Google Technical Report provos-2008a

# All Your iFRAMEs Point to Us

*Niels Provos* *Panayiotis Mavrommatis* *Moheeb Abu Rajab* *Fabian Monrose*

**Abstract**

As the web continues to play an ever increasing role in information exchange, so too is it becoming the prevailing platform for infecting vulnerable hosts. In this paper, we provide a detailed study of the pervasiveness of so-called *drive-by downloads* on the Internet. Drive-by downloads are caused by URLs that attempt to exploit their visitors and cause malware to be installed and run automatically. Our analysis of billions of URLs over a 10 month period shows that a non-trivial amount, of over 3 million malicious URLs, initiate drive-by downloads. An even more troubling finding is that approximately 1.3% of the incoming search queries to Google's search engine returned at least one URL labeled as malicious in the results page. We also explore several aspects of the drive-by downloads problem. We study the relationship between the user browsing habits and exposure to malware, the different techniques used to lure the user into the malware distribution networks, and the different properties of these networks.

February 4th, 2008

Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043

DRAFT

DRAFT

# All Your `iFRAME`s Point to Us

Niels Provos* Panayiotis Mavrommatis* Moheeb Abu Rajab† Fabian Monrose†

## 1 Introduction

It should come as no surprise that our increasing reliance on the Internet for many facets of our daily lives (*e.g.,* commerce, communication, entertainment, etc.) makes the Internet an attractive target for a host of illicit activities. Indeed, over the past several years, Internet services have witnessed major disruptions from attacks, and the network itself is continually plagued with malfeasance [12]. While the monetary gains from the myriad of illicit behaviors being perpetrated today (*e.g.,* phishing, spam) is just barely being understood [9], it is clear that there is a general shift in tactics—wide-scale attacks aimed at overwhelming computing resources are becoming less prevalent, and instead, traditional scanning attacks are being replaced by other mechanisms. Chief among these is the exploitation of the web, and the services built upon it, to distribute malware.

This change in the playing field is particularly alarming, as unlike traditional scanning attacks that use push-based infection to increase their population, web-based malware infection follows a pull-based model. For the most part, the techniques in use today for delivering web-malware can be divided into two main categories. In the first case, attackers use various social engineering techniques to entice the visitors of a website to download and run malware. The second, more devious case, involves the underhanded tactic of targeting various browser vulnerabilities to *automatically* download and run—i.e., unknowingly to the visitor—the binary upon visiting a website. When popular websites are exploited, the potential victim base from these so-called *drive-by downloads* can be far greater than other forms of exploitation because traditional defenses (e.g., firewalls, dynamic addressing, proxies) pose no barrier to infection. While social engineering may, in general, be an important malware spreading vector, in this work we restrict our focus and analysis to malware delivered via drive-by downloads.

Recently, Provos *et al.* [17] provided insights on this new phenomenon, and presented a cursory overview of web-based malware. Specifically, they described a number of server- and client-side exploitation techniques that are used to spread malware, and elucidated the mechanisms by which a successful exploitation chain can start and continue to the automatic installation of malware. In this paper, we present a detailed analysis of the malware serving infrastructure on the web using a large corpus of malicious URLs collected over a period of ten months. Using this data, we estimate the global prevalence of drive-by downloads, and identify several trends for different aspects of the web malware problem. Our results reveal an alarming contribution of Chinese-based web sites to the web malware problem– overall, 67% of the malware distribution servers and 64% of the web sites that link to them are located in China. These results raise serious question about the security practices employed by web site administrators.

*Google Inc., e-mail:{niels, panayiotis}@google.com

†Computer Science Department, Johns Hopkins University, e-mail:{moheeb, fabian}@cs.jhu.edu

Additionally, we study several properties of the malware serving infrastructure, and show that (for the most part) the malware serving networks are composed of tree-like structures with strong fan-in edges leading to the main malware distribution sites. These distribution sites normally deliver the malware to the victim after a number of indirection steps traversing a path on the distribution network tree. More interestingly, we show that several malware distribution networks have linkages that can be attributed to various relationships.

In general, the edges of these malware distribution networks represent the hop-points used to lure users to the malware distribution site. By investigating these edges, we reveal a number of causal relationships that eventually lead to browser exploitation. More troubling, we show that drive-by downloads are being induced by mechanisms beyond the conventional techniques of controlling the content of compromised websites. In particular, our results reveal that ad serving networks are increasingly being used as hops in the malware serving chain. We attribute this increase to syndication, a common practice which allows advertisers to rent out part of their advertising space to other parties. These findings are problematic as they show that even protected web-servers can be used as vehicles for transferring malware. Additionally, we also show that contrary to common wisdom, the practice of following "safe browsing" habits (*i.e.,* avoiding gray content) by itself is not an effective safeguard against exploitation.

The remainder of this paper is organized as follows. In Section 2, we provide background information on how vulnerable computer systems can be compromised solely by visiting a malicious web page. Section 3 gives an overview of our data collection infrastructure and in Section 4 we discuss the prevalence of malicious web sites on the Internet. In Section 5, we explore the mechanisms used to inject malicious content into web pages. We analyze several aspects of the web malware distribution networks in Section 6. In Section 7 we provide an overview of the impact of the installed malware on the infected system. Section 8 presents related work. Finally, we conclude in Section 9.

## 2 Background

Unfortunately, there are a number of existing exploitation strategies for installing malware on a user's computer. One common technique for doing so is by remotely exploiting vulnerable network services. However, lately, this attack strategy has become less successful (and presumably, less profitable). Arguably, the proliferation of technologies such as Network Address Translators (NATs) and Firewalls make it difficult to remotely connect and exploit services running on users' computers. This, in turn, has lead attackers to seek other avenues of exploitation. An equally potent alternative is to simply lure web users to connect to (compromised) malicious servers that subsequently deliver exploits targeting vulnerabilities of web browsers or their plugins.

Adversaries use a number of techniques to inject content under their control into benign websites. In many cases, adversaries exploit web servers via vulnerable scripting applications. Typically, these vulnerabilities (*e.g.,* in phpBB2 or InvisionBoard) allow an adversary to gain direct access to the underlying operating system. That access can often be escalated to super-user privileges which in turn can be used to compromise any web server running on the compromised host. In general, upon successful exploitation of a web server the adversary injects new content to the compromised website. In most cases, the injected content is a link that redirects the visitors of these websites to a `URL` that hosts a script crafted to exploit the browser. To avoid visual detection by website owners, adversaries normally use invisible HTML components (*e.g.,* zero pixel `IFRAME`s) to hide the

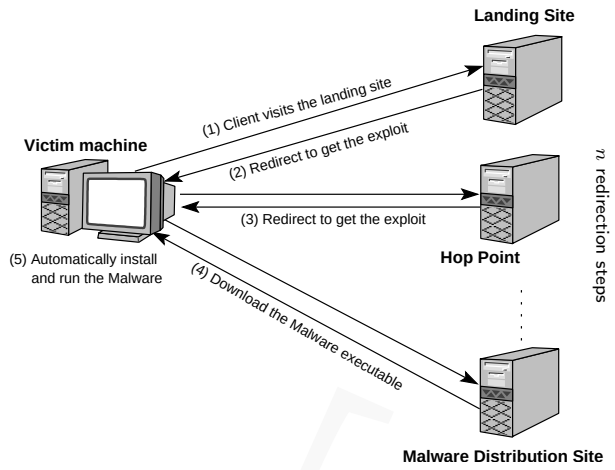



Figure 1: A typical Intercation with of drive-by download victim with a landing `URL` .

injected content.

