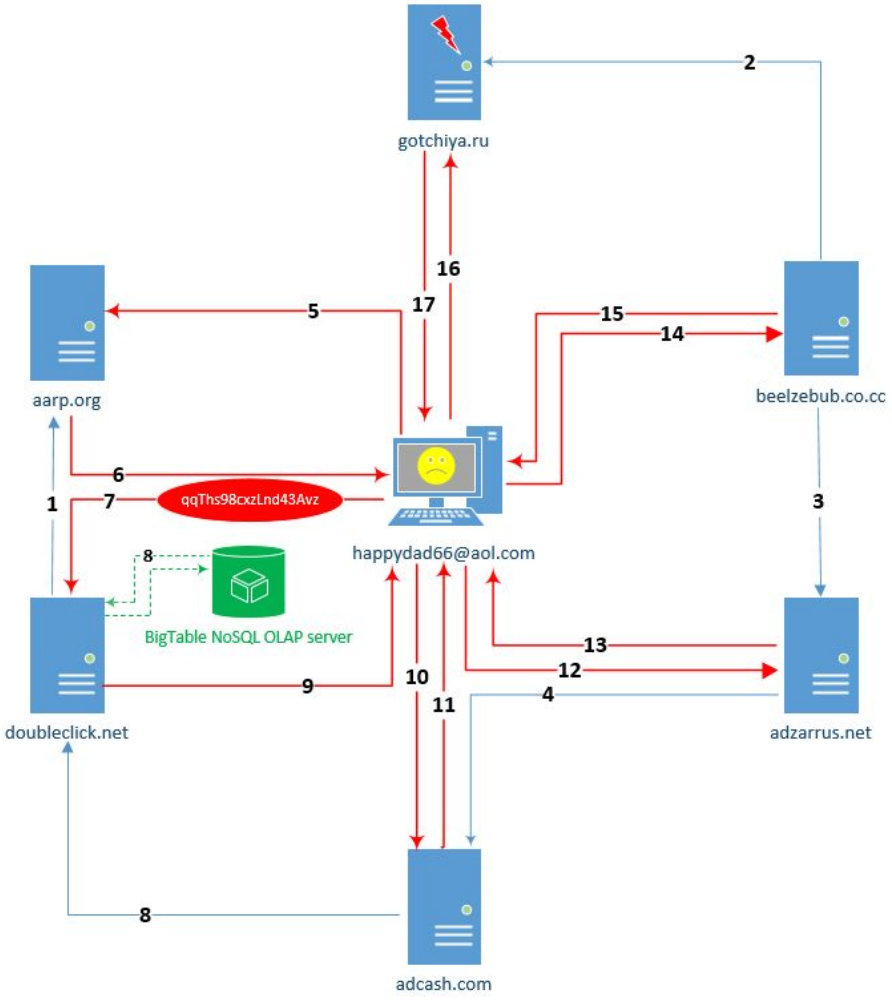
To overcome experienced business performance limitations of the advertising networks— chiefly due to the circumstance that no one of them grew quickly enough to enjoy sufficient market domination and breadth— there subsequently also evolved today's so-called advertising exchanges. An advertising exchange, after its insertion into the series of HTTP-protocol transactions described in the preceding paragraph, actually conducts a live auction of a space on a publisher's web page to interested advertising networks *in real time*, as a response to some user's having clicked an HTML link that results in a request for display of that same page; the conclusion of all of this business activity that precedes the display of the advertisement on the web page typically requires, once again, less than a second to execute. (OpenX, 2013). The complexity of the structure of participants engaged in a typical modern online advertising transaction has also been increased by the kind of specialization exemplified by the emergence of trackers, who accumulate, market and disseminate, on demand, the web-browsing history and other personal data of targetable online consumers, as well as by ad syndication, or the practice of advertising networks reselling and buying publisher inventory among themselves. (Li, Zhang, Xie, Yu, & Wang, 2012). Because of syndication more than a half-dozen networks are sometimes involved in the creation of a single impression. (Provos, Mavrommatis, Abu Rajab, & Monrose, 2008; Zarras *et al.*, 2014).

The word "impression" is a term of art in the advertising trade that refers to a single display of an advertisement's content to a single user, viewing a web page that supplies its context on a single occasion. An impression is one of the two units that may serve as the countable bases of a web publisher's compensation, the other being a single "click" by a viewer on an HTML link contained within the frame of the advertisement. (Zarras *et al.*, 2014). In so-called click fraud simulated clicks are generated within legitimate advertisements displayed on a web page controlled by the perpetrator of the fraud, and the malicious advertisements that are the subject of this paper are sometimes used for this purpose, too. But the practice of click fraud, hitting the advertising industry as it does directly in its pocketbook, has already been thoroughly researched, and effective means to detect and prevent it by now enjoy widespread deployment. (Pauli, 2015b). This paper will further discuss click fraud only incidentally, if at all.

Modern online advertising allows an advertiser to affordably reach a geographically dispersed audience of many millions, including visitors to the most well-known, popular and reputable of websites, while permitting an advertiser to precisely tailor its message to any identifiable segment of its entire targeted audience. Though the ability to reach a mass audience has been available for over half a century through the channels of modern telecommunications, and though an advertiser wishing to aim its message at a particular subgroup of the population was

[2] When a web browser transmits an HTTP request to a web server it attaches to its request any unexpired cookies on the user's computer that are associated with the base domain of the requested URL.

The Steps in an Illustrative Malvertising Attack



Legend

1. DoubleClick establishes an ongoing contractual relationship with AARP to compensate it on a cost-per-impression basis for online display advertising. Ad tags directing the audience to doubleclick.net are installed in different places on several of AARP's web pages, and pixel tags to manage tracking cookies are installed on other pages of the site.

2. A crew of Eastern Europeans operating a malware hub at the registered-for-free domain beelzebub.co.cc configures an exploit, hosted on a server under their control at the hijacked domain gotchiya.ru, that targets consumers browsing with Internet Explorer 6 on a Windows XP system The exploit installs a banking Trojan and runs a fake ad for a work-out-of-your-own-home scam.

3. An entity identifying itself as Creative Campaigns, LLC, an agency, contacts the Adzarrus advertising network and explains that it is representing an undisclosed client who wishes to purchase 10,000 banner display impressions targeting retired individuals who are interested in part-time work and who maintain at least one account with a financial institution at which they do regular online banking, for which the client is willing to pay $750 CPM. The network accepts a $7500 advance cash payment, and a tag URL for beelzebub.co.cc, identified as the client's content delivery network, is subsequently uploaded to the Adzarrus network through a portal.

4. Adzarrus has few relationships with actual web publishers and so maintains little or no inventory, seeking to profit instead by the practice of arbitrage. It packages the Creative Campaigns contract with others in its portfolio that are similarly targeted, and purchases publisher ad space to cover them all from the syndicating network adcash.com.

5. Seeking to compare the relative costs and advantages of Medigap versus Medicare Advantage insurance plans, AOL user happydad66 navigates to aarp.org with Internet Explorer 6 on his Windows XP Dell computer.

6. The web server at aarp.org responds to the web browser's HTTP request in (5) by sending HTML with which the browser can begin to render some of the site's home page. Within a banner iframe tag pair there is enclosed a script tag whose source attribute has as its value a URL served by the doubleclick.net domain; this is one of the ad tags installed in (1).

7. The ad tag mentioned in (6) causes happydad66's web browser to send an HTTP request for the script source to doubleclick.net. Because happydad66 has visited this domain many times in the past, there was already stored on his computer a cookie associated with that domain, and Internet Explorer automatically attaches that cookie to the HTTP request for the script tag's source.

8. When this last request reaches doubleclick.net, which operates here as an ad exchange, the exchange queries a database to see what it already knows about AOL user happydad66, using as an identifier the cookie mentioned in (7). Because this user has been referred to doubleclick.net in the past from many other pages upon which DoubleClick also has an advertising presence or stores a pixel tag, it develops that DoubleClick knows or can use analytics to deduce very much about happydad66 indeed, including the facts that he is interested in securing part-time work and does his online banking at citizensbank.com. After learning these and other facts about happydad66 that DoubleClick supplies at the real-time auction that it conducts, the advertising network adcash.com purchases the right to display a single impression to him in a banner space on the AARP home page for $0.55.

9. DoubleClick responds to the HTTP request in (7) by sending text/javascript content containing a document.write() command. Its argument will cause there to be written in the source of the aarp.com home page HTML code that displays a banner advertisement, for the source of whose images, text and other content the web browser is referred to adcash.com.

10-11.To retrieve this content the web browser visits adcash.com. This network redirects the browser instead to the URL furnished in (4) by Adzarrus, for whose account the syndicator purchased the impression at the auction in the first place.

12-13. Adzarrus redirects the web browser to the URL supplied to it by Creative Campaigns in (3), which now sends the browser to the hub server at the beelzebub.co.cc domain.

14-15. From the combination of its requested URL and the value of its header's referrer field the HTTP request sent to server at beelzebub.co.cc by happydad66's web browser allows the server to quickly associate the transaction with the group's Creative Campaigns initiative. Other values like the header's user agent field also permit a script on the server to realize that the fingerprint of the user's software fits the vulnerable profile targeted by the malware on gotchiya.ru, to which it accordingly sends his web browser with an HTTP 302 ("Moved Temporarily") redirection. A different fingerprint might have produced redirection to a harmless domain.

