Peering Through the iFrame

Brett Stone-Gross[§], Marco Cova[‡], Christopher Kruegel[§], and Giovanni Vigna[§]

[§]University of California, Santa Barbara
{bstone,chris,vigna}@cs.ucsb.edu

‡ University of Birmingham, United Kingdom
m.cova@cs.bham.ac.uk

***Abstract*—Drive-by-download attacks have become the method of choice for cyber-criminals to infect machines with malware. Previous research has focused on developing techniques to detect web sites involved in drive-by-download attacks, and on measuring their prevalence by crawling large portions of the Internet. In this paper, we take a different approach at analyzing and understanding drive-by-download attacks. Instead of horizontally searching the Internet for malicious pages, we examine in depth one drive-by-download *campaign*, that is, the coordinated efforts used to spread malware. In particular, we focus on the Mebroot campaign, which we periodically monitored and infiltrated over several months, by hijacking parts of its infrastructure and obtaining network traces at an exploit server.**

By studying the Mebroot drive-by-download campaign from the inside, we could obtain an in-depth and comprehensive view into the entire life-cycle of this campaign and the involved parties. More precisely, we could study the security posture of the victims of drive-by attacks (e.g., by measuring the prevalence of vulnerable software components and the effectiveness of software updating mechanisms), the characteristics of legitimate web sites infected during the campaign (e.g., the infection duration), and the *modus operandi* of the miscreants controlling the campaign.

I. Introduction

Drive-by-download attacks [11], [12] exploit vulnerabilities in web browsers and web browser components. These attacks are triggered when a victim uses her browser to load a web page that contains malicious code. To attract potential victims, attackers can prepare malicious web pages and distribute their URLs (e.g., by including links in spam mails). Alternatively, attackers can compromise existing web sites and inject malicious code into legitimate pages (e.g., by embedding iframes). Typically, the exploit code used in a drive-by-download attack is written in a client-side scripting language, such as JavaScript or ActionScript (as part of an Adobe Flash file). This code is directly run by the browser or executed with the help of a browser extension. When the exploit is successful, it downloads and installs malware on the victim's machine, frequently in an effort to recruit additional members for a botnet [10].

Over the last few years, drive-by-download attacks have become the most popular means used by cyber-criminals to compromise and infect hosts. The reason for the popularity of this type of attack is that direct attacks against hosts and their operating systems have become more difficult. This is in part due to the growing efforts by Microsoft to improve the security of its Windows products [8], but also due to the fact that users are increasingly shielded by firewalls and NAT devices (such as home routers).

Given the importance of drive-by-download attacks, researchers have recently proposed first steps to harden browsers against these exploits [5], [13]. Moreover, a number of papers have presented approaches to detect malicious web pages [7], [15] and to study their prevalence by crawling large portions of the Internet [11], [12]. Finally, follow-up work has studied the behavior of web-based malware once a machine is compromised [10]. However, malicious web pages and web-based malware do not live in isolation. Someone has to develop the code used for drive-by-download attacks, infect web sites, and set up exploit servers to which victims are redirected, so that they can download the latest malware instances. Also, malicious sites rely on visitors with vulnerable software components to carry out successful attacks. These aspects are all important pieces of a drive-by campaign, which we characterize as a coordinated effort of a group of cyber-criminals to propagate malware via drive-by-download attacks.

So far, researchers have focused on malicious web sites and their prevalence, mostly neglecting other aspects of drive-by-download campaigns. However, we believe that a more comprehensive study of these campaigns is important and has the potential of revealing a number of interesting facts about the various parties that are involved. In particular, we would like to understand in more detail the *modus operandi* of cyber-criminals, the ways in which they craft and update their exploit code, and the infrastructure they use to host their attacks. In addition, it is interesting to see how the exploit code is distributed and which web sites are targeted as part of a single campaign. Finally, it is important to study the (potential) victims of drive-by-download attacks on the Internet and determine the software they are using and the corresponding vulnerability surface.