Another common content injection technique is to use websites that allow users to contribute their own content, for example, via postings to forums or blogs. Depending on the site's configuration, user contributed content may be restricted to text but often can also contain HTML such as links to images or other external content. This is particularly dangerous, as without proper filtering in place, the adversary can simply inject the exploit `URL` without the need to compromise the web server.

Figure 1 illustrates the main phases in a typical interaction that takes place when a user visits a website with injected malicious content. Upon visiting this website, the browser downloads the initial exploit script (*e.g.,* via an `IFRAME`). The exploit script (in most cases, `javascript`) targets a vulnerability in the browser or one of its plugins. Interested readers are referred to Provos *et al.* [17] for a number of vulnerabilities that are commonly used to gain control of the infected system. Successful exploitation of one of these vulnerabilities results in the automatic execution of the exploit code, thereby triggering a drive-by download. Drive-by downloads start when the exploit instructs the browser to connect to a malware distribution site to retrieve malware executable(s). The downloaded executable is then automatically installed and started on the infected system.

Finally, attackers use a number of techniques to evade detection and complicate forensic analysis. For example, the use of randomly seeded obfuscated `javascript` in their exploit code is not uncommon. Moreover, to complicate network based detection attackers use a number or redirection steps before the browser eventually contacts the malware distribution site.

# 3 Infrastructure and Methodology

Our primary objective is to identify malicious web sites (*i.e.*, `URLs` that trigger drive-by downloads) and help improve the safety of the Internet. Before proceeding further with the details of our data collection methodology, we first define some terms we use throughout this paper. We use the terms *landing pages* and *malicious* `URLs` interchangeably to denote the `URLs` that initiate drive-by downloads when users visit them. In our subsequent analysis, we group these `URLs` according to their top level domain names and we refer to the resulting set as the *landing sites*. In many cases,

the malicious payload is not hosted on the landing site, but instead loaded via an IFRAME or a SCRIPT from a remote site. We call the remote site that hosts malicious payloads a *distribution site*. In what follows, we detail the different components of our data collection infrastructure.

**Pre-processing Phase.** As Figure 2 illustrates, the data processing starts from a large web repository maintained by Google. Our goal is to inspect URLs from this repository and identify the ones that trigger drive-by downloads. However, exhaustive inspection of each URL in the repository is prohibitively expensive due to the large number of URLs in the repository (on the order of billions). Therefore, we first use light-weight techniques to extract URLs that are likely malicious then subject them to a more detailed analysis and verification phase.

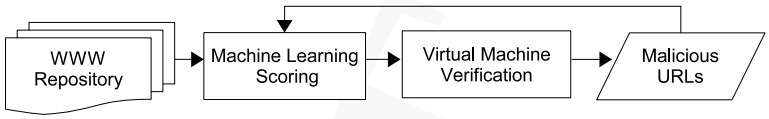



Figure 2: URL selection and verification workflow.

We employ the *mapreduce* [7] framework to process billions of web pages in parallel. For each web page, we extract several features, some of which take advantage of the fact that many landing URLs are hijacked to include malicious payload(s) or to point to malicious payload(s) from a distribution site. For example, we use "out of place" IFRAMEs, obfuscated JavaScript, or IFRAMEs to known distribution sites as features. Using a specialized machine-learning framework [5], we translate these features into a likelihood score. We employ five-fold cross-validation to measure the quality of the machine-learning framework. The cross-validation operates by splitting the data set into 5 randomly chosen partitions and then training on four partitions while using the remaining partition for validation. This process is repeated five times. For each trained model, we create an ROC [1] curve and use the average ROC curve to estimate the overall accuracy. Using this ROC curve, we estimate the false positive and detection rate for different thresholds. Our infrastructure pre-processes roughly *one billion* pages daily. In order to fully utilize the capacity of the subsequent detailed verification phase, we choose a threshold score that results in an outcome false positive rate of about $10^{-3}$ with a corresponding detection rate of approximately 0.6. This amounts to about one million URLs that we subject to the computationally more expensive verification phase.

In addition to analyzing web pages in the crawled web repository, we also regularly select several hundred thousands URLs for in-depth verification. These URLs are randomly sampled from popular URLs as well as from the global index. We also process URLs reported by users.

**Verification Phase.** This phase aims to verify whether a candidate URL from the pre-processing phase is malicious (*i.e.,* initiates a drive-by download). To do that, we developed a large scale *web-honeynet* that simultaneously runs a large number of Microsoft Windows images in virtual machines. Our system design draws on the experience from earlier work [22], and includes unique features that are specific to our goals. In what follows we discuss the details of the URL verification process.

Each honeypot instance runs an unpatched version of Internet Explorer. To inspect a candidate URL , the system first loads a clean Windows image then automatically starts the browser and

[1] ROC stands for receiver operating characteristic which plots the false-positive rate against the detection rate.

instructs it to visit the candidate URL . We detect malicious URLs using a combination of execution based heuristics and results from anti-virus engines. Specifically, for each visited URL we run the virtual machine for approximately two minutes and monitor the system behavior for abnormal state changes including file system changes, newly created processes and changes to the system's registry. Additionally, we subject the HTTP responses to virus scans using multiple anti-virus engines. To detect malicious URLs , we develop scoring heuristics used to determines the likelihood that a URL is malicious. We determine a URL score based on a combined measure of the different state changes resulting from visiting the URL . Our heuristics score URLs based on the number of created processes, the number of observed registry changes and the number of file system changes resulting from visiting the URL .

To limit false positives, we choose a conservative decision criteria that uses an empirically derived threshold to mark a URL as malicious. This threshold is set such that it will be met if we detect changes in the system state, including the file system as well as creation of new processes. A visited URL is marked as *malicious* if it meets the threshold *and* one of the incoming HTTP responses is marked as malicious by at least one anti-virus scanner. Our extensive evaluation shows that this criteria introduces negligible false positives. Finally, a URL that meets the threshold requirement but has no incoming payload flagged by any of the anti-virus engines, is marked as *suspicious*.

On average, the detailed verification stage processes about one million URLs daily, of which roughly $25,000$ new URLs are flagged as malicious. The verification system records all the network interactions as well as the state changes. In what follows, we describe how we process the network traces associated with the detected malicious URLs to shed light on the malware distribution infrastructure.

**Constructing the Malware Distribution Networks.** To understand the properties of the web malware serving infrastructure on the Internet, we analyze the recorded network traces associated with the detected malicious URLs to construct the *malware distribution networks*. We define a distribution network as the set of malware delivery trees from all the landing sites that lead to a particular malware distribution site. A malware delivery tree consists of the landing site, as the leaf node, and all nodes (*i.e.,* web sites) that the browser visits until it contacts the malware distribution site (the root of the tree). To construct the delivery trees we extract the edges connecting these nodes by inspecting the Referer header from the recorded successive HTTP requests the browser makes after visiting the landing page. However, in many cases the Referer headers are not sufficient to extract the full chain. For example, when the browser redirection results from an external script the Referrer, in this case, points to the base page and not the external script file. Additionally, in many cases the Referer header is not set (*e.g.,* because the requests are made from within a browser plugin or newly-downloaded malware).