16-17. After his final arrival at gotchiya.ru AOL user happydad66 finally sees displayed in a banner running across the top of AARP's home page an image of a fit, smiling 65-ish male who proudly declares, " I earned $55,000 working at my computer from my home in just six months! Click on the link to learn whether you have the qualifications to do this yourself." If happydad66 accepts the invitation he will be given the opportunity to enter his own resume of personal credentials while comparing them to those of other successful part-time independent consultants. Whether he accepts the invitation or not, a customized Zbot banking Trojan has already been installed upon his computer in a drive-by download attack.

A Few Sample Ad Tags

The global technology company AppNexus conducts advertising exchange auctions. Here is an ad tag from a publisher's web page whose source URL directs a client browser to the AppNexus domain for retrieval of the JavaScript code that it will be called upon to execute:

```
<script src="http://ib.adnxs.com/ttj?id=1234" type="text/javascript"></script>
```

After AppNexus auctions the space that it references by the identifier 1234 it will return JavaScript code to the client's browser that looks very much like this:

```
document.write('<iframe frameborder="0" width="160" height="600" marginheight="0" marginwidth="0"
target="_blank" scrolling="no"
src="http://ad.yieldmanager.com/st?ad_type=iframe&ad_size=160x600&section=560122&m6li=1302146">
</iframe>');
```

If the publisher determines to ensure with its own HTML code that the ad will be confined to an iframe within its web page the first tag might instead have looked something like this:

```
<iframe src="http://ib.adnxs.com/tt?id=1234" width=160 height=600 scrolling=no frameborder=0 padding=0>
</iframe>
```

Source: AppNexus Industry Source Reference: Introduction to Ad Serving: Ad Tags. Retrieved October 9, 2015, from https://wiki.appnexus.com/display/industry/Ad+Tags.

Here is an interesting example, provided by OpenX, of a "simple" image ad tag that directs the client browser to an OpenX delivery server with the hostname corresponding to *delivery_server_domain*:

```
<a href="http://delivery_server_domain/w/1.0/rc?cs=4cbf34e0354e7&cb=INSERT_RANDOM_NUMBER_HERE"
target="_blank">
<img src="http://delivery_server_domain/w/1.0/ai?auid=8&cs=4cbf34e0354e7&cb=INSERT_RANDOM_NUMBER_HERE"
border="0" alt="">
</a>
```

In this HTML the image for which the browser is referred to the URL value of the "src" (source) attribute in the "img" (image) tag functions as a clickable link, to a second link-destination URL which in turn is the value of the "href" attribute of the image's enclosing "a" (link anchor) tag. The value of the "auid" parameter in the URL of the image source is the OpenX identifier for the particular ad unit, while the value of the "cs" parameter designates a user cookie specific to that ad unit. OpenX explains that when the client browser arrives at its delivery server the server will redirect the browser through an HTTP 302 status code to some URL appropriate to the particular ad being served.

When redirected away from the OpenX URL within the "img" tag the destination will of course provide the advertising image, or the "creative" as it is termed in the business. When redirected away from the URL within the "a" anchor link tag the value of its "target" attribute will instead ensure that the browser opens a new window, to display the new content that the OpenX web application causes to be summoned by the click. OpenX also explains, however, that the "rc" ("record click") action of its web application will record a user click and produce the new content for display only if the user-cookie value of its "cs" parameter is identical to the value first sent with the request for an ad image. The method here employed obviously serves to combat click fraud at the same time as it enables tracking the user's progress from the published web advertisement to the advertiser's online business premises.

The "cb" parameters whose values are given as INSERT_RANDOM_NUMBER_HERE are cache-busters. See footnote 3 in section (b) of the main text.

Source: OpenX Help: Ad Request API: Image ad tags. Retrieved October 9, 2015, from http://docs.openx.com/ad_server/#adtagguide_image.html.

able, long before the advent of radio and television, to attempt to do so by publishing in a magazine of limited subject-matter interest, the ability to precisely target a single message to the salient characteristics of its immediate individual recipient, on the fly, is new, and has been claimed to constitute the distinguishing economic essence of contemporary online advertising. (Goldfarb, 2014).

Unsurprisingly these same capabilities have made online advertising very attractive to criminals as a channel for distributing malware, or for perpetrating fraud. By placing a malicious advertisement on a popular web page the perpetrator can not only reach a large number of potential victims in a very short time, but can at the same time exploit the implicit trust that visitors reflexively repose in the owner of a reputable web domain. (Zarras *et al*., 2014; Kashyap, 2014; Pauli, 2015b). Meanwhile the ability to precisely target his victims not only improves an attacker's odds of success while sometimes enhancing the likely value of his ultimate reward, but it also allows him to better cloak the attack, and thereby avoid or postpone detection. (Kashyap, 2014; Pauli, 2015b).

Added appeal of the advertising channel to an attacker results from the circumstance that the complexity of the delivery structure described above, as well as the evanescence of the online advertisement itself— the same advertisement is rarely displayed by a web page even to the same end user twice, on a page refresh, for example, or a page-back navigation not retrieved from the web browser's cache[3]— make it difficult to forensically trace an infection back to its ultimate source, and complicate efforts to effectively deploy in the first instance techniques to detect and defeat the attack. (Kashyap, 2014; Pauli, 2015b). They also make it also make it more plausible for other participants in the advertising chain to deny any degree of responsibility for the ultimate result.

c. Body

(1) The Threat: Now You See Them, Now You Don't

It is currently estimated that malvertising, as the criminal abuse of online advertising is known, will cause over $1 billion in damage in 2015. (Pauli, 2015a). In 2013 the Online Trust Alliance[4] logged malvertising incidents serving some 12.4 billion impressions, an increase of more than 200% over traffic during the previous year, and in 2014 the security firm Cyphort found malvertising incidents to have increased by another 300%. (Pauli, 2015b). By crawling 53,100 publisher pages in the Alexa top 90,000 Li *et al.* (2012) already found that more than 1% of those well-maintained pages had been exploited to deliver malvertising attacks, and popular, well-known publisher sites that have so far served as their online landing pages include those operated by the *New York Times*, Reuters.com, *The Guardian*, the *Huffington Post* (several times), *The Atlantic*, the *Drudge Report*, Weather.com, and Major League Baseball. (Vance,

[3] To the URL within the ad tag returned to the user's browser by the publisher's web page there is commonly appended a string of random text, called a "cache-buster", that forces the display of a new advertising impression when the same web page is revisited or refreshed by the user.

[4] Better known members of this organization, formed originally to combat spam, include Microsoft, Symantec and the Direct Marketing Association.

2009; Bilogorskiy, 2015a; Santillan, 2015; Bilogorskiy, 2015b; Goodin, 2015; Kashyap, 2014; U.S. Senate, Permanent Subcommittee on Investigations, 2014). Although Zarras *et al*. (2014) found evidence that larger and well-established advertising networks seem better able, perhaps because of more extensive or technically sophisticated pre-scanning efforts, to avoid delivering malicious advertising to online publishers' webpage readers, Li *et al.* (2012) confirmed on the contrary what later news reports have also demonstrated to be true: major advertising networks that have been subverted for delivery of malicious content include all or some of those operated by Google — its subsidiary DoubleClick, for example, which maintains a presence on 80% of all ad publishing pages, as well as Google's AdSense and YouTube Ads— OpenX, Yahoo!, and AOL Advertising. (Kashyap, 2014; Mansfield-Devine, 2014; Gallagher, 2015; Segura, 2015; Bilogorskiy, 2015a; Santillan, 2015; Sinegubko, 2015; Bilogorskiy, 2015b).

Until recently the payloads of successful attacks, while various, have invariably disclosed an underlying financial motive, supporting the widespread assumption that the attackers are predominantly affiliated with organized crime. Attack payloads have included fake antivirus scareware, ransomware like Cryptowall, banking trojans like Zbot, and remote Trojan agents for botnet command-and-control. (Mansfield-Devine, 2014; Gallagher, 2015; Pauli, 2015a). Social engineering is also often deployed as an alternative to induce a user to part with identifying information and credentials that can be later sold on the black market, or put to profitable misuse. (Li *et al.*, 2012; Zarras *et al*., 2014). Pauli (2015b) summarizes reports from the French malware researcher Kafeine that operators trading on underground Dark Web forums are selling malvertising traffic at prices ranging from $4,000 per 100,000 to $70 for 1,000 multi-geographic hits (known as loads), with the premium operator GrandClix attracting, for each 1,000 hits, $450 in the United Kingdom or Australia, and $500 in the United States.

A "watering-hole" attack is an indirect attack upon an organization in which a vulnerable website frequented by the organization's employees is first compromised, and then afterwards exploited as a vector for delivery of malware to the information system of the organization itself; in November 2014, for example, Chinese attackers compromised Forbes.com to launch a watering-hole attack aimed at U.S. firms in the financial and defense sectors. (Rashid, 2015). Inasmuch as the targeting capabilities of web display advertising permit a threat agent to launch an indirect attack from the watering hole of his choice without having to somehow compromise it first, and moreover to precisely limit the scope of the exploit to those site visitors who fit some predetermined profile of the attackers' intended final victims, it was perhaps inevitable that malvertising would sooner or later be used to mount indirect attacks upon the systems of large organizations too.