In this paper, we present the results of an in-depth study of the Mebroot drive-by-download campaign. Our study was carried out over one year, covering different time periods from May to September 2009, and one week in April 2010. During the four months starting in mid 2009, we seized control of parts of the infrastructure used by the cyber-criminals to spread Mebroot. In April 2010, we obtained direct network-level access (via a mirror port) to one of the exploit servers used by the Mebroot gang to serve drive-by download attacks. This allowed us to examine infected web sites, analyze the clients who visit these sites, and obtain a behind-the-scenes view of the infrastructure used by these criminals. The main contribution of the paper is the analysis of a large-scale, successful, long-lasting drive-by campaign from unique vantage points. By studying the operations of the miscreants running the Mebroot campaign, we can answer questions

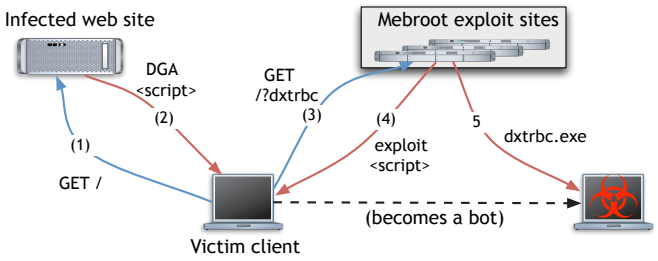

Figure 1. The Mebroot drive-by-download infrastructure.

about the actors and victims of drive-by-download attacks and the infrastructure that is used to carry out these attacks. We argue that our results are relevant to most existing drive-by campaigns. Therefore, a more thorough understanding of the processes and the lessons learned during our analysis will help to develop more effective mechanisms to combat these threats.

II. Background

The measurements described in this paper are performed on the Mebroot drive-by-download campaign. While a detailed description of Mebroot is outside the scope of this paper, we review here its key mechanisms and organizational units. Mebroot is a sophisticated piece of malware that infects the Master Boot Record (MBR) of victim machines to hide itself and download and bootstrap additional malware plugins [6]. One such plugin is Torpig, a bot that, in the recent past, has been very successful in harvesting sensitive data (e.g., credit card and banking information) from its victims [14].

Mebroot spreads through drive-by-download attacks, which start when a victim visits a legitimate web site that has been compromised by the Mebroot gang (Step 1 in Figure 1). We call these sites *infected web sites*. Pages on the infected site have been modified to contain JavaScript code that redirects the client to a site controlled by the Mebroot gang. This site, which we call the *exploit site*, then launches a number of exploits against vulnerabilities in the browser or its plugins.

In typical drive-by campaigns, the malicious code injected into infected web sites directly encodes the domain name of the site used to launch the exploits. For example, the injected code may be a static string that creates an `iframe` tag pointing to a page on a particular site (domain). In these cases, until the injected code is removed or otherwise modified, all visitors will be redirected to the same site referenced by the iframe tag. Mebroot follows a different, and more robust strategy, where the exploit site is not encoded in a static, fixed string, but it is dynamically computed by using a domain generation algorithm (DGA). This computation is carried out in the web browser of the victim (the DGA is implemented in JavaScript) when the infected page is retrieved (Step 2). With this scheme, different visits to the same infected page may trigger the generation of different domain names, and thus, cause the redirection to different exploit sites.

From the malware authors' perspective, it is more desirable to use DGAs for the domain names of exploit sites rather than relying on a fixed domain name, because this potentially improves the resilience of the drive-by-download campaign to take-down and blacklisting efforts. In fact, while these efforts may successfully block the current exploit site, they do not have the effect of permanently stopping the campaign or forcing the campaign authors to update the infected sites with a new version of the malicious code (redirecting to a fresh exploit site).

Mebroot's DGAs are seeded with the current date and, optionally, additional parameters such as Twitter trends. These inputs are hashed using simple, custom hash functions to produce as output a domain name. The victim is then automatically redirected to this domain, where the exploit site is hosted (Step 3). The exploit site employs Neosploit, a popular exploit pack, to fingerprint the victim's system, identify vulnerable applications on her machine, and serve the most appropriate exploits (Step 4). If any exploit is successful, the Mebroot malware is automatically downloaded from the exploit site and is executed on the victim's machine (Step 5).