To connect the missing causality links, we interpret the HTML and JavaScript content of the pages fetched by the browser and extract all the URLs from the fetched pages. Then, to identify causal edges we look for any URLs that match any of the HTTP fetches that were subsequently visited by the browser. In some cases, URLs contain randomly generated strings, so some requests cannot be matched exactly. In these cases, we apply heuristics based on edit distance to identify the most probable parent of the URL . Finally, for each malware distribution site, we construct its associated distribution network by combining the different malware delivery trees from all landing pages that lead to that site.

Our infrastructure has been live for more than one year, continuously monitoring the web and detecting malicious URLs . In what follows, we report our findings based on analyzing data collected during that time period. Again, recall that we focus here on the pervasiveness of malicious activity (perpetrated by drive-by downloads) that is induced simply by visiting a landing page, thereafter requiring *no* additional interaction on the client's part (*e.g.,* clicking on embedded links). Finally, we note that due to the large scale of our data collection and some infrastructural constraints, a number longitudinal aspects of the web malware problem (*e.g.,* the lifetime of the different malware distribution networks) are beyond the scope of this paper and are a subject of our future investigation.

# 4 On the Prevalence of Drive-by Downloads

We provide an estimate of the prevalence of web-malware based on data collected over a period of ten months (Jan 2007 - Oct 2007). During that period, we subjected over 60 million URLs for in-depth processing through our verification system. Overall, we detected more than 3 million malicious URLs hosted on more than 180 thousand landing sites. Overall, we observed more than 9 thousand different distribution sites. The findings are summarized in Table 1. Overall, these results show the scope of the problem, but do not necessarily reflect the exposure of end-users to drive-by downloads. In what follows, we attempt to address this question by estimating the overall impact of the malicious web sites.

| | |
|---|---|
| Data collection period | Jan - Oct 2007 |
| Total URLs checked in-depth | 66,534,330 |
| Total suspicious landing URLs | 3,385,889 |
| Total malicious landing URLs | 3,417,590 |
| Total malicious landing sites | 181,699 |
| Total distribution sites | 9,340 |

Table 1: Summary of Collected Data.

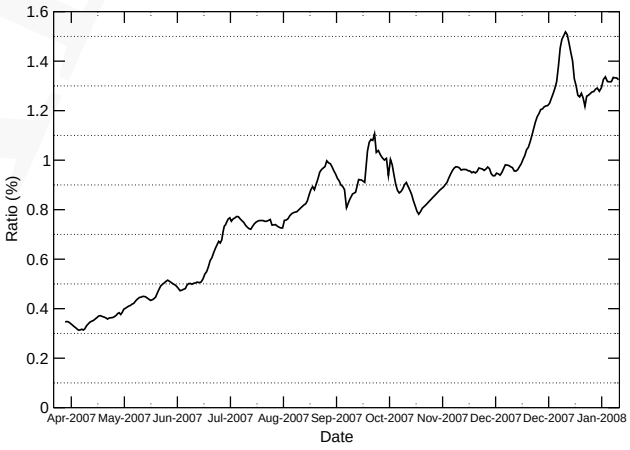



Figure 3: Fraction of search queries that resulted in at least one malicious URL . (7-day running avg.)

## 4.1 Impact on the end-user

To study the potential impact of malicious web sites on the end-users, we first examine the fraction of incoming search queries to Google's search engine that return at least one URL labeled as malicious in the results page. Figure 3 provides a running average of this fraction. The graph shows an increasing trend in the search queries that return at least one malicious result, with an average approaching 1.3% of the overall incoming search queries. This finding is troubling as it shows that

a significant fraction of search queries return results that may expose the end-user to exploitation attempts.

To further understand the importance of this finding, we inspect the prevalence of malicious sites among the links that appear most often in Google search results. From the top one million URLs appearing in the search engine results, about 6,000 belong to sites that have been verified as malicious at some point during our data collection. Upon closer inspection, we found that these sites appear at uniformly distributed ranks within the top million web sites—with the most popular landing page having a rank of 1,588. These results further highlight the significance of the web malware threat as they show the extent of the malware problem; in essence, about 0.6% of the top million URLs that appeared most frequently in Google's search results led to exposure to malicious activity at some point.

### 4.2 Geographic locality

One noteworthy result is the geographic locality of web based malware. Tables 2 and 3 show the geographic breakdown of IP addresses of the malware distribution sites and the landing sites, respectively. The results show that a significant number of Chinese-based sites contribute to the drive-by problem. Overall, 67% of the malware distribution sites and 64.6% of the landing sites are hosted in China. These findings provide more evidence [11] of poor security practices by web site administrators (*e.g.,* running out-dated and unpatched versions of the web server software).

Finally, we examined the geographic locality of the web-malware distribution networks as a whole (*i.e.,* the correlation between the location of a distribution site and the landing sites pointing to it). Interestingly, the malware distribution networks are highly localized within common geographical boundaries. Upon closer investigation, we see that this locality varies across different countries, and is most evident in China, with 96% of the landing sites pointing to malware distribution servers hosted in China.

| Malware Dist. site hosting country | % of all distribution sites |
|---|---|
| China | 67.0% |
| United States | 15.0% |
| Russia | 4.0% |
| Malaysia | 2.2% |
| Korea | 2.0% |
| Panama | 1.1% |
| Germany | 1.0% |
| Hong Kong | 0.8% |
| Turkey | 0.7% |
| France | 0.7% |
| Other | 5.7% |

Table 2: Hosting countries for the distribution sites.

| Landing site hosting country | % of all landing sites |
|---|---|
| China | 64.4% |
| United States | 15.6% |
| Russia | 5.6% |
| Korea | 2.0% |
| Germany | 2.0% |
| Czech Republic | 0.9% |
| Ukraine | 0.8% |
| Taiwan | 0.8% |
| Poland | 0.7% |
| Canada | 0.6% |
| Other | 6.5% |

Table 3: Hosting countries for the landing sites.

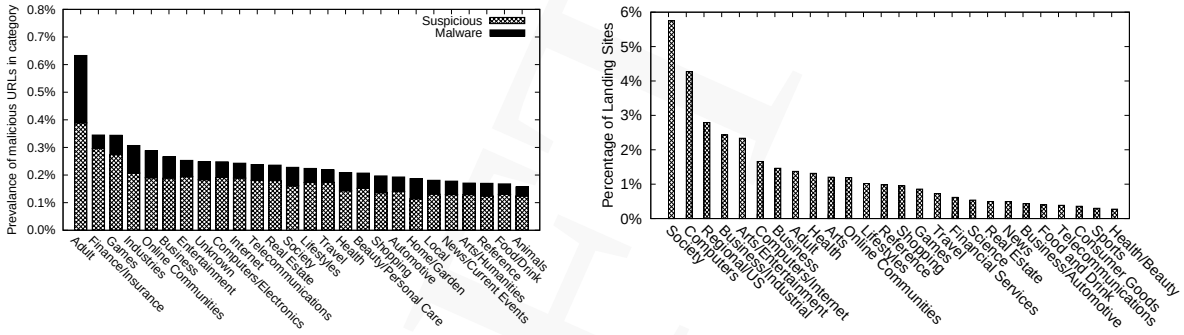
### 4.3 Impact of browsing habits

To examine the potential impact of users' browsing habits on their exposure to web exploitation, where possible, we map the set of detected landing URLs to the DMOZ functional categories [1][2]. We

[2]This mapping is readily available at Google.