And in late September 2014 the security firm Invincea indeed uncovered highly targeted malvertising watering-hole attacks against three U.S. military contracting firms, attacks whose goal was presumptively the exfiltration of internal defense secrets, and whose nature was thus presumptively cyber-espionage. (Auchard & Saba, 2014). Remarking an inversion of the historical pattern according to which cyber-criminals have traditionally learned from advanced nation-state actors and come to acquire their techniques, Invincea officials reported that the

malicious advertising rights in what it called Operation DeathClick were purchased through one or more exchanges in online real-time auctions, and involved use of an advertising network that allowed the cyber-espionage agents to direct their attack to website visitors whose internet-protocol addresses lay within ranges associated with the domains of the targeted military firms. (Invincea, 2014). In this white paper Invincea reports that the malware delivered in such sophisticated malvertising attacks is often hosted on innocent but compromised web servers running vulnerable WordPress or similar content-management software, and that the exploits are often taken down by their creators from these servers *within a few hours* after the exploits are first set up.

Malvertising frequently directs its victim to a phishing counterfeit website or a host serving malware by a technique known as link hijacking. Security constraints associated with the so-called same-origin policy will prevent JavaScript or other code running from within an advertisement's space on a web page from accessing its document object model (DOM) and interacting with other objects within the page, but JavaScript within the advertisement, by setting the standard top.location property while executing in the enclosing browser context of the publisher's web page, may redirect the user's browser to another page that is served from an entirely distinct domain. (Zarras *et al*., 2014). ActionScript code within a Flash file run by the browser's Adobe Flash plug-in may call the getURL() method of a MovieClip object to bring about the same result. (Ford, Cova, Kruegel, & Vigna, 2009). And the code or file that accomplishes the malicious redirection may be invisibly housed within a 1-pixel advertising iframe that is defined within the web page's HTML source.[5] (Gallagher, 2015).

Malware is usually delivered to an end-user's computer by a drive-by download, a form of delivery which is especially pernicious in that it requires no action at all to be taken by the victimized user. (Narvaez, Endicott-Popovsky, Seifert, Aval, & Frincke, 2010). After the user is redirected by some means to a server hosting the malware, vulnerabilities in the web browser itself, in built-in components like Microsoft's ActiveX web controls, or in installed third-party plug-ins like Adobe Flash or Oracle Java that share the browser's execution permissions and system-allotted memory space, are exploited within the advertisement's JavaScript code; this latter is executed by a JavaScript interpreter that is an integral part of the web browser application. (Cova, Kruegel, & Vigna, 2010). The download of the malvertisement's ultimate malicious payload that is thus brought about is often a two-stage process, meaning that a "dropper" with a smaller digital footprint is first pulled onto the system to be infected, where it may disarm critical host protection measures before commencing a second download of the attack's larger, featured malware package. (Narvaez *et al.*, 2010).

As described in Cova *et al.* (2010), the process that the host will launch when the exploit initially gains control of program execution is often defined in a sequence of binary executable machine-language instructions, known as shellcode, that is packed into a variable of the JavaScript string primitive type, to ensure that the elements of the entire sequence will be loaded into a contiguous portion of dynamic heap memory when the JavaScript string variable

[5] These 1x1 transparent so-called pixel tags are also used to initially place and afterwards read third-party tracking cookies, which act as "web beacons" of browsing activity, on the user's computer. This practice is not today deemed to be illegitimate or malicious.

is initialized by the browser's interpreter. To enhance the probability of a successful exploit multiple instances of the long shellcode string value, in a technique known as heap spraying, are successively pushed into the browser's heap by dynamic memory allocations, while at the same time the executable shellcode is interspersed with long sequences of instructions— a so-called NOP sled— that in themselves cause the CPU to do nothing, but create no error condition either. It is only after the host's RAM has first been prepared in a manner like this that some vulnerability in a software component is then attacked, in an attempt to transfer running program execution to some address within the infected web browser memory heap; in one well-known exploit, for example, JavaScript calls the LinkSBIcons() method of a SuperBuddy ActiveX component instance and passes a large integer argument to the method's single parameter. (Cova *et al.*, 2010). Similarly, the older CVE-2007-00671 vulnerability allows a running ECMA ActionScript to write to function pointers, return execution addresses or other critical memory by supplying a crafted integer value four bytes long or greater to the SceneCount field within the DefineSceneAndFrameLabelData tag of a Flash SWF multimedia file. (Ford *et al.*, 2009).

More than a third of all drive-by exploits are created with the use of commercially-available kits like Blackhole or its descendants, Cool, NeoSploit, LuckySploit, Styx, Angler or Nuclear. (Pauli, 2015b; Segura, 2015; Tubin, 2013; Gallagher, 2012; Li *et al.*, 2012; Cova *et al.*, 2010). Provos, Mavrommatis, Abu Rajab, and Monrose (2008) similarly found that within a subset comprising at least 25% of all malware distribution sites that they surveyed at least one binary was shared between any pair of sites, indicating a significant degree of content replication across domains. Findings like these might suggest that a method that checks hashed code fingerprints could be an effective measure for filtering malware, but the creators of malvertising exploits are not script kiddies, and typically demonstrate considerable sophistication by resorting to a battery of technical measures to evade detection. To frustrate the efforts of static, signature-based antivirus or IDS software filters, for example, code is commonly obfuscated by concealing the namespace identifiers of variables, functions and objects, or strings that represent interpretable sequences of scripted instructions, with mangling techniques that range in complexity from simple base-64 text encoding to full-scale, high-level encryption. (Ford *et al.*, 2009; Cova *et al.*, 2010).

Code syntax and structure are likewise frequently concealed by making use of the built-in eval() function whose implementation is required by JavaScript's ECMA standard, and which allows for dynamic interpretation and execution, in a running script, of code that is passed as a text argument of the function's parameter; to better disguise the purpose of their code the authors of exploits often create a layered series of several nested or stacked calls to the JavaScript eval() function, passing to each call as its argument a secondary call to a deobfuscator method defined elsewhere within the script. (Cova *et al.*, 2010). In exploits targeting the Flash plugin an exploit's ActionScript code is also commonly hidden within sections of a file meant to house video or images, instead of sections bearing the standard DoAction or DoInitAction tags where code is meant to be reposed, and then is later accessed and run by *jump* bytecode VM instructions; in versions of Flash as new or newer than version 9.0 binary data can moreover be successively pushed onto the running program's execution

stack and used there to initialize a ByteArray packed with executable bytecode instructions,[6] which is then passed as an argument to ActionScript 3.0's Loader.loadBytes() method. Sometimes the executable instructions are more simply stored within some file as a hex-encoded string, and a decoder with this file as an argument is passed instead to the loadBytes() method. (Ford *et al.*, 2009).

Techniques collectively known as cloaking are especially adaptable to the targeting capabilities of online advertising. Malvertising scripts can access values of the fields of the HTTP header, values of standard properties in the execution environment, and values returned by standard built-in JavaScript functions to determine the precise time and geographic location at which script execution is taking place, as well as to "fingerprint" the user's operating system, web browser application ("user agent") brand and version, and the list of installed web browser plug-ins. These values, augmented with the user profile data obtained by the use of tracking cookies and online advertising databases, are then used as bases for finely tuning the behavior of an exploit's script, so that its malicious behavior is exhibited only at a limited time and place of the attacker's own choosing, and only upon systems whose software configurations he knows in advance to be vulnerable. (Ford *et al.*, 2009; Cova *et al.*, 2010; Stringhini, Kruegel, & Vigna, 2013; Invincea, 2014; Pauli, 2015b).

Geographically limited, time-delayed and system-dependent script execution are commonly used to frustrate the pre-scanning detection efforts of the more careful advertising networks. In an interesting attempt to hoist defenders upon their own petards, scripts have also been known to incorporate blacklists of URL patterns at which malware detectors are believed likely to be found running; when launched at a URL that matches this pattern malicious script execution is similarly disabled. (Ford *et al.*, 2009; Kapravelos, Cova, Kruegel, & Vigna, 2011; Stringhini *et al.*, 2013). To complicate later forensic identification of the source of an attack, a script that has already executed on a user's machine will drop a dated cookie there that, when read by the script, prevents code execution on a second visit. (Ford *et al.*, 2009). And to frustrate later offline analysis of the body of the malicious script itself, the URL of the server site that hosts it is selected as a key for the encryption algorithm initially used to obfuscate the plaintext of the code, while the deobfuscator function is defined to use the URL of the script's current execution environment as its key for the decryption of ciphertext. (Cova *et al.*, 2010).

The internet domains from which malvertisers mount their attacks, as well as the network topologies into which these domains are frequently integrated, are also characterized by features intended, at least in significant part, to help them evade detection. To limit the attackers' up-front investment any domains created are short-lived and often free, established within top-level domains whose registrars impose few if any meaningful registration requirements, and the attackers can quickly abandon any one of them without cause for later regret. (Li *et al.*, 2012; Invincea, 2014). These domains are usually embedded, moreover, in application-layer topologies that exploit multiple HTTP redirections, often making use of a hub-and-spokes design that facilitates vulnerability-based central dispatching or cloaking while

[6] In a Flash exploit a ByteArray can instead be packed by this same method with native shellcode, which is then used as the fundamental element of a heap-spraying operation resembling the JavaScript operation described above.

rendering the entire structure more agile, able to respond nimbly and with resilience to domain-takedown defense reprisals. (Stringhini *et al.*, 2013; Provos, Mavrommatis, Abu Rajab, & Monrose, 2008).

(2) The Countermeasures: No Magic Bullet, Just a Poison Pill

Among the simpler of the technical means available to thwart malvertising attacks are the use of blacklists to block navigation to internet domains known to be malicious, and of anti-malware scanners to block the execution of code whose syntactical pattern matches a signature of some known exploit technique. In operation both of these methods are very fast and their use is comparatively more practical for that reason, but a weakness that is common to both of them is signaled by the use just made of the qualifier “known”: today new malicious internet domains appear and then quickly vanish at an ever-increasing rate, while signature-based software is impotent to detect brand new, “zero-day” techniques that exploit hitherto undiscovered vulnerabilities. Static scanning can also be easily frustrated by obfuscation techniques like those that were described above in this paper. (Cova *et al.*, 2010; Dai, Fyodor, Wu, Huang, & Kuo, 2012; Stringhini *et al.*, 2013).