III. Data Collection

Our approach for carrying out parts of our study relies on *sinkholing* some of the components used in this campaign (in particular, the exploit sites), passively monitoring the traffic directed to these sites, and actively fingerprinting the clients issuing these requests. The factor enabling our sinkholing is the existence of domains, generated by DGAs, that have not been registered by the Mebroot gang. We reverse-engineered the DGAs and found several domains names that were to serve as exploit sites and were still available. Of these, we registered 22 domain names. These domains were active on 56 distinct days over the 4-month interval, between May and September 2009. As a consequence of our sinkholing, when users visited an infected web site containing the Mebroot JavaScript code, instead of being redirected to an exploit server, they were redirected to a domain (and host) that was under our control.

As part of our experiments, we recorded all the requests directed at our sites. Furthermore, we served to visitors of our domains JavaScript code that fingerprinted the clients' machines to identify vulnerable software components. Finally, from the `Referer` header contained in client requests, we were able to identify infected web sites, and leveraging this knowledge, were able to monitor and track site infections. Hence, we built two data sets. The first contains the requests issued by the victims of Mebroot attacks that connected to the sites under our control, and the results of the active fingerprinting we performed on them. In total, this data set includes data from 354,360 distinct IP addresses. The second data set includes data about 6,541 distinct infected web sites. For each of these web sites, we recorded name, address, and the content of the pages that redirected clients to our sinkhole.

IV. Mebroot Infiltration

In this section, we discuss the results of our infiltration of the Mebroot infrastructure. We begin our analysis by studying the visitors of Mebroot-infected web sites. We will

then examine the information that we obtained through our active measurement infrastructure, in particular, the vulnerable software run by these hosts.

Data Collection Methodology. After registering the Mebroot domain names, we pointed the corresponding DNS records at our servers. We then set up a web page to perform a number of measurements of the visitors to these sites. For convenience, we will refer to these visitors as *victims* because these users are involuntarily redirected to our site, and had we not purchased the exploit domain, may have been infected with malware.

In addition to counting the number of Mebroot victims, we also wanted to determine other characteristics of these clients, in particular, whether they were running vulnerable software, which could be targeted by the exploits used in the Mebroot or other drive-by-download campaigns. To detect the client software installed on a user's machine, we used techniques commonly used by web analytics services. More specifically, our web site served to visitors JavaScript code that tested for the presence of 45 ActiveX controls and browser plugins, including 39 that are known to be vulnerable to remote code execution. When possible, our script also collected the version of each tested plugin, by reading an appropriate property or method exposed by the component under review.

Since ActiveX controls only work in Microsoft Internet Explorer, we detected other browser plugins via the `navigator` browser plugin array. This array exposes a convenient interface to iterate through and detect various plugins using only JavaScript, and it exists in most popular browsers including Firefox, Safari, Chrome, and Opera.

Victim Characteristics. During the four-month monitoring period from May to September 2009, our server received 1,245,348 requests from 354,360 unique IP addresses. Of these, 559,627 requests from 339,150 distinct IP addresses were in a format consistent with the requests generated by Mebroot's JavaScript code. We attribute the remaining requests to web crawlers, and security researchers. In the following paragraphs, we will consider only those requests that were compatible with Mebroot's JavaScript code and that reported the results from our host fingerprinting script.

Geographic Locations. We mapped each client IP to its respective country, using an IP-based geolocation database from ip2location.com. Visitors from the top 5 countries are shown in the rightmost column of Table I. Overall, we observed visitors from 201 countries all over the world.

Operating System	IPs (%)	Browser	IPs (%)	Country	IPs (%)
Windows XP	63.8%	IE 7	30.4%	US	27.2%
Windows Vista	23.1%	Firefox 3	25.6%	IT	8.55%
Mac OSX 10.5	4.56%	IE 6	17.3%	IN	5.29%
Mac OSX 10.4	2.01%	IE 8	8.94%	UK	4.75%
Linux	0.94%	Firefox 2	2.80%	ES	4.70%

Table I
TOP-5 OPERATING SYSTEMS, BROWSERS, AND COUNTRIES OF VICTIMS.