DRAFT

examined two different data sets: the first is a sample of 7.2 million randomly selected URLs and the second is all 3.3 million malicious URLs found during the course of this study. Of the two URL sets, we were able to categorize about 50% and 30%, respectively. Figure 4 shows the top DMOZ categories for the inspected URLs. While the DMOZ categorization is far from perfect, the results in Figure 4.a show that browsing gray content, such as adult web pages, may increase the risk of exploitation. Among the set of landing pages detected in the random set of URLs , we found that approximately 0.24% were adult web pages whereas the other categories were as low as 0.018%. That said, as Figure 4.b shows, the total number of drive-by downloads on such pages is much lower compared to the other categories, demonstrating that ordinary web pages are also contributors to web exploitation. Therefore, even if someone was to avoid adult web pages, s/he would still be exposed to risk. As we show later, this can be explained by the mechanisms being used to distribute web malware.




(a) Random URL sample.

(b) All Malicious URLs .

Figure 4: Distribution of malicious URLs in DMOZ categories.

## 5 Malicious Content Injection

In Section 4, we showed that exposure to web-malware is not strongly tied to a particular browsing habit. Our assertion is that this is due, in part, to the fact that drive-by downloads are triggered by visiting staging sites that are not necessarily of malicious intent but have content that lures the visitor into the malware distribution network.

In this section, we validate this conjecture by studying the properties of the web sites that participate in the malware delivery trees. As discussed in Section 2, attackers use a number of techniques to control the content of benign web sites and turn them into nodes in the malware distribution networks. These techniques can be divided into two categories: web server compromise and third party contributed content (*e.g.,* blog posts). Unfortunately, it is generally difficult to determine the exact contribution of either category. In fact, in some cases even manual inspection of the content of each web site may not lead to conclusive evidence regarding the manner in which the malicious content was injected into the web site. Instead, in this section we provide insights into some features of these web sites that may explain their presence in the malware delivery trees. We only focus on the features that we can determine in an automated fashion. Specifically, where possible, we first inspect the version of the software running on the web server for each landing site. Additionally, we explore one important angle that we discovered which contributes significantly to

the distribution of web malware—namely, drive-by downloads via Ads.

### 5.1 Web Server Software

We first begin by examining (where possible) the software running on the web-servers for all the landing sites that lead to the malware distribution sites. Specifically, we collected all the "`Server`" and "`X-Powered-By`" header tokens from each landing page (see Table 4). Not surprisingly, of those servers that reported this information, a significant fraction were running outdated versions of software with well known vulnerabilities [3]. For example, 38.1% of the Apache servers and 39.9% of servers with PHP scripting support reported a version with security vulnerabilities. Overall, these results reflect the weak security practices applied by the web site administrators. Clearly, running unpatched software with known vulnerabilities increases the risk of content control via server exploitation.

| Server Software | Servers | Unknown version | Up-to-date | Old version |
|---|---|---|---|---|
| Apache | 55,088 | 14,569 (26.45%) | 19,527 (35.45%) | 20,992 (38.1%) |
| Microsoft IIS | 113,905 | n/a | n/a | n/a |
| Unknown | 12,706 | n/a | n/a | n/a |
| Scripting Software | | | | |
| PHP | 27,873 | 2,361 (8.5%) | 14,379 (51.6%) | 11,133 (39.9%) |

Table 4: Server version for landing sites. In the case of Microsoft IIS, we could not verify their version.

### 5.2 Drive-by Downloads via Ads

Today, the majority of Web advertisements are distributed in the form of third party content to the advertising web site. This practice is somewhat worrisome, as a web page is only as secure as it's weakest component. In particular, even if the web page itself does not contain any exploits, insecure Ad content poses a risk to advertising web sites. With the increasing use of Ad syndication (which allows an advertiser to sell advertising space to other advertising companies that in turn can yet again syndicate their content to other parties), the chances that insecure content gets inserted somewhere along the chain quickly escalates. Far too often, this can lead to web pages running advertisements to untrusted content. This, in itself, represents an attractive avenue for distributing malware, as it provides the adversary with a way to inject content to web sites with large visitor base without having to compromise any web server.

To assess the extent of this behavior, we estimate the overall contribution of Ads to drive-by downloads. To do so, we construct the malware delivery trees from all detected malicious `URLs` following the methodology described in Section 3. For each tree, we examine every intermediary node for membership in a set of $2,000$ well known advertising networks. If any of the nodes qualify, we count the landing site as being infectious via Ads. Moreover, to highlight the impact of the malware delivered via Ads relative to the other mechanisms, we weight the landing sites associated with Ads based on the frequency of their appearance in Google search results compared to that

[3] We consider a version as outdated if it is older than the latest corresponding version released by January, 2007 (the start date for our data collection).

of all landing sites. Figure 5 shows the percentage of landing sites belonging to Ad networks. On average, 2% of the landing sites were delivering malware via advertisements. More importantly, the overall weighted share for those sites was substantial—on average, 12% of the overall search results that returned landing pages were associated with malicious content due to unsafe Ads. This result can be explained by the fact that Ads normally target popular web sites, and so have a much wider reach. Consequently, even a small fraction of malicious Ads can have a major impact (compared to the other delivery mechanisms).

Another interesting aspect of the results shown in Figure 5 is that Ad-delivered drive-by downloads seem to appear in sudden short-lived spikes. This is likely due to the fact that Ads appearing on several advertising web sites are centrally controlled, and therefore allow the malicious content to appear on thousands of web sites sites almost instantaneously. Similarity, once detected, these Ads are removed simultaneously, and so disappear as quickly as they appeared. For this reason, we notice that drive-by downloads delivered by other content injection techniques (*e.g.,* individual web servers compromise) have more lasting effect compared to Ad delivered malware, as each web site must be secured independently.

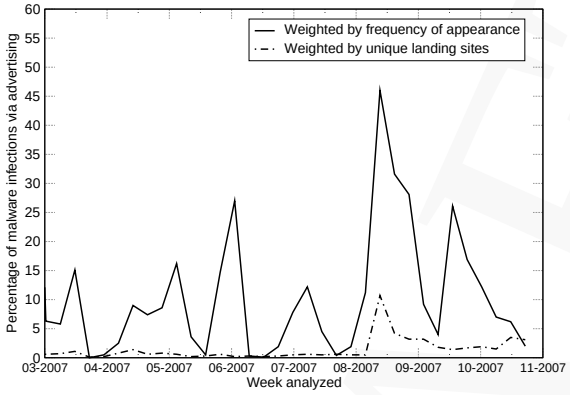



Figure 5: Percentage of landing sites potentially infecting visitors via malicious advertisements, and their relative share in the search results.

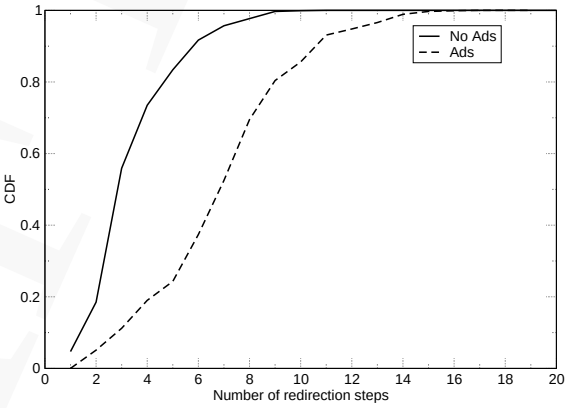



Figure 6: CDF of the number of redirection steps for Ads that successfully delivered malware.