In research results reported by Narvaez, Endicott-Popovsky, Seifert, Aval and Frincke (2010), only one out of the five popular antivirus products that the authors tested was able to detect 70% or more of the malware known to have been installed after test client systems visited an experimental set of malicious URLs. Detection rates invariably improved significantly when the signature files of these products were updated after an interval of 30 days and they were retested upon the same malware, but the study’s already discouraging results must be assessed in the light of one experimental circumstance that is especially alarming: the reported detection rates were produced by training pristine and fully potent sample antivirus software on file-system malware entities whose download and installation was deliberately first permitted by their client machines. Recall that most drive-by download attacks, as was already described in this paper, are conducted in two stages, and then consider that the study’s own analysis of a single low-profile, first-stage dropper discovered that the many preparatory actions that it performed when its parent process was launched on a client machine included disabling that host’s software firewall, and *any installed antivirus application* that it could find.

The client machines used in the just-described study were configured to act like honeyclients, a term which derives from use of “honeypot” to describe a counter-espionage or intelligence asset-recruitment operation in which a sexual or other temptation is used to entice a foreign national into some compromising situation that can afterwards be exploited for blackmail. This is a dynamic, behavior-based mode of malware analysis and detection; in a low-profile honeyclient like the open-source PhoneyC detection is still based upon the use of a signature file, but the patterns for which a match is sought are based upon the executable’s runtime behavior, such as particular sequences of identifiable built-in JavaScript function calls. (Qassrawi, 2011; Cova *et al.*, 2010). Like a static signature-based detector a low-profile honeyclient is fast, but unlike a static method it cannot so easily be frustrated by an exploit’s code obfuscation techniques. Unfortunately a dynamic low-profile honeyclient, again like a

static, signature-based method, is without means to detect a zero-day code exploit to which it has not already been introduced. (Cova *et al.*, 2010; Dai *et al.*, 2012).

This significant shortcoming is overcome by the use of a slower but more adaptive high-profile honeyclient like the open-source Capture-HPC client/server software suite. In a high-profile honeyclient a real or emulated instrumented web browser is equipped with a panoply of third-party plugin software and made to run inside a virtual machine, where it is then caused to establish HTTP connections with one or a sequence of web URLs that are suspected or must be proved innocent of hosting malware. Any exploit that the browser encounters is permitted or even encouraged[7] to run its full course, while the sandboxed client system is externally monitored for file-system changes, network activity, process launches, and edits of the Windows registry or other configuration files, all of which are meticulously captured and logged. Any unexpected system event that is not contained in a pre-established whitelist of normal system activity like the file-system reading and writing associated with routine web content caching is counted as conclusive evidence of an external attack, and records are consulted to map the event to the agent URL that is ascertained to have been its cause. After detection of an exploit the virtual machine— which is often one of a battery of clients, all of them controlled by a single server, that are crawling a large segment of the web— is rebooted, and a clean system image is restored. (Qassrawi, 2011; Dai *et al.*, 2012).

Note that while a low-profile honeyclient must at least *appear*, to any exploit that performs system fingerprinting for use in cloaking or selective activation, to possess that precise combination of operating system, browser and plugin software that the exploit judges to be vulnerable and will trigger its attack behavior, the same vulnerable combination must in fact be *running* on a high-profile honeyclient for detection to be theoretically possible, even when the malware is not itself environmentally aware. This is no trivial requirement, given the very large number of software configurations that are possible. (Cova *et al.*, 2010). Both types of honeyclient detectors, moreover, must be specially configured to trigger malware that delays its own execution with a timeout, or that first requires evidence of user presence like the mechanical motion of a mouse cursor across the screen surface. (Kapravelos *et al.*, 2011). And while a low-profile honeyclient can at least detect the kind of code behavior that results in a link hijacking, a high-profile honeyclient is entirely incapable of detecting a pure incident of malicious social engineering, because from the point of view of a client system such attacks involve no abnormal system events at all. (Stringhini *et al.*, 2013).

In case such complications are not troublesome enough, Kapravelos, Cova, Kruegel and Vigna (2011) describe a few sophisticated additional techniques— such as exploitation of the whitelists with which high-profile honeyclients must be equipped to allow them to recognize normal system behavior— that may also permit an exploit to cause damage that will be undetectable by a high-profile honeyclient. These are interesting in themselves as illustrations of the elaborate and often seemingly perpetual cat-and-mouse minuet of countermeasure and evasion that has come to be characteristic of modern security defense; most similar here to

[7] Le, Welch, Gao, and Komisarckzuk (2013) modify Capture-HPC to enable it to execute an exploit's shellcode and avoid a false negative result even when the attackers failed to accurately predict the address within the heap of the the first shellcode instruction or its NOP preamble, or the system altogether lacks the targeted vulnerability.

techniques that have already been described above in this paper is the host fingerprinting that might be performed to discover the presence of a running virtual machine, or of a browser make, configuration and version that is only an emulation.

An alternative both to static and to behavior-based detection that overcomes some of the described disadvantages of both is what is known as the anomaly-based approach to detection. In a scheme of this kind that is designed to look for anomalies in code or script that is subjected to its analysis, the very techniques that an exploit employs to evade detection by other methods— obfuscation and cloaking, for example, or the excessive use of code that is dynamically loaded and/or evaluated at program runtime— can form part of the basis upon which the code is flagged by the scheme as anomalous, and therefore probably malicious too. In methods using this approach a number of countable or measurable characteristics that are believed to possess high predictive value are preselected at the outset to serve as benchmarks, whose values will become the relevant properties of any object examined.

Harmless code may have perfectly legitimate reasons for ascertaining the properties of the environment in which it is executing, of course, or apparently even for resorting to obfuscation of its sense and purpose in life. In an anomaly-based method samples of “good” and “bad” objects must first be digitally subjected to a machine-learning algorithm, in which rules that attempt to reliably distinguish the two classes are iteratively generated and refined by the use of probabilistic decision trees. Cova, Kruegel and Vigna (2010) describe the creation and operation of such an anomaly-based tool, called JSAND, which is designed to detect and analyze malicious JavaScript code used in drive-by download attacks, a tool that has been incorporated into the freely available Wepawet web service maintained by the University of California at Santa Barbara.[8] The ten benchmarks whose values are examined by the rules of JSAND comprise: the number and target of redirections, and the length of each redirection chain (#1); differences in code execution that are determined by browser personality or history (#2); the ratio of string definitions to actual string uses (#3, with a high ratio being indicative of the probable running of internal code de-obfuscation routines); the number of dynamic code executions (#4), and the length of dynamically evaluated code (#5, measuring the length of string arguments sent to JavaScript’s built-in eval() function); the number of bytes allocated through string operations, including variable initializations (#6, with a high value tending to suggest heap-spraying operations); the number of strings that are likely to contain shellcode (#7, with likely shellcode being detected by interpreting the string as a sequence of two-byte Unicode character-codes, and then looking for the presence of non-printable characters); the number and type of instantiated plugins, or ActiveX controls (#8); the values defined for the attributes of these components, and the arguments sent to their methods' parameters (#9, looking for long strings used in attempts to overflow buffers, or large integers intended to represent an expected address in the memory heap); and sequences of component method calls that in combination may prove malicious (#10, exemplified by a plugin that downloads a file to the local file system and then runs an executable from the same system). JSAND measures the values of these benchmarks while running the subject JavaScript on an emulated browser that can impersonate any browser personality and its configuration of

[8] https://wepawet.iseclab.org/index.php.

components at will, while forcing the execution of any script function that may not otherwise have been called because of the values of cloaking parameters.

JSAND is able to achieve a detection rate that equals or surpasses other state-of-the-art tools, with few false positives. It can analyze about two samples per minute, a rate of performance that, although not blazing, substantially exceeds the rate achievable by a high-interaction honeyclient, and that can be improved by running the analysis on several machines in parallel. The tool will attempt to classify any exploit that it flags by reference to any corresponding vulnerability discovered in MITRE's CVE repository, and can be used to generate signatures for a dynamic low-interaction honeyclient like the open-source PhoneyC.

The UCSB Wepawet service also incorporates the OdoSwiff tool developed by Ford, Cova, Kruegel, and Vigna (2009) to detect and analyze malicious Flash advertisements, and the FlashDetect tool developed by Van Overveldt, Kruegel, and Vigna (2012) to augment Odoswiff for the second iteration of Flash Player's virtual machine (for ActionScript version 3.0, in the Flash Player version 9.0 and later).[9] Both of these tools resemble JSAND by sampling the values of a list of predetermined relevant code characteristics during actual execution that may be indicative of obfuscating or cloaking efforts, the presence of malicious shellcode, or the attempted exploitation of some inherent weakness. OdoSwiff and FlashDetect, however, use a combination of static and dynamic analyses, in the former case resorting to an open-source tool from an emulation library whose heuristics can detect shellcode even when hardened by encryption, and in the latter running SWF files on its own custom instrumented Flash player built to generate an execution trace. In arriving at its ultimate verdict FlashDetect relies upon a probabilistic Bayesian classifier, but both tools consider that a redirection to another URL that was not initiated by the user is alone sufficient ground to definitively classify a Flash SWF file as malicious.