Operating Systems. There were 39 different operating systems that we observed. Many of these were mobile phone devices running embedded operating systems and game consoles. Overall, Windows XP was by far the most common operating system, accounting for more than 63% of the visitors, with Windows Vista and Mac OSX Leopard being the second and third most prevalent operating systems, as shown in Table I. Note that the Mebroot malware is written only for Windows. However, web browsers and plugins may be vulnerable regardless of the operating system, and platform specific shellcode and malware could be delivered in future drive-by-download attacks.

Web Browsers. To determine a visitor's web browser, we analyzed the User-Agent field that is part of the HTTP headers and the navigator array (when supported by the browser). Overall, we observed 79 different web browser versions with 40,326 unique User-Agent strings. The large number of User-Agent strings is the result of modifications done by other software applications installed on visitors' systems.

The most common browsers were Internet Explorer, Firefox, Safari, Google Chrome, and Opera. For each of these browsers, we examined the percentage of visitors using the latest web browser versions. More specifically, we aimed to determine how long it takes for users to update their web browser when a new version is released. Note that we did not have daily statistics for browser updates during the entire four-month interval, but in many cases, we could clearly see when updates occurred by comparing trends before and after an update was released. Our results, shown in Figure 2, demonstrate that users of particular web browsers such as Internet Explorer and Safari take significantly longer to update than other browsers. This is likely due to the update mechanisms used by Microsoft and Apple, where browser updates are done as part of operating system updates. In contrast, the Chrome web browser automatically updates to the latest browser version regardless of whether the browser is being used at the moment, and the updates run silently in the background. As a result, more than 92% of users were running the latest version of Chrome at all times. In comparison, visitors using Firefox were running the latest version an average of 70.8% of the time. While Firefox automatically downloads updates by default, it prompts users to install these updates at their convenience. This is evident in Figure 2. Only about 43.9% of Opera users were using the latest version at any time even though it was released in early March 2009. This is likely due to the fact that Opera did not have a built-in update mechanism and consequently was the most cumbersome to update (users had to go to Opera's web site and manually download and install the latest version [4]).

Vulnerable Applications. In the next step, we examined the vulnerable applications installed on the computers of users who visited Mebroot-infected domains. That is, we focus on the potential number of victims that the Mebroot drive-by-download exploits would likely compromise, and the likelihood that a potential victim is running at least one vulnerable application that could potentially compromise her system.

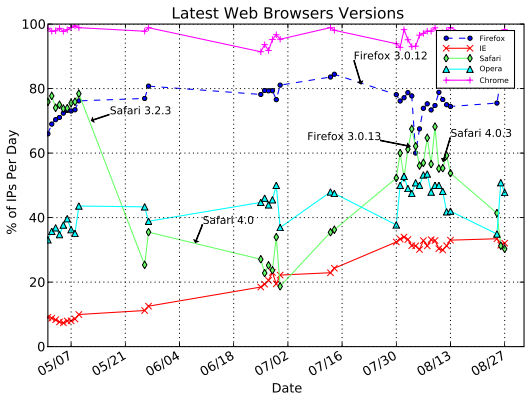

Figure 2. Percentage of victims with latest web browser versions.

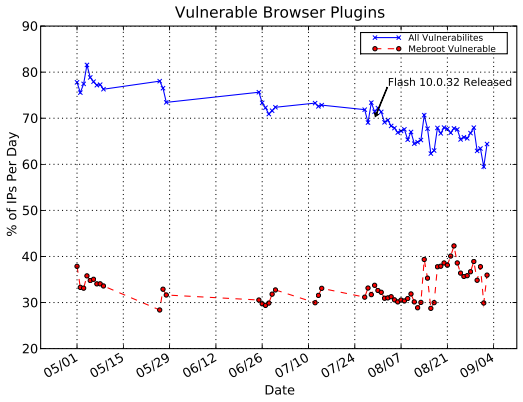

Figure 3. Percentage of victims with vulnerable browser plugins.