The general practice of Ad syndication contributes significantly to the rise of Ad delivered malware. Our results show that overall 75% of the landing sites that delivered malware via Ads use multiple levels of Ad syndication. Inspecting the delivery trees that featured syndication reveals a total of 55 unique Ad networks participating in these trees. We further studied the relative role of the different networks by evaluating the frequency of appearance of each Ad network in the malware delivery trees. Interestingly, our results show that few networks appear in most of the malware delivery trees. In particular, one network appeared in 60% of the chains that delivered malware via Ads. Closer inspection revealed that this network belongs to a major Ad provider. Consequently, the high involvement of this network is likely due to its general prominence in the market.

To further understand how far trust would have to extend in order to limit the Ad delivered drive-by downloads, we plot the distribution of the path length from the landing site leading to the malware distribution sites for each delivery tree. The edges connecting the nodes in these paths reflect the number of redirects a browser has to follow before receiving the final payload. Hence, for syndicated Ads that delivered malware the path length is indicative of the number of

syndication steps before reaching the final Ad; in our case, the malware payload. Figure 6 shows the distribution of the number of redirects for syndicated Ads that delivered malware relative to the other malicious landing `URLs` . The results are quite telling: malware delivered via Ads exhibits longer delivery chains, in 50% percent of all cases, more than 6 redirection steps were required before receiving the malware payload. Clearly, it is increasingly difficult to maintain trust along such long delivery chains.

Finally we further elucidate this problem via an interesting example from our data corpus. The landing page in our example refers to a Dutch radio station's web site. The radio station in question was showing a banner advertisement from a German advertising site. Using JavaScript, that advertiser redirected to a prominent advertiser in the US, which in turn redirected to yet another advertiser in the Netherlands. That advertiser redirected to another advertisement (also in the Netherlands) that contained obfuscated JavaScript, which when un-obfuscated, pointed to yet another JavaScript hosted in Austria. The final JavaScript was encrypted and redirected the browser via multiple `IFRAME`s to *adxtnet.net*, an exploit site hosted in Austria. This resulted in the automatic installation of multiple Trojan Downloaders. While it is unlikely that the initial advertising companies were aware of the malware installations, each redirection gave another party control over the content on the original web page—with predictable consequences.

# 6 Properties of the Malware Distribution Infrastructure

In this section, we explore various properties of the hosting infrastructure for web malware. In particular, we explore the size of of the malware distribution networks, and examine the distribution of binaries hosted across sites. We expand on each in turn.

## 6.1 Size of the Malware Distribution Networks

Recall that a malware distribution network constitutes all the landing sites that point to a single malware distribution site. Using the methodology described in Section 3, we identified the distribution networks associated with each malware distribution site.

We first evaluate their size in terms of the total number of landing sites that point to them. Figure 7 shows the distribution of sizes for the different distribution networks. The graph reveals two main types of malware distribution networks: (1) networks that use only one landing site. These networks make up roughly 45% of all detected malware distribution sites. In about 40% of these networks the malware is hosted directly on the landing sites (*i.e.* the landing site is the same as the distribution site), and (2) networks that have multiple landing sites. As the graph shows, these networks can grow to have well over 21,000 landing sites pointing to them.

## 6.2 IP Space Locality

We also examined the network location of the malware distribution servers and the landing sites linking to them. Figure 8 shows that the malware distribution sites are concentrated in a limited number of /8 prefixes. About 70% of the malware distribution sites have IP addresses within `58.* -- 61.*` and `209.* -- 221.*` network ranges. Interestingly, Anderson *et al.* [3] observed comparable IP space concentrations for the scam hosting infrastructure. The landing sites, however exhibit relatively more IP space diversity; Roughly 50% of the landing sites fell in the above ranges. We further investigated the Autonomous System (AS) locality of the malware distribution sites by

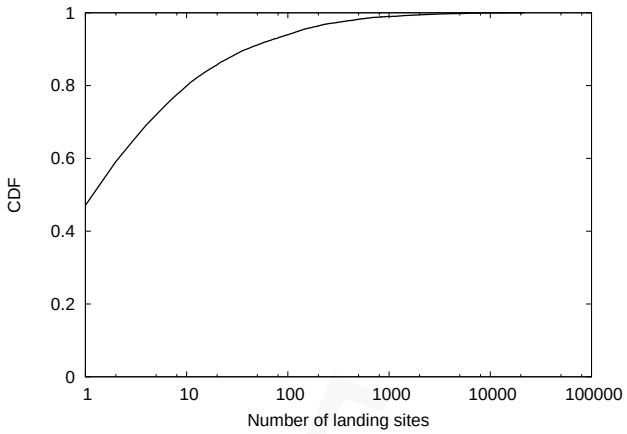



Figure 7: CDF of the number of landing sites pointing to a particular malware distribution site.

mapping their IP addresses to the AS responsible for the longest matching prefixes for these IP addresses. We use the latest BGP snapshot from Routeviews [20] to do the IP to AS mapping. Our results show that all the malware distribution sites' IP addresses fall into only 500 ASes. Figure 9 shows the cumulative fraction of these sites across the 500 ASes hosting them (sorted in descending order by the number of sites in each AS). The graph further shows the highly nonuniform concentration of the malware distribution sites— 95% of these sites map to only 210 ASes. Finally, the results of mapping the landing sites (not shown) produced 2,517 ASes with 95% of the sites falling in these 500 ASes.

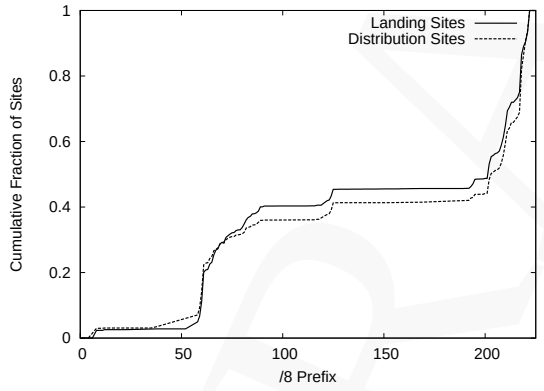



Figure 8: The cumulative fraction of malware distribution sites over the /8 IP prefix space.

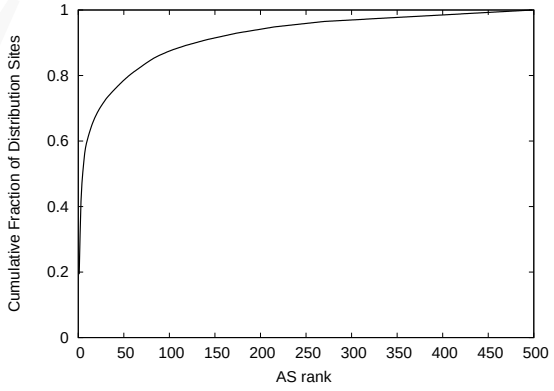



Figure 9: The cumulative fraction of the malware distribution sites across the different ASes.