Despite its better performance than a high-profile honeypot Wepawet is unfortunately still far too slow in its operation for use in the practical context of "live" web browsing, where a user with a broadband connection has come to expect that a web page will finish loading in about three or fewer seconds. "Offline" tools like Wepawet and Capture-HPC have proved useful in research, for conducting statistical surveys of the internet or for generating experimental sample sets. They can also be advantageously deployed to crawl the web in search of malicious URLs and domains for blacklisting, or to develop signatures for online use in detecting new malware family variants and zero-day exploits. These tools seem to be suitable, finally, for gatekeeping deployment by any advertising or content distribution network that serves as a portal through which advertising content is launched into the online HTTP stream, although there is little extant evidence to show that such prophylactic use for pre-scanning is

[9] In this more recent paper there are also described two new sophisticated forms of attack, the first of which exploits the just-in-time (JIT) compiler of the Flash version 2 virtual machine to cause it to compile arbitrary native shellcode (if read from the proper offset), and the second of which exploits a phenomenon called type confusion to entirely bypass the data-execution-prevention (DEP) and address-space-layout-randomization (ASLR) protections of modern systems to leak object memory addresses, read arbitrary memory addresses, and gain control of execution. FlashDetect can recognize both of these exploits.

as extensive as it should, by this late date, have long ago become. (Mansfield-Devine, 2014; Zarras *et al.,* 2014).

Faster speeds have been achieved, on the other hand, without at the same time incurring the limitations of signature-based methods, by imaginative techniques that focus the attention of some machine-learning algorithm of an anomaly-based method like JSAND upon the characteristic features, not of the malicious content that is delivered, but rather of the structures involved in its delivery. Li, Zhang, Xie, Yu, and Wang (2012) have developed a content-independent detection method called MadTracer that is based upon an ongoing probabilistic analysis of the topologies of the *ad hoc* network chains of HTTP redirection that are temporarily constructed in the process of delivering an advertisement, and of certain selected high-value characteristics of the several nodes that comprise these chains; quantified features that serve as bases for this anomaly-based method include: the lengths of the ICANN registration lives of domain nodes; the top-level-domain types of domain nodes (*e.g.*, ".co.cc" as opposed to ".com"); the functions of domain nodes (publishing node, advertising node— as established by a domain's presence on the EasyList or EasyPrivacy lists maintained by Adblock Plus— or node with an unknown function); the popularities (*i.e.*, rates of recurrence in distinct redirection chains) of domain nodes; URL suspiciousness (as established by a match to some regex pattern that suggests automated URL generation from a template, perhaps with an exploit kit); and the stability of the relationships between neighboring path nodes. MadTracer was reported to achieve success in detecting malvertising exploits well in excess of the rates achieved by Google's SafeBrowsing or Microsoft's (since-discontinued) Forefront API, with an acceptably low rate of false positives, while its performance overhead increased the average page loading time in a tested browser from about 3.5 to just 5.8 seconds.

Adopting a conceptually similar approach, Stringhini, Kruegel, and Vigna (2013) have used machine-learning algorithms to develop another anomaly-based but content-independent method that they named SpiderWeb. From a data source comprising the online histories of a large and diverse contingent of web browsing users, SpiderWeb collects, for the URL of any web page targeted for classification, the set of all chains of HTTP redirection that lead from any user's IP address to the targeted final URL, and from this set constructs a composite redirection network graph. Node characteristics captured for analysis are fewer in number than are those collected by MadTracer, but include a fingerprint of each user's software, geoIP node location data, and lexical URL features. With just these limited few characteristics and the topological features of the entire redirection graph SpiderWeb was surprisingly able to detect, after very limited training of the model, more than four out of every five of the malicious domains that were included in its experimental trial sample. Some of the characteristics that the authors found to characterize graphs that ended in a malicious domain included the uniform software configurations but diverse geographic locations of the users that finally arrived at the domain, the presence in the graphs of nodes that appeared to serve as hubs, long redirection chains with nodes located in several different countries, and the lack of looping inter-domain redirections along constituent chains.

The developers of SpiderWeb point out that its potential use is not limited, as MadTracer's is, to the detection of malicious advertising only, but a constraining logic common to both

schemes of detection especially underscores the troublesome aspect of the business model of online advertising model that is perhaps the gravamen of the observation, made by Invincea's Pat Belcher, that, "any real-time ad bidding service that allows for automatic redirection is *inherently insecure* [emphasis added]." (*quoted by* Auchard & Saba, 2014). The domains that launch malvertising attacks tend to be located, for reasons easily understood, at the very ends of advertising's chains of HTTP redirection, and by the time that such a chain has been constructed, during the less-than-one-second interval following a user's request for delivery of a publisher's page, all of the preceding nodes responsible for the chain's creation, including the publisher and participating exchanges and networks (except perhaps for the entity who sold directly to the malvertiser, and who often would be no more likely than his downstream counterparty to aid anyone in detecting the imminent attack) have technologically washed their hands of the entire transaction. There is no longer any party who can stop it from culminating at the user's web browser.

This does not escape notice by the developers of MadTracer and SpiderWeb, who acknowledge that any future practical deployments of these tools will require their use in conjunction with a client web browser. Browsers in use today already employ some countermeasures, of course, to a greater or lesser extent. Firefox, Google Chrome and Apple Safari, for example, incorporate the use of Google's SafeBrowsing API, which continuously updates a blacklist of malicious web domains. Google Chrome has deprecated and will soon disable entirely the use of plugins built with the older NPAPI, allowing only the use of the newer and more secure PPAPI, while Microsoft's just-introduced Edge web browser drops support for *all* third-party plug-ins, even Microsoft's own ActiveX web components (although built-in support for Adobe Flash and pdf content will continue). Commercial software like Symantec Endpoint Protection can be purchased and installed to add additional protection, chiefly signature-based or blacklist-consulting, to any client web browser. Google's Chrome browser incorporates sandboxing techniques, finally, to help it defend against drive-by downloads, separating the browser's untrusted rendering engine from its trusted core kernel and running the former with limited privileges (Barth, Jackson, Reis, & Google Chrome Team, 2008; Sylvain, 2008); Apple's Safari, Microsoft's Edge, and even later versions of Internet Explorer incorporate some version of this countermeasure, too, and it is scheduled for deployment in Mozilla's Firefox browser soon. (Hoffman, 2015a; Hoffman, 2015b).

According to Lu, Yegneswaran, Porras, and Lee (2010), however, there are published client-side exploits for Chrome that demonstrate that its sandboxing approach is no panacea, a proposition demonstrated in the Pwn2Own web hacking competition sponsored by Hewlett-Packard at CanSecWest 2015, where the countermeasure was evaded by successful exploits targeting Internet Explorer 11 and Safari as well as Chrome. (Reynolds, 2015). The Lu-Yegneswaran-Porras-Lee team, working at Georgia Tech with the support of the U.S. Navy, completed development of their own prototype of a software project called BLADE, a kernel extension intended for use with the Windows OS that the development team reported to have realized a 100% success rate in thwarting covert web downloads not explicitly authorized by the user through his browser's graphical interface. BLADE works like a high-profile honeyclient in permitting such an incoming stream to be written to file, but a low-level input/output

redirector prevents the file from being executed by fencing it within a secure zone until it can be correlated with peripheral hardware input demonstrating user consent. Despite its heralded imminence a publicly available implementation of the BLADE prototype, unfortunately, has never appeared; the same is true of a browser-based alternative developed by Egele, Wurzinger, Kruegel, and Kirda (2009) that instead attempts to accomplish BLADE's objective by means of techniques that it shares with several existing Wepawet tools, using them to detect shellcode in JavaScript strings and proactively frustrate heap-spraying injection attacks.

If and when client-side countermeasures like those described in the last few paragraphs eventually do become publicly available, one can expect that some of them will come to be deployed by corporate IT security departments, perhaps being made to work closely and in tandem with the enterprise's existing web proxy servers. With products like these, however, the burden of protecting themselves from malicious exploits hidden in unsolicited online advertisements will also be placed once more upon the internet's end users, about 637 million or 45% of whom were already found, in a study conducted by Frei, Dübendorfer, Ollman, and May (2008), to be ignoring vendor warnings to discontinue their use of vulnerable, outdated web browsing software.

Mass success with client-side measures depends, in other words, upon the unlikely presence in the average consumer of a certain level of technical sophistication, while the criminal who syndicates a malvertisement retains the ability to target his attack to a grandmother who browses the web with Internet Explorer 6, on a desktop computer that is still running Windows XP. The larger issue presented, however, is that a consumer who has acquired the technical sophistication needed to protect himself with measures like those mentioned in the preceding paragraphs already enjoys access to (and indeed may well have already availed himself of) simple means that afford him a vastly superior level of protection— *i.e.*, installing ad-blocker software in his browser that will be nearly 100% effective. (Zarras *et al.*, 2014).

Mere mention of this possibility raises the specter of the one known in economics as the tragedy of the commons, a possibility of which the internet publishers' and advertisers' own self-interests should by now have made them keenly aware: according to Pauli (2015b), 198 million users have already chosen to operate ad-blocking software, thereby digging a $22 billion per year hole in the online advertising industry.[10] The protest heard when Apple

---

[10] In an article published in June 2015 *The Economist*, after citing comparable current estimates of ad-blocking software use, described legal and other countertactics that some web publishers and online advertising providers are beginning to deploy to strike back against the practice. (Erasmus, 2015).