During the four-month monitoring period, the Mebroot exploits targeted Adobe Acrobat and Reader, Apple QuickTime, AOL's SuperBuddy, MDAC, and DirectShow. Figure 3 displays the number of victims per day who had one or more browser plugins that were vulnerable to the Mebroot exploits. The Adobe Acrobat/Reader component, targeted by the Mebroot exploit toolkit, was by far the most dangerous, since 32.1% of all visitors had a vulnerable version installed. Over these four months, between 25% and 42% of visitors were vulnerable to at least one of the Mebroot exploits at any given time. Thus, if we consider the maximum number of users that we observed during a single day, this particular drive-by-download campaign could have infected between 9,500 and 12,700 machines per day (assuming these users had not been previously infected and the exploits were reliable). We will show the actual effectiveness of Mebroot's exploits in Section V.

While Mebroot exploits targeted six vulnerabilities in our first monitoring period, there are considerably more browser plugins that were vulnerable. Thus, we checked a total of 39 ActiveX and browser plugins that are commonly found in exploit toolkits to determine what the potential success rate could be if a drive-by campaign had attempted a larger number of exploits. The potential success rates per day are shown in Figure 3. A large contributing factor behind the high percentage of vulnerable users is due to the ubiquity and vulnerability of the Adobe Flash Player, which in our data set was installed on 86% of all victims. Notice that when Flash version 10.0.32 was released at the end of July, there was an almost immediate reduction in about 10% of the vulnerable population.

Infected Web Sites. By analyzing the requests sent to our sinkholed domains, we were also able to identify infected web sites. More precisely, we inspected the value of the `Referer` header sent by clients to the sinkholed sites under our control. The assumption we make is that referrer URLs identify pages on (presumably) legitimate sites that have been compromised to redirect their visitors to the exploit-serving domains.

There are two potential issues with relying on the `Referer` header to identify infected sites. More specifically, the `Referer` header may be suppressed [1], or may be spoofed and set to arbitrary values. To assess the impact of missing or fake headers in our data set, we monitored a number of malware feeds and online forums for reports of Mebroot-infected sites. We found no site from these sources that was not also present in the data we collected. We also removed invalid URLs and used Wepawet, our tool to detect and analyze web-based malware [3], to discard URLs that were not infected.

Prevalence and Distribution of Infected Sites. In total, we received 381,768 requests that were Mebroot-compatible and that specified a referring page (contained a `Referer` entry). From the `Referer` data (after validation with Wepawet), we obtained 33,195 distinct URLs from 6,541 sites.

We wanted to determine the distribution of all sites across different functional categories (types of web sites). Unfortunately, we found that online directories, such as dmoz.org [9], only contained a minimal fraction of the domains we observed. Therefore, we randomly sampled 100 sites and classified them using a classification service [2]. Most sites (22%) were classified as business-related, 15% as technology-related, 10% as adult, 9% as travel, and 5% as sports related. In our experience, the classifier worked accurately, and these results resemble those of previous studies on the impact of browsing habits on the exposure to drive-by-download attacks [11].

Infection and Cleanup Dynamics. The last stage in the infection of a web site is when the malicious code is finally detected by web masters and removed. Cleaning up infected sites takes a long time. Indeed, in the worst case, infected sites are never cleaned up. In fact, 900 sites out of the 4,927 that we monitored were still infected at the end of our monitoring period. Figure 4 focuses on the web sites that were infected and cleaned up. We see that after roughly 25 days, only 50% of these web sites had been cleaned up. A cleanup rate of 80% is achieved only after 45 days. Remediating an infection, rather than identifying its root cause, does not lead to a lasting

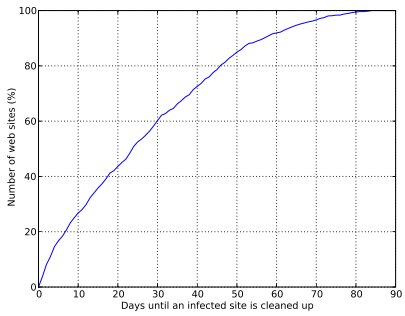

Figure 4. Infection duration for web sites that were infected and cleaned up.

cleanup. We found 467 sites where the Mebroot code was removed and injected again.

V. Network-level Monitoring

We were able to gain even more insight into the effectiveness of the Mebroot drive-by campaign when we were given access to a mirror port of a switch connected to an exploit server for one week in April 2010. Using this mirror port, we could monitor all requests sent and received by this server. In total, we collected over 300GB of data.