### 6.3 Distribution of Malware Binaries Across Domains

The distribution of malware across domains also gives rise to some interesting insights. Figure 10 shows the distribution of the number of unique malware binaries (as inferred from their MD5 hash) downloaded from each malware distribution site. As the graph shows, approximately 42% of the distribution sites delivered a single malware binary. The remaining distribution sites hosted

multiple distinct binaries over their observation period in our data, with 3% of the servers hosting more than 100 binaries. In many cases, we observed that the multiple payloads reflect deliberate obfuscation attempts to evade detection. Finally, we observed a number of cases where the same hash was hosted on multiple distribution servers. This suggests that these distribution servers were controlled by the same entity. Next, we take a more in-depth analysis by studying the different forms of relationships among different distribution networks.

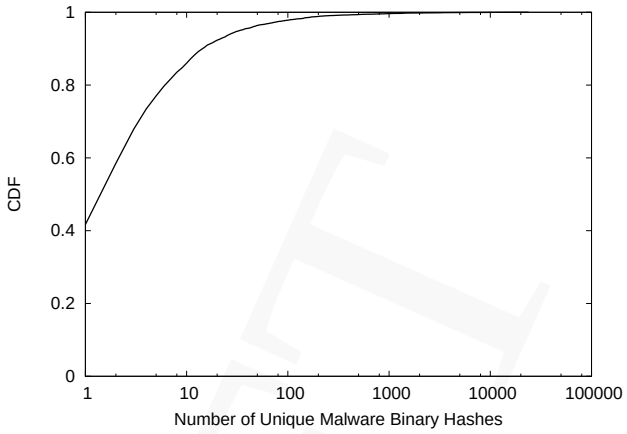



Figure 10: CDF of the number of unique binaries downloaded from each malware distribution site.

### 6.4 Relationships Among Different Malware Distribution Networks

We now examine some of the relationships between the different malware distribution networks we observed throughout our measurement period. Specifically, we investigate the common properties of the hosting infrastructure across the malware distribution sites. We also evaluate the degree of overlap among the landing sites linking to the different malware distribution sites. Finally, we investigate whether the malware content is replicated among different distribution servers.

**Malware hosting infrastructure.** Throughout our measurement period we detected 9,430 malware distribution sites. In 90% of the cases each site is hosted on a single IP address. The remaining 10% sites are hosted on IP addresses that host multiple malware distribution sites. Our results show IP addresses that hosted up to 210 malware distribution sites. Closer inspection revealed that these addresses refer to public hosting servers that allow users to create their own accounts. These accounts appear as sub-folders of the the virtual hosting server DNS name (*e.g.,* `512j.com/akgy`, `512j.com/alavin`, `512j.com/anti`) or in many cases as separate DNS aliases that resolve to the IP address of the hosting server. We also observed several cases where the hosting server is a public blog that allows users to have their own pages (*e.g.,* `mihanblog.com/abadan2`, `mihanblog.com/askbox`).

**Overlapping landing sites.** We further evaluate the overlap between the landing sites that point to the different malware distribution sites. To do so, we calculate the pairwise intersection between the sets of the landing sites pointing to each of the distribution sites in our data set. For a distribution network $i$ with a set of landing sites $X_i$ and network $j$ with the set of landing sites $X_j$, the normalized pairwise intersection of the two networks, $C_{i,j}$, is calculated as,

$$C_{i,j} = \frac{|X_i \cap X_j|}{|X_i|} \quad (1)$$

Where $|X|$ is the number of elements in the set $X$. Interestingly, our results showed that 80% of the distribution networks share at least one landing page. Figure 11 shows the normalized pair-wise landing sets intersection across these distribution networks. The graph reveals a strong overlap among the landing sites for the related network pairs. These results suggest that many landing sites are either shared among multiple distribution networks. For example, in several cases we observed landing pages with multiple `IFRAME`s linking to different malware distribution sites. Finally, we note that the sudden jump to a pair-wise score of one is mostly due to network pairs in which the landing sites for one network are a subset of those for the other network.

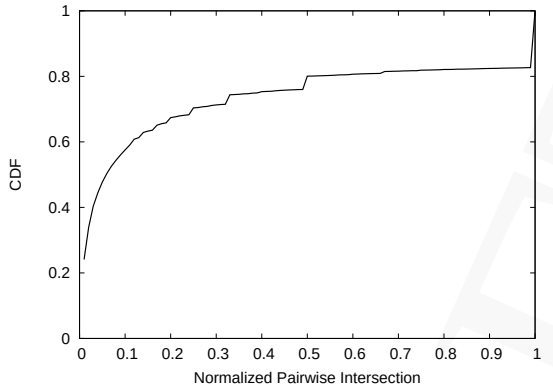



Figure 11: CDF of the normalized pairwise intersection between landing sites for the different malware distribution networks.

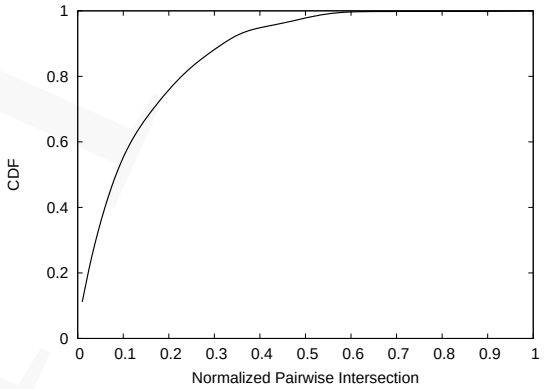



Figure 12: CDF of the normalized pairwise intersection between malware hashes for the different malware distribution networks.

**Content replication across malware distribution sites.** We finally evaluate the extent to which malware is replicated across the different distribution sites. To do so, we use the same metric in Equation 1 to calculate the normalized pairwise intersection of the set of malware hashes served by each pair of distribution sites. Our results show that in 25% of the malware distribution sites, at least one binary is shared between a pair of sites. While malware hashes exhibit frequent changes as a result of obfuscation, our results suggest that there is still a level of content replication across the different sites. Figure 12 shows the normalized pair-wise intersection of the malware sets across these distribution networks. As the graph shows, binaries are less frequently shared between distribution sites compared to landing sites.

# 7 Post Infection Impact

Recall that upon visiting a malicious `URL`, the browser downloads the initial exploit. The exploit (in most cases, `javascript`) targets a vulnerability in the browser or one of its plugins and takes control of the infected system, after which it retrieves and runs the malware executable(s) downloaded from the malware distribution site. Rather than inspecting the behavior of each phase in isolation, our goal is to give an overview of the collective changes that happen to the system state after visiting a

malicious URL . Figure 13 shows the distribution of the number of Windows executables downloaded after visiting a malicious URL as observed from monitoring the interaction between the browser and the malware distribution site. As the graph shows, visiting malicious URLs can lead to a large number of downloads (8 on average, but as large as 60 in the extreme case).