It has since become evident that *The Economist* itself decided to join ranks with this camp by engaging the services of PageFair, an Irish startup that negotiates deals on behalf of publishers with firms like the developer of AdBlock Plus, and offers technical tools to circumvent ad-blocking software. As a result of this alliance *The Economist* was required in the first week of November 2015 to warn all readers who visited its site between 11:52 PM and 12:15 AM GMT on the preceding Halloween night and installed what appeared to be a downloadable update of the Adobe Flash Player onto a Windows OS computer to run a full system scan with a reputable anti-virus application, and remove whatever malware it may find. http://www.economist.com/help/pagefair.

The phishing payload in question, identified by Sophos as Mal/MSIL-LO, is a Trojan that includes a keylogger. The JavaScript exploit was delivered to readers by an ad-blocker-blocker supplied to *The Economist* by PageFair, who admitted that 501 different publishing websites— serving tens of millions of monthly visitors— were compromised by the breach of its service. (Vaas, 2015).

Computer recently announced, with its introduction of iOS 9, that it would henceforth support the use of ad-blocking apps in conjunction with the mobile version of its Safari web browser is evidence that at least sometimes advertising providers and publishers can be provoked to pay due heed to this existential danger. But in this regard consider also that although the newest HTML5 standard already gives webpage publishers the ability, for instance, to qualify iframes in which advertisements will appear with a "sandbox" attribute that would substantially limit the damage that can be inflicted from within their bounds, few publishers have yet judged it expedient to make any use of this handy and relatively complication-free restriction.[11] (Zarras *et al*., 2014).

d. Conclusion: There Ought to be a Law

In 2014 the Senate Permanent Committee on Investigations, having conducted hearings at the conclusion of which ranking minority member Senator John McCain severely upbraided representatives of Google and Yahoo! for the positions that they had expressed during the course of their testimony, issued a report in which it found industry efforts at self-regulation and policing to be wholly insufficient, and complained of a pervasive, across-the-board lack of accountability for the safety of the advertising content now being profitably delivered for mass consumption; while the majority report stopped short of recommending immediate legislation, its findings clearly constituted an official shot across the advertisers' bow. And even in the absence of new federal legislation, regulatory FTC action, or other changes in public policy, the principles of the law of tort existing at common law may give some enterprising personal-injury attorney sufficient legal foundation to mount a successful class action for a future malvertising attack that results in widespread consumer damage. (Guy, 2011-2012).

An attempt to impose legal liability, however, upon any party other than the attacker himself— who, if not unknown, is likely to be beyond the court's jurisdiction and judgment-proof in any event— would encounter daunting doctrinal obstacles under existing precedent and statutory law. A contractual model of the commercial relationship between a web publisher and his audience is exceedingly strained, and founding the liability of participants in the online advertising chain upon the tort doctrine of negligence has the weakness, among others, that the standard of ordinary care to which any one of them may be held accountable is likely to be established by consulting the practices now prevalent in the industry. The model of strict tort liability for the introduction of a dangerously defective product into the stream of commerce is more promising, but suffers from the fatal weakness that damages are not typically awarded in products-liability cases for injuries that are purely economic in nature. To overcome these and other difficulties this paper advocates that Congress exercise its interstate commerce power to enact a federal statute that will establish clear legal liability for the delivery of malvertising, as well as an ancillary regime of licensing and regulation to establish standards of responsibility and make actors accountable.

[11] To one who is inclined to believe that this may be because publishers don't appreciate the importance of keeping abreast of technological developments it should be pointed out that advertising exchanges and like entities often work closely with web publishers to create the on-page HTML used to facilitate display advertising.

Under this statute a website publisher, and any person or entity in the chain of advertising providers of particular content whose rights derive ultimately from that publisher, would be strictly liable to any member of the general public who is served malicious advertising from the publisher's webpage and as a consequence suffers damage. Recovery for purely economic damages would be authorized, including consequential damages up to a per-host/per-incident limit; optional presumptive liquidated damages in the case of difficult-to-quantify harm would be provided in a reasonable amount, as would be proportionate punitive damages, in cases of deliberate disregard of established regulatory requirements or egregious recklessness. Liability would be strict; that is, no proof of a party's negligence would be prerequisite to recovery from that party. Audience "contributory negligence" (*e.g.*, the use of an outdated browser, or the Java plug-in) would be irrelevant, and no recovery would barred by the doctrine of an "assumption of the risk" (*e.g.,* by visiting a pornographic website[12]). Class actions would be specifically authorized.

"Malicious advertising" would be defined to include any content that surreptitiously pushes any software whatsoever onto a client device, as well as specially-elaborated cases of "fraud by ad-based social-engineering" (carefully defined to exclude more general cases of economic loss caused by arguably false or misleading representations of the sort that existed well before the advent of web-based advertising), to wit: the purported sale, *via* e-commerce, of products that are in fact non-existent; the delivery of an executable that performs something other than, or anything material in addition to, its ostensible described function; and any phishing pretense or imposture, defined to comprise falsely posing online as another person or entity as a premise for or inducement of some additional transaction. "Publisher" would be carefully defined in such a way that one who merely refers another to a web page by supplying an HTML link would not thereby become a publisher of all advertising content that may be present on the target page, but careful subsidiary definitions would be needed for this distinction so as not to open a wide loophole when drawing it.

Any publisher or advertising provider who becomes liable to any other person for damages caused by a malicious advertisement placed upon a publisher's web pages would enjoy, in turn, a right to recovery from, and to indemnification for all of its losses by, its immediate and any subsequent transferee/assignee of the right that permitted the advertisement to be displayed; this right would be akin to one for breach of a covenant or the contractual terms of service and so also would require no proof of negligence or other fault to become effective. This provision should alleviate worries about a "chilling effect" that imposition of liability might have on the publishers of small web sites who rely upon advertising revenues to allow them to publish content free of charge, inasmuch as their chief burden would be limited to ensuring that they deal directly with reputable advertising exchanges whose financial resources are adequate to make this indemnification meaningful. Business liability insurance should also be available to them for the transfer of any residual risk in the ad publishing transaction.

---

[12] Actually Zarras *et al*. (2014) found that adult websites only ranked third in the delivery of malvertisements, with just 9.7% of the total. Categories ranking higher were entertainment (15.2%) and news (13.8%). "Safe browsing" habits offer the user no safe harbor.

The imposition of legal liability just proposed in the foregoing paragraphs is not primarily or even incidentally intended to punish,[13] but malicious advertising has an economic cost that inevitably is going to be borne by some unlucky *someone* somewhere. Under the *status quo* this cost is now being imposed upon the general public, who does not financially benefit from online advertising transactions, does not in any way solicit the content provided, and often lacks the technical sophistication to protect themselves from the dangers that it may present. Placing the burden instead on the participants in the advertising chain themselves will promote the efficient allocation of social resources by redistributing costs that are presently economic "externalities," will serve justice by imposing the costs of the activity upon those who presently profit from carrying it on, and will provide the incentive that is now lacking to devise innovative, efficient countermeasures, and to exercise ordinary due care, for those who are in the best technical, logistical and financial position to do so.

In the scheme of regulatory oversight that is proposed to be established by federal statute, the Federal Trade Commission would be tasked with licensing and enforcement, while the National Institute of Standards and Technology would assist by the development of detailed technical standards. Any "advertising provider", or member of a class that would be broadly and flexibly defined to include advertising exchanges, advertising networks, demand-side platforms, demand-management platforms, content distribution networks, and any business advertisers or their agencies whose own actions make it appropriate,[14] would be prohibited from engaging in, facilitating or soliciting any online advertising transaction unless it is a currently accredited provider. Accreditation would initially require satisfactory evidence of verifiable identity and good repute, as well as evidence of financial responsibility (bond-like insurance, or attested proof of adequate net worth to self-insure), and could subsequently be lost or suspended for failure to observe NIST/FTC standards and requirements.

The statute would prescribe, critically, that any advertising content that will ultimately be displayed at any one time in a single space of a page served on a publisher's web site (for a given single display advertising impression) must be entirely hosted by a *single* accredited content distribution network or other advertising content provider host. For this purpose an advertising host might contractually employ, for example, a cloud-service platform or infrastructure provider, so long as critical capabilities of the delivery infrastructure remain subject to its exclusive oversight and control to the extent necessary to enforce compliance

---

[13] It is assumed that the deliberate offenses of the malvertisers themselves, were the latter to be hunted down and apprehended, are already punishable under provisions of existing criminal statutes, including preeminently the Computer Fraud and Abuse Act of 1986, *as amended*, 18 U.S.C. § 1030 *et seq*. If this assumption is incorrect the necessary but lacking criminal sanctions could appropriately be provided by the statute here being described. But in essence this paper's policy prescription remains a civil and regulatory one.

[14] In a truly impressive feat of social engineering, an impostor purporting to act as an agent for the carmaker Suzuki purchased space for display advertisements in 2009 from the network Gawker Media, whose blogs *Gawker*, *Deadspin*, *Gizmodo, Jezebel* and *Lifehacker* thereupon treated their readers to malware-laden advertisements hosted on domains in Latvia; Gawker later posted its correspondence with the impostor— whose verisimilitude is astounding— at *Business Insider*. (Blodget, 2009). Under the statutory scheme of regulation herein proposed the "advertiser" in this episode could not have directly served self-hosted content to an online web publisher as it did unless it had first secured FTC provider accreditation, the continuing existence of which Gawker Media would also have been required to first verify itself.

with the specific regulatory requirements to which it is subject under the law. Any advertising provider that finally hosts and serves online advertising content that will be delivered to an audience browser would be required by regulation, in this regard:

> (1) to take reasonable steps to vet the business advertisers for whom it contracts to serve such content (*e.g.*, require like Apple does the creation of a permanent account, proof of established corporate identity, minimum financial responsibility, and so on);
>
> (2) to require that the content that it serves be self-contained, and prohibit any redirection of the client's browser by code within the advertising content to any off-network server location not under the host's exclusive effective control;
>
> (3) to prohibit the use of obfuscated JavaScript or similarly-purposed code in all advertisements that it serves, in order to facilitate transparent manual or automated static signature-based inspection;
>
> (4) to exercise due diligence (that is, to employ best-practices technical detection methods, using, for example, multiple-host-based, randomized-interval-intermittent anomaly- and/or behavior-based ad code scanning) to attempt to provide reasonable assurance that the advertising content that it will host is safe in actual fact;
>
> (5) to take care, using cryptographic hashing or equivalent procedures, to maintain the post-verification integrity of all advertising content that it hosts; and
>
> (6) to act quickly to investigate and effectively respond to any notice or report that particular advertising content that it has served is malicious.