Based on the exploit and download requests, we can determine the number of machines that were actually infected, the exploits that were successful in compromising a host, and the versions of the vulnerable browser components. The Mebroot JavaScript code sets a cookie on the targeted machine to identify and avoid re-attacking victims that have already been compromised. The cookie is active for one week on a targeted machine. Since this monitoring period spanned one week, we make the assumption that there is approximately a one-to-one correlation between an IP address and a victim machine. Overall, we observed requests from 202,879 unique IP addresses. Of these, 45,816 hosts were exploited, and we observed the download of the malware binary. That is, the exploits compromised more than 22.6% of the machines (assuming antivirus/firewall software did not subsequently block the execution of the malware). We would like to point out that during our monitoring, the Mebroot controllers used two different servers, and hence, we had only visibility into roughly half of the campaign. If we assume that the other server received a similar number of visitors and the exploit success rates are comparable, we can estimate that more than 91,000 machines were infected in a single week from this drive-by-download campaign alone. These results are consistent with our estimations from the previous monitoring periods.

Overall, exploits that targeted Java applets compromised the most machines, accounting for 40.5% of the infections, followed by an IE exploit (33.9%), multiple Adobe Reader exploits (24.7%), and a DirectShow (7.5%) exploit. Notice that these percentages add up to more than 100% because roughly 6% of the compromised machines were exploited multiple times through multiples vulnerabilities.

VI. Conclusions

In this paper, we have performed an in-depth analysis of the drive-by-download campaign used to distribute the Mebroot malware. Our approach was based on infiltrating the campaign by sinkholing the web sites used by the Mebroot authors to launch exploits against their victims, and by directly monitoring the network traffic on an exploit server. From these internal vantage points, we had direct visibility of the victims of the campaign and of the infected web sites that were unwittingly redirecting their visitors to the exploit sites.

We performed a number of measurements to improve our understanding of drive-by-download campaigns. On the client-side, we found that drive-by-download campaigns affect a significant number of web users and that an alarming fraction of these users rely on vulnerable systems, such as old versions of the browser or vulnerable additional components. Similarly, on the server-side, we found that web masters of infected sites are slow at removing malicious code injected in their pages and often, when they do, they are prone to be infected again.

References

[1] A. Barth, C. Jackson, and J. Mitchell. Robust Defenses for Cross-Site Request Forgery. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2008.

[2] CONTEXTin.com. URLclassifier – Classify a Web Page Using Advanced Semantic Analysis. http://www.urlclassifier.com/URLClassifierWS.

[3] M. Cova, C. Kruegel, and G. Vigna. Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. In *Proceedings of the World Wide Web Conference (WWW)*, 2010.

[4] T. Duebendorfer and S. Frei. Why Silent Updates Boost Security. Technical report, ETH Zurich, 2009.

[5] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda. Defending Browsers against Drive-by Downloads: Mitigating Heap-Spraying Code Injection Attacks. In *Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2009.

[6] P. Kleissner. Analysis of Sinowal. http://web17.webbpro.de/index.php?page=analysis-of-sinowal, 2008.

[7] J. Ma, L. Saul, S. Savage, and G. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *ACM SIGKDD Conference*, 2009.

[8] Microsoft Corporation. Security Development Lifecycle (SDL). http://msdn.microsoft.com/en-us/security/cc448177.aspx, 2009.

[9] Netscape. ODP – Open Directory Project. http://www.dmoz.org.

[10] M. Polychronakis, P. Mavrommatis, and N. Provos. Ghost Turns Zombie: Exploring the Life Cycle of Web-based Malware. In *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[11] N. Provos, P. Mavrommatis, M. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the USENIX Security Symposium*, 2008.

[12] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The Ghost in the Browser: Analysis of Web-based Malware. In *Proceedings of the USENIX Workshop on Hot Topics in Understanding Botnet*, 2007.

[13] P. Ratanaworabhan, B. Livshits, and B. Zorn. Nozzle: A Defense Against Heap-spraying Code Injection Attacks. In *Proceedings of the USENIX Security Symposium*, 2009.

[14] B. Stone-Gross, M. Cova, L. Cavallaro, R. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2009.

[15] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Proceedings of the Symposium on Network and Distributed System Security (NDSS)*, 2006.