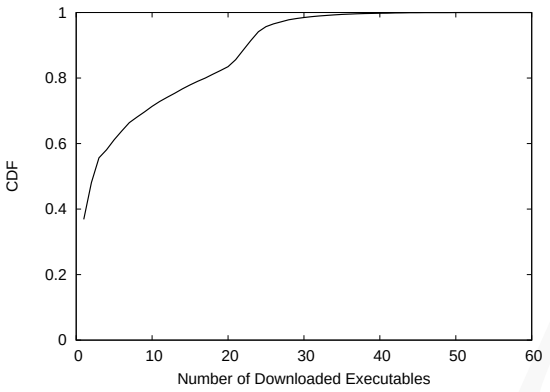



Figure 13: CDF of the number of downloaded executables as a result of visiting a malicious URL

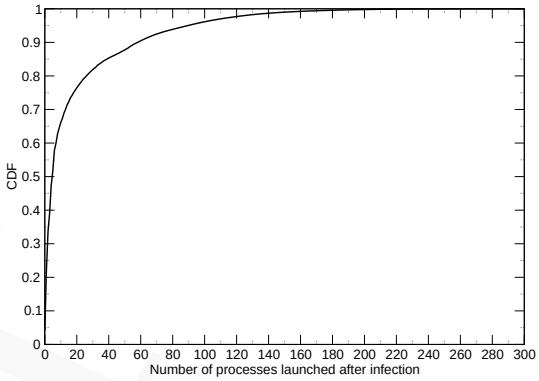



Figure 14: CDF of the number of processes started after visiting a malicious URL

Another noticeable outcome is the increase in the number of running processes on the virtual machine. This increase is associated with the automatic execution of binaries. For each landing URL , we collected the number of processes that were started on the guest operating system after being infected with malware. Figure 14 shows the CDF of the number of processes launched after the system is infected. As the graph shows visiting malicious URLs produces a noticeable increase in the number of processes, in some cases, inducing so much overhead that they "crashed" the virtual machine.

Additionally, we examine the type of registry changes that occur when the malware executes. Overall, we detected registry changes after visiting 57.5% of the landing pages. We divide these changes into the following categories: *BHO* indicates that the malware installed a Browser Helper Object that can access privileged state in the browser; *Preferences* means that the browser home page, default search engine or name server where changed by the malware; *Security* indicates that malware changed firewall settings or even disabled automatic software updates; *Startup* indicates that the malware is trying to persist across reboots. Notice that these categories are not mutually exclusive (*i.e.,* a single malicious URL may cause changes in multiple categories). Table 5 summarizes the percentage of registry changes per category. Notice that "Startup" changes are more prevalent indicating that malware tries to persist even after the machine is rebooted.

| Category | BHO | Preferences | Security | Startup |
|---|---|---|---|---|
| URLs % | 6.99% | 23.5% | 36.18% | 51.27% |

Table 5: Registry changes from drive-by downloads.

In addition to the registry changes, we analyzed the network activity of the virtual machine post infection. In our system, the virtual machines are allowed to perform only DNS and HTTP connections. Table 6 shows the percentage of connection attempts per destination port. Even though we omit the HTTP connections originating from the browser, HTTP is still the most

prevalent port for malicious activity post-infection. This is due to "downloader" binaries that fetch, in some cases, up to 60 binaries over HTTP. We also observe a significant percentage of connection attempts to typical IRC ports, accounting for more than 50% of all non-HTTP connections. As a number of earlier studies have already shown (e.g., [4, 16, 6, 18, 19, 10]), the IRC connection attempts are most likely for unwillingly (to the owner) adding the compromised machine to an IRC botnet, confirming the earlier conjecture by Provos *et al.* [17] regarding the connection between web malware and botnets.

| Protocol/Port | HTTP (80, 8080) | IRC (6660-7001) | FTP (21) | UPnP (1900) | Mail (25) | Other |
|---|---|---|---|---|---|---|
| Connections % | 87% | 8.3% | 0.9% | 0.8% | 0.75% | 2.25% |

Table 6: Most frequently contacted ports directly by the downloaded malware.

### 7.1 Anti-virus engine detection rates

As we discussed earlier, web based malware uses a *pull-based* delivery mechanism in which a victim is required to visit the malware hosting server or any `URL` linking to it in order to download the malware. This behavior puts forward a number of challenges to defense mechanisms (*e.g.,* malware signature generation schemes) mainly due to the inadequate coverage of the malware collection system. For example, unlike active scanning malware which uses a *push-based* delivery mechanism (and so sufficient placement of honeypot sensors can provide good coverage), the web is significantly more sparse and, therefore, more difficult to cover.

In what follows, we evaluate the potential implications of the web malware delivery mechanism by measuring the detection rates of several well known anti-virus engines [4]. Specifically, we evaluate the detection rate of each anti-virus engine against the set of *suspected* malware samples collected by our infrastructure. Since we can not rely on anti-virus engines, we developed a heuristic to detect these suspected binaries before subjecting them to the anti-virus scanners. For each inspected `URL` via our in-depth verification system we test whether visiting the `URL` caused the creation of at least one new process on the virtual machine. For the `URL`s that satisfy this condition, we simply extract any binary [5] download(s) from the recorded HTTP response and "flag" them as suspicious.

We applied the above methodology to identify suspicious binaries on a daily basis over a one month period of April, 2007. We subject each binary for each of the anti-virus scanners using the latest virus definitions on that day. Then, for an anti-virus engine, the detection rate is simply the number of detected (flagged) samples divided by the total number of suspicious malware instances inspected on that day. Figure 15 illustrates the individual detection rates of each of the anti-virus engines. The graph reveals that the detection capability of the anti-virus engines is lacking, with an average detection rate of 70% for the best engine. These results are disturbing as they show that even the best anti-virus engines in the market (armed with their latest definitions) fail to cover a significant fraction of web malware.

**False Positives.** Notice that the above strategy may falsely classify benign binaries as malicious. To evaluate the false positives, we use the following heuristic: we optimistically assume that all suspicious binaries will eventually be discovered by the anti-virus vendors. Using the set of suspicious

[4]For contractual reasons, we keep the identities of the anti-virus vendors anonymous.

[5]We only restrict our analysis to Windows executables identified by searching for PE headers in each payload.

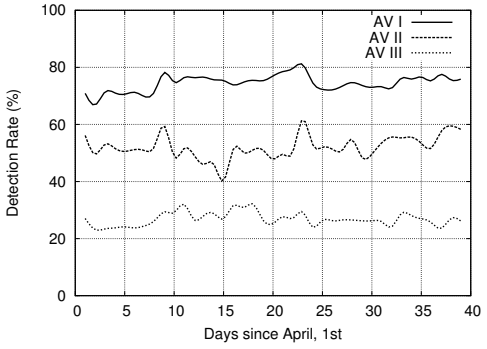



Figure 15: Detection rates of 3 anti-virus engines.

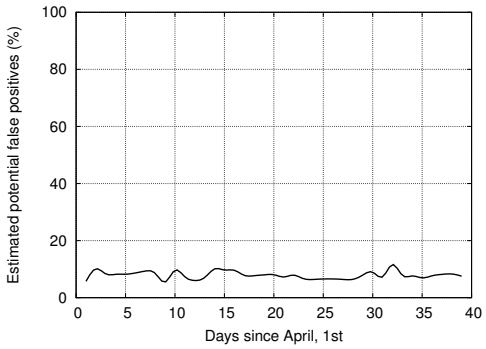



Figure 16: Estimated false positive rate.

binaries collected over a month historic period, we re-scan all undetected binaries two months later (in July, 2007) using the latest virus definitions. Then, all undetected binaries from the rescanning step are considered false positives. Figure 16 shows the results of applying this heuristic for the binaries analyzed earlier. Under our assumptions, the earlier analysis is fairly accurate with false positive rates of less than 10%. We further investigated a number of binaries identified as false positives and found that a number of popular installers exhibit a behavior similar to that of drive-by downloads, where the installer process first runs and then downloads the associated software package. To minimize the impact of false positives, we created a white-list of all known benign downloads, and all binaries in the white-list are exempted from the analysis in this paper.