The "self-contained" stricture in (2) above is not intended to prevent the presence in an online advertisement of clickable links directing a member of the public to the advertiser's business web pages. Such internal links do, however, present the dangers of phishing and other social-engineering fraud, and difficult issues for which this paper can at the moment offer no satisfactory practical solution.

The single-host provisions in their entirety would also not eliminate the present practices of advertising syndication and real-time auction. But the statute would require that a publisher, advertising exchange, advertising network or other entity that sells, resells, assigns or otherwise transfers a right to place such display content must first take reasonable steps to verify that its transferee, while perhaps not the ultimate advertising hosting provider, is nevertheless an accredited advertising provider. An accreditation database would for that purpose be maintained under the ultimate authority of the FTC, although the latter might well deem it preferable— because online verification of an entity's identity and current status would often need to be routinely accomplished during an interval measured in milliseconds— to delegate its day-to-day custody and maintenance instead to a certificate authority like Verisign.

In effect the contemplated statute would mandate the use of a public key infrastructure (PKI) to establish a chain of trust (and perhaps even more importantly, of non-repudiation and accountability) along the chain of HTTP redirection that constitutes the modern online display

advertising transaction. To effectuate this use of a PKI regulations might require the supplementation of the existing processes for the online transfer of digitized ad tags with the use of a special digital publisher token, which would legally evidence a time-limited right to display advertising content within a particular space on a particular web page that is served from the web publisher's domain. Upon any transfer, assignment or like negotiation of all or any part of such a right, the entity who does so would endorse the token by appending his cryptographic signature to its very end, after first adding to the token, immediately below the signature of its own predecessor if there is one or if there is not at the very top, textual descriptions of the right being conferred (*e.g.*, host URL, web-page pixel dimensions and location, allowable content type, number of impressions, valid-after and valid-before timestamps) and of the particulars of the immediate transfer being made (*e.g.*, the transferee's ad-tag URL and certificated corporate identity, with a transfer-date timestamp). By its digital signature the transferor would warrant to its transferee and any of the transferee's successors the authenticity of all preceding signatures, and its legal right to effect the transfer; more importantly the transferor would warrant to the ultimate recipient of the advertising content, as well as to the publisher and any other entities who precede it in the chain of HTTP indirection, that it has verified the identity of its transferee and the transferee's continuing good standing as an accredited provider. Before transmitting the token the transferor would encrypt it with the public key attached to the X.509 certificate whose named subject is its intended accredited transferee. Retention of a copy of the token after transfer would also be required.

Upon ultimate use of the publisher token by the advertising provider that hosts and serves the advertising content the procedure would be similar, except that the textual particulars of the transfer being made would include the timestamp of the HTTP response and the IP address of the audience recipient, while instead of a description of the rights conferred there would now be delivered the advertising content itself. The token and its contents would not be encrypted before delivery by the content provider to the end user, although the integrity of both would be protected by the provider's digital signature, which would here also function like the signature on downloaded application code, and attest especially to compliance with the requirements for responsible content hosting. Web browser software might be adapted so that before display and/or execution of the transmitted advertising content the browser would at a minimum authenticate the signatures of the content provider and publisher; check that the publisher URL in the token corresponds to the visited web page; and verify that the final advertising provider is currently accredited by the FTC. Tokens might also be locally cached for the length of some pre-determined interval, to enable their later forensic or legal use.

A mere detail of implementation that nevertheless here suggests itself is that the entire publisher token could be made to function as a digital envelope. The ever-adaptable XML, perhaps coupled with the use of an XSLT transformation at the final stage of the transaction into renderable HTML, seems to be ideally suited for such a use. In this case the necessary token semantics might be captured in a single XSD schema document; if this were done the first step in the defensive verification performed by the end user's web browser might be validation of the token's XML content against that same XSD schema document.

e. Related Research:

A group working in association with Google performed valuable early work in identifying the kind of web exploits that are the source of the problem described in this paper, and in conducting a statistical survey of the worldwide internet to assess the problem's magnitude. (Provos, McNamee, Mavrommatis, Wang, & Modadugu, 2007; Provos, Mavrommatis, Abu Rajab, & Monrose, 2008). There is another group apparently working at or in close association with the University of California at Santa Barbara whose members have performed much valuable research aimed at developing countermeasures for web-based exploits, and who authored an excellent general survey of the malvertising problem as well; to the very readable work of that group this paper is indebted, having already cited much of its work. (Cova, Kruegel, & Vigna, 2010; Egele, Wurzinger, Kruegel, & Kirda, 2009; Ford, Cova, Kruegel, & Vigna, 2009; Kapravelos, Cova, Kruegel, & Vigna, 2011; Stringhini, Kruegel, & Vigna, 2013; Zarras, Kapravelos, Stringhini, Holz, Kruegel, & Vigna, 2014). Similar to the work of the UCSB group in its value and pertinence (although not in available published extent) is the work of Li, Zhang, Xie, Yu, and Wang (2012).

Even in the conclusions of their surveys, however, whose results are as a general matter alarming, the foregoing authors have invariably stopped short of the kind of recommendations that are made in the conclusion of the present paper. Perhaps it is unsurprising that public policy recommendations might be thought inappropriate in academic papers of this kind,[15] but at the same time it is a shibboleth of contemporary doctrine that information security is more a matter of management processes and policies than it is one of technology. It is only because a web advertising transaction is so much more an inter-organizational rather than an intra-organizational one that the policies needed to control it must be supra-organizational, and therefore public in scope.

Public policy is of course the business of the subcommittee of the United State Senate whose 2014 report was cited in the conclusion of this paper. It is perhaps disappointing that despite the findings that the subcommittee adopted in its report no consensus could be found for measures stronger than a call for improved advertising industry self-policing. But in any event the report is itself a thought-provoking and insightful survey of the malvertising problem.

References:

Angelia, & Pishvar, D. (2013, January 27-30). Online advertising and its security and privacy concerns. *2013 15th International Conference on Advanced Communication Technology (ICACT).* 372-377.

[15] Zarras *et al*. (2014) did conclude with a few suggestions for better industry cooperation and self-policing, the most drastic of which resembles a private version of this paper's proposed provider accreditation scheme.

Less circumspect was the Invincea representative who publicly declared that the current prototypical advertising transaction is "fundamentally insecure." To Invincea is owed the insight that the malvertising problem would be significantly reduced or even disappear if it could somehow be required that an advertisement be wholly hosted and served from a single ad server.

Auchard, E., & Saba, J. (2014, October 16). 'Malvertising' targets U.S. military firms in new twist on old web threat. *Reuters (U.S. edition)*. Retrieved September 12, 2015, from http://www.reuters.com/article/2014/10/16/us-cybersecurity-military-idUSKCN0I529H20141016.

Barth, A., Jackson, C., Reis, C., & Google Chrome Team. (2008). The security architecture of the Chromium browser. *Stanford Technical Report*. 1-10.

Bilogorskiy, N. (2015, January 5). Huffington Post serving malware via AOL ad-network. *Cyphort Blog.* Retrieved September 1, 2015, from http://www.cyphort.com/huffingtonpost-serving-malware/.

Bilogorskiy, N. (2015, August 15). Huffington Post serves malvertising, again. *Cyphort Blog.* Retrieved September 1, 2015, from http://www.cyphort.com/100m-huffington/.

Blodget, H. (2009, October 26). Gawker scammed by malware crew pretending to be Suzuki. *Business Insider.* Retrieved October 10, 2015, from http://www.businessinsider.com/henry-blodget-gawker-scammed-by-malware-pretending-to-be-suzuki-2009-10

Cova, M., Kruegel, C., & Vigna, G. (2010, April). Detection and analysis of drive-by attacks and malicious JavaScript code. *WWW '10: Proceedings of the 19th International Conference on World Wide Web.* 281-290.

Dai, S. -Y., Fyodor, Y., Wu, M. -W., Huang, Y., & Kuo, S. -Y. (2012, September). Holography: a behavior-based profiler for malware analysis. *Software: Practice and Experience 42*(9). 1107-1136.

Egele, M., Wurzinger, P., Kruegel, C., & Kirda, E. (2009). Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment* (88-106). Springer Berlin Heidelberg.

Erasmus. (2015, June 6). Online advertising: Block shock. *The Economist*. Retrieved November 7, 2015, from http://www.economist.com/news/business/21653644-internet-users-are-increasingly-blocking-ads-including-their-mobiles-block-shock.

Ford, S., Cova, M., Kruegel, C., & Vigna, G. (2009). Analyzing and detecting malicious Flash advertisements. *2009 Annual Computer Security Applications Conference.* 363-372.

Frei, S., Dübendorfer, T., Ollman, G., & May, M. (2008). Understanding the web browser threat: Examination of vulnerable online web browser populations and the "insecurity iceberg". *Proceedings of DefCon 16.*

Gallagher, S. (2012, January 25). "Blackhole" toolkit dominates Web malware attacks, says Sophos. *Ars Technica.* Retrieved October 10, 2015, from http://arstechnica.com/business/2012/01/blackhole-dominates-web-malware-attacks-says-sophos/

Gallagher, S. (2015, April 17). Faked Flash-based ads on HuffPo, other sites downloaded extortionware. *Ars Technica.* Retrieved September 1, 2015, from http://arstechnica.com/security/2015/04/faked-flash-based-ads-on-huffpo-other-sites-downloaded-extortionware/.

Goldfarb, A. (2014, March). What is different about online advertising? *Review of Industrial Organization 44*(2). 115-129.

Goodin, D. (2015, August 14). My browser visited Weather.com and all I got was this lousy malware (Updated). *Ars Technica.* Retrieved September 1, 2015, from http://arstechnica.com/security/2015/08/my-browser-visited-drudgereport-and-all-i-got-was-this-lousy-malware/.

Guy, C. (2011-2012). Malvertising: Tort law's role in ridding the web of malicious advertising. *South Texas Law Review 53.* 421-457.

Hern, A. (2015, August 5). Yahoo users hit by 'malvertising' campaign. *The Guardian.* Retrieved September 1, 2015, from http://www.theguardian.com/technology/2015/aug/05/yahoo-users-malvertising-campaign-malware.

Hoffman, C. (2015, September 1). What is malvertising and how do you protect yourself? *How-To Geek.* Retrieved October 7, 2015, from http://www.howtogeek.com/227205/what-is-malvertising-and-how-do-you-protect-yourself/.

Hoffman, C. (2015, September 11). Firefox is about to become an almost complete copy of Chrome. *How-To Geek.* Retrieved October 7, 2015, from http://www.howtogeek.com/228131/firefox-is-about-to-become-an-almost-complete-copy-of-chrome/.

Hoglund, G., & McGraw, G. (2004). *Exploiting software: How to break code.* Saddle River, NJ: Pearson Education, Inc.

Hong, J. (2010, December). Malvertisements growing as online security threat. *Communications of the ACM 53*(12). 10-11.

Invincea, Inc. (2014, October 27). Micro-targeted malvertising via real-time ad-bidding. *Invincea White Paper*. Retrieved September 12, 2015, from http://www.invincea.com/wp-content/downloads/Micro-Targeted-Malvertising-WP-10-27-14-1.pdf.

Kapravelos, A., Cova, M., Kruegel, C., & Vigna, G. (2011). Escape from Monkey Island: Evading high-interaction honeyclients. *Proceedings of the 8th International Conference on the Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2011)*. 124-143.

Kashyap, R. (2014, November). Why malvertising is cybercriminals' latest sweet spot. *Wired.com*. Retrieved September 1, 2015, from http://www.wired.com/insights/2014/11/malvertising-is-cybercriminals-latest-sweet-spot/.

Le, V. L., Welch, I., Gao, X., Komisarckzuk, P. (2013, October 21-14). Detecting heap-spray attacks in drive-by downloads: Giving attackers a hand. *38th Annual IEEE Conference on Local Computer Networks*. 300-303.

Li, Z., Zhang, K., Xie, Y., Yu, F., & Wang, X.F. (2012). Knowing your enemy: Understanding and detecting malicious web advertising. *Proceedings of the ACM Conference on Computer and Communications Security*. 674-686.

Lu, L., Yegneswaran, V., Porras, P., & Lee, W. (2010, October). BLADE: An attack-agnostic approach for preventing drive-by malware infections. *CCS '10: Proceedings of the 17th ACM Conference on Computer and Communications Security.* 440-450.

Mansfield-Devine, S. (2014, November). The dark side of advertising. *Computer Fraud & Security, 2014*(11), 5-8.

Moonsamy, V., & Batten, L. (2014, October 27-29). Android applications: Data leaks via advertising libraries. *2014 International Symposium on Information Theory and its Applications.* 314-317.

Narvaez, J., Endicott-Popovsky, B., Seifert, C., Aval, C., & Frincke, D.A. (2010). Drive-by downloads. *Proceedings of the 43rd Hawaii International Conference on System Sciences.* 1-10.

OpenX. (2013, October 13). Ad networks vs. ad exchanges: How they stack up. *OpenX White Paper.* Retrieved September 24, 2015, from http://openx.com/whitepapers/.

OpenX. (2014, May 27). Data equals dollars. *OpenX White Paper.* Retrieved September 24, 2015, from http://openx.com/whitepapers/.

Pauli, D. (2015, August 13). Malvertising is set to wreak one BEELLION dollars in damage this year. *The Register.* Retrieved September 1, 2015, from http://www.theregister.co.uk/2015/08/13/june_worst_malvertising_month/.

Pauli, D. (2015, August 27). Malvertising menaces poison ads as Google, Yahoo! look away. *The Register.* Retrieved September 1, 2015, from http://www.theregister.co.uk/2015/08/27/malvertising_feature/.

Pricewaterhousecoopers, L.L.C. (Internet Advertising Bureau, sponsor). (2015, April 22). IAB internet advertising revenue report, 2014 full results. Retrieved September 12, 2015, from http://www.iab.net/media/file/IAB_Internet_Advertising_Revenue_FY_2014.pdf.

Provos, N., Mavrommatis, P., Abu Rajab, M., & Monrose, F. (2008). All your iFRAMES point to us. *Proceedings of the 17th Conference on Security Symposium, USENIX Association, Berkeley, CA, USA.* 1-15.

Provos, N., McNamee, D., Mavrommatis, P., Wang, K., & Modadugu, N. (2007, April). The ghost in the browser: Analysis of web-based malware. *Proceedings of the First Conference on Hot Topics in Understanding Botnets (HotBots '07), USENIX Association, Berkeley, CA, USA.* 10 pages.

Qassrawi, M.T. (2011) Detecting malicious web servers with honeyclients. *Journal of Networks 6*(1). 145-152.

Rashid, F.Y. (2015, February 11). Chinese attackers hacked Forbes website in watering hole attack: security firms. *Security Week*. Retrieved September 21, 2015, from http://www.securityweek.com/chinese-attackers-hacked-forbes-website-watering-hole-attack-security-firms.

Reynolds, S. (2015, March 23). No browser was safe at Pwn2Own 2015. *VR World: Technology Security.* Retrieved October 7, 2015, from http://www.vrworld.com/2015/03/23/no-browser-was-safe-at-pwn2own-2015/.

Santillan, M. (2015, January 7). Malvertising campaign hits the Huffington Post through AOL ad network. *Tripwire.* Retrieved September 1, 2015, from http://www.tripwire.com/state-of-security/latest-security-news/malvertising-campaign-hits-the-huffington-post-through-aol-ad-network/.

Segura, J. (2015, August 3). Large malvertising campaign takes on Yahoo! *Malwarebytes Unpacked.* Retrieved September 1, 2015, from https://blog.malwarebytes.org/malvertising-2/2015/08/large-malvertising-campaign-takes-on-yahoo/.

Sinegubko, D. (2015, January 14). AdSense abused with malvertising campaign. *Sucuri Blog.* Retrieved October 10, 2015, from https://blog.sucuri.net/2015/01/adsense-abused-with-malvertising-campaign.html.

Sood, A.K., & Enbody, R.J. (2011, April). Malvertising: exploiting web advertising. *Computer Fraud & Security, 2011*(4), 11-16.

Stringhini, G., Kruegel, C., & Vigna, G. (2013, November). Shady paths: Leveraging surfing crowds to detect malicious web pages. *Proceedings of the 2013 ACM SIGSAC (Special Interest Group of Security, Auditing and Control) Conference on Computer & Communications Security, CCS '13.* 133-144.

Sylvain, N. (2008, October 2). A new approach to browser security: the Google Chrome sandbox. *The Chromium Blog.* Retrieved October 7, 2015, from http://blog.chromium.org/2008/10/new-approach-to-browser-security-google.html.

Tubin, G. (2013, January 20). Malvertising gets a boost from unpatched Java zero-day exploits. *Security Intelligence.* Retrieved October 10, 2015, from https://securityintelligence.com/malvertising-campaigns-get-boost-unpatched-java-zero-day-exploits/.

United States Senate, Committee on Homeland Security and Governmental Affairs, Permanent Subcommittee on Investigations. (2014, May 15). Online advertising and hidden hazards to consumer privacy and data security (staff report).

Vaas, L. (2015, November 4). PageFair analytics hacked and used to distribute malware on Halloween. *nakedsecurity (Sophos)*. Retrieved November 7, 2015, from

https://nakedsecurity.sophos.com/2015/11/04/pagefair-analytics-hacked-and-used-to-distribute-malware-on-halloween/.

Van Overveldt, T., Kruegel, C., & Vigna, G. (2012). FlashDetect: ActionScript 3 malware detection. *Research in Attacks, Intrusions, and Defenses. Proceedings of the 15th International Symposium, RAID 2012*. 274-293.

Vance, A. (2009, September 14). Times web ads show security breach. *New York Times*. Retrieved September 1, 2015, from http://www.nytimes.com/2009/09/15/technology/internet/15adco.html?_r=0

Zarras, A, Kapravelos, A., Stringhini, H., Holz, T., Kruegel, C., & Vigna, G. (2014, November 5). The dark alleys of Madison Avenue: Understanding malicious advertisements. *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*. 373-379.