Of course, we are being overly conservative here as our heuristic does not account for binaries that are never detected by any anti-virus engine. However, for our goals this method produces an upper bound for the resulting false positives. As an additional benchmark we asked for direct feedback from anti-virus vendors about the accuracy of the undetected binaries that we (now) share with them. On average, they reported about 6% false positives in the shared binaries, which is within the bounds of our prediction.

# 8 Related Work

Virtual machines have been used as honeypots for detecting unknown attacks by several researchers [2, 14, 15, 22, 23]. Although, honeypots have traditionally been used mostly for detecting attacks against servers, the same principles also apply to client honeypots (e.g., an instrumented browser running on a virtual machine). For example, Moshchuk *et al.* used client-side techniques to study spyware on the web (by crawling 18 million `URLs` in May 2005 [15]). Their primary focus was not on detecting drive-by downloads, but in finding links to executables labeled spyware by an adware scanner. Additionally, they sampled 45,000 `URLs` for drive-by downloads and showed a *decrease* over time. However, the fundamental limitation of analyzing the malicious nature of `URLs` discovered by "spidering" is that a crawl can only follow content links, whereas the malicious nature of a page is often determined by the web hosting infrastructure. As such, while the study of Moshchuk *et al.* provides valuable insights, a truly comprehensive analysis of this problem requires a much more in-depth crawl of the web. As we were able to analyze many billions of `URLs` , we believe our findings are more representative of the state of the overall problem.

More closely related is the work of Provos *et al.* [17] and Seifert *et al.* [21] which raised awareness of the threat posed by drive-by downloads. These works are aimed at explaining how different web page components are used to exploit web browsers, and provides an overview of the different exploitation techniques in use today. Wang *et al.* proposed an approach for detecting exploits against Windows XP when visiting webpages in Internet Explorer [23]. Their approach is capable of detecting zero-day exploits against Windows and can determine which vulnerability is being exploited by exposing Windows systems with different patch levels to dangerous URLs. Their results, on roughly 17,000 URLs, showed that about 200 of these were dangerous to users.

This paper differs from all of these works in that it offers a far more comprehensive analysis of the different aspects of the problem posed by web-based malware, including an examination of its prevalence, the structure of the distribution networks, and the major driving forces.

Lastly, malware detection via dynamic tainting analysis may provide deeper insight into the mechanisms by which malware installs itself and how it operates [8, 13, 24]. In this work, we are more interested in structural properties of the distribution sites themselves, and how malware behaves once it has been implanted. Therefore, we do not employ tainting because of its computational expense, and instead, simply collect changes made by the malware that do not require having the ability to trace the information flow in detail.

## 9 Conclusion

The fact that malicious URLs that initiate drive-by downloads are spread far and wide raises concerns regarding the safety of browsing the Web. However, to date, little is known about the specifics of this increasingly common malware distribution technique. In this work, we attempt to fill in the gaps about this growing phenomenon by providing a comprehensive look at the problem from several perspectives. Our study uses a large scale data collection infrastructure that continuously detects and monitors the behavior of websites that perpetrate drive-by downloads. Our in-depth analysis of over 66 million URLs (spanning a 10 month period) reveals that the scope of the problem is significant. For instance, we find that 1.3% of the incoming search queries to Google's search engine return at least one link to a malicious site.

Moreover, our analysis reveals several forms of relations between some distribution sites and networks. A more troubling concern is the extent to which users may be lured into the malware distribution networks by content served through online Ads. For the most part, the syndication relations that implicitly exist in advertising networks are being abused to deliver malware through Ads. Lastly, we show that merely avoiding the dark corners of the Internet does not limit exposure to malware. Unfortunately, we also find that even state-of-the-art anti-virus engines are lacking in their ability to protect against drive-by downloads. While this is to be expected, it does call for more elaborate defense mechanisms to curtail this rapidly increasing threat.

## Acknowledgments

We would like to thank Oliver Fisher, Dean McNamee, Mark Palatucci and Ke Wang for their help with Google's malware detection infrastructure.

DRAFT

## References


[1] The open directory project. See `http://www.news.com/2100-1023-877568.html`.

[2] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting Targeted Attacks Using Shadow Honeypots. August 2005.

[3] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proceedings of the USENIX Security Symposium*, August 2007.

[4] Paul Barford and Vinod Yagneswaran. *An Inside Look at Botnets*. Advances in Information Security. Springer, 2007.

[5] Jermey Bem, Georges Harik, Joshua Levenberg, Noam Shazeer, and Simon Tong. Large scale machine learning and methods. US Patent: 7222127.

[6] Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disturbing Botnets. In *Proceedings of the first Workshop on Steps to Reducing Unwanted Traffic on the Internet*, July 2005.

[7] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation*, pages 137–150, Dec 2004.

[8] Manuel Egele, Christopher Kruegel, Engin Kirda, Heng Yin, and Dawn Song. Dynamic Spyware Analysis. In *Proceedings of the USENIX Annual Technical Conference*, June 2007.

[9] Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, October 2007.

[10] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. BotHunter: Detecting Malware Infection through IDS-driven Dialog Correlation. In *Proceedings of the* $16^th$ *USENIX Security Symposium*, pages 167–182, 2007.

[11] Nagendra Modadugu. Web Server Software and Malware, June 2007. See `http://googleonlinesecurity.blogspot.com/2007/06/web-server-software-and-malware.html`.

[12] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial of Service Activity. In *Proceedings of* $10^{th}$ *USENIX Security Symposium*, August 2001.

[13] A. Moser, C. Kruegel, and E. Kirda. Exploring Multiple Execution Paths for Malware Analysis. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, May 2007.

[14] A. Moshchuk, T. Bragin, D. Deville, S.D. Gribble, and H.M. Levy. SpyProxy: Execution-based Detection of Malicious Web Content. August 2007.

[15] Alexander Moshchuk, Tanya Bragin, Steven Gribble, and Henry Levy. A crawler-based study of spyware in the web. In *Proceedings of Network and Distributed Systems Security Symposium*, 2006.

[16] Honeynet Project and Research Alliance. Know your enemy: Tracking Botnets, March 2005. See `http://www.honeynet.org/papers/bots/`.

[17] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. The Ghost in the Browser: Analysis of Web-based Malware. In *Proceedings of the first USENIX workshop on hot topics in Botnets (HotBots'07).*, April 2007.

[18] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC)*, pages 41–52, Oct., 2006.

[19] Anirudh Ramachandran, Nick Feamster, and David Dagon. Revealing Botnet Membership using DNSBL Counter-Intelligence. In *Proceedings of the 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, July 2006.

[20] The Route Views Project. `http://www.antc.uoregon.edu/route-views/`.

[21] Christian Seifert, Ramon Steenson, Thorsten Holz, Yuan Bing, and Michael A. Davis. Know Your Enemy: Malicious Web Servers. `http://www.honeynet.org/papers/mws/`, August 2007.

[22] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Automated web patrol with strider honeymonkeys. In *Proceedings of Network and Distributed Systems Security Symposium*, pages 35–49, 2006.

[23] Yi-Min Wang, Yuan Niu, Hao Chen, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Strider honeymonkeys: Active, client-side honeypots for finding malicious websites. 2007. See `http://research.microsoft.com/users/shuochen/HM.PDF`.

[24] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In *Proceedings of the 14th ACM Conference of Computer and Communication Security*, October 2007.

DRAFT