Survey of Malware Distribution Networks

Zahra Behfarshad
Electrical and Computer Engineering
Faculty of Applied Science
UBC, Canada
Email: janab@ece.ubc.ca

***Abstract*—A vast spectrum of people are interacting with Internet, either to perform their daily activities online such as banking, or to simply surf the web. As more people are connected not only they become more dependent on it, yet their systems can be the target of adversaries with different aims. Recently, one of the attacks that has become very common in the network society, is the distribution of malware by malware distribution networks or MDNs. The main components in a malware distribution network, are the landing pages and malware repository. The goal of the MDNs is to redirect the user to the malware repository through the landing pages where a malicious binary is downloaded by the user and thus, the adversary can extend his attack. In this paper, a complete overview of the malware distribution network is provided, explaining many different aspects regarding the malware distribution network.**

I. Introduction

A vast spectrum of people now take advantage of what was going to be only used by the scientists and military, the Internet. Today, the Internet is a door to unlimited knowledge and information with can be easily used by any person throughout the world at any time regardless of the time zone and geographical issues. Although it is still growing, yet it has taken a mighty grasp on people's lives from children to elders, making them more dependent on it each day and the interesting fact is that people have no idea what they would do without it.

The World Wide Web(WWW), a collection of all existing technologies which are constructed upon the Internet, basically simplifies the delivery of a wide range of services to any user from simple and elementary services such as reading the news to complicated and critical services like online military services. In order to provide these services to the customers, different technologies are applied, enabling web browsers to become one of the most important communication techniques in this regard. Web browsers play an important role in allowing users to easily interact with the World Wide Web by traversing, retrieving and finally presenting the related topic(s) for them. Whilst, one may say the Internet is a powerful resource to gain knowledge, yet the Internet has a darker side as well.

Many people benefit from using the Internet since they can simply access so many information in little time this is one of the strongest advantages of it. However, it is also a reflection of the society containing both good and bad impacts concluding to individual and community concerns. One of the most important problems in using the Internet is related to the user's security. Although many concepts are defined for user security and they might have different levels of obtaining it, yet one common aspect between all is how to provide it, specially while they are using online services. Without security, the user might or even might not encounter a threat which can somehow result in gaining access to the user's assets without his notice and thus, allows the adversary to attack his system or to simply carry out another different kind of attack that leads to losing everything which possesses great value like bank accounts. One type of attack out of the so many existing attacks is *malware* which is installed and spread easily.

Malware, short for malicious software is as old as software itself where the programmers intentions were to produce malwares for higher level and low level reasons, that are perform and conduct experiments and to prank with friends respectively. Malware is seen in many different forms and the most common are viruses, trojans, worms and spywares.

In the earliest days of malware's history, a virus would simply hide itself in files on the victim's local system and just wait to be transferred to new hosts by the propagation of the infected files. However, these viruses were not self-propagated.[1] As the technology improves each day so does the attack models, yet the main objective and goal remains constant. Since when many people today connect regularly to their systems in order to use Internet, the previous malware attack model has modified, shifting towards the network with goal of infecting the network by worms. The general idea of

these network worms are to exploit the vulnerabilities that exist in the network services in order to easily spread from one host to another[2]-[5]. However, this action has been less effective due to the deployment of Network Address Translation(NAT) and firewalls which prevent any untrusted network traffic [6]. In all cases, the web is the root cause since it is the main delivery mechanism for spreading malware. In order to prevent such attacks, the perimeter of network security and their devices has extended massively leading to complex web browsers. In other words, the common browsers used today such as Google Chrome, Mozilla Firefox, Microsoft Internet Explorer and Apple Safari carry out a variety of complex client side operations which are encoded in ECMAScript[7] based languages. To handle these complicated contents, web browsers use code plug-ins which are written and maintained by third-party companies. However, it should be noticed that by increasing complexity to any subject, many vulnerabilities can appear and thus, easily be exploited.
In this paper, I survey the area of malware distribution networks also known as MDNs. The goal of this paper is to provide a complete overview in different aspcets of MDNs including how the attack takes place, the features, structure of these MDNs and how to identify and prevent them. In Section 1, the attack model is explained completely. In Section 2 the structure of malware distribution networks is covered. In Section 3 the features that all MDNs should include and in Section 4 the adversarial information retrieval system and their challenges are explained. In Section 5, the methods in identifying the MDNs is discussed and finally, a conclusion is reached in Section 6.

II. Malware Distribution Network

One of the attack models in the network area is the use of malware distribution networks. This attack model is quite simple from one perspective since it takes advantage of our daily online searches. The main goal of this attack is to download and execute a malicious binary on the victim's computer system without his notice and consent after some redirections. The basic searching method is very obvious: A user is searching for information related to a specific subject online and as usual he will use different search engines and finally, the search engines will retrieve and display the results for him. The user might find a result that seems useful and thus by clicking on it, he will be redirected to that specific web site immediately.
In all malware distribution networks one or several malicious agents are collaborating with one another. Their intentions is to attract the user somehow so he can be redirected to a new web site that is called *Landing Pages or Intermediate Pages*. The landing pages are consisted of pages with code injected by the adversary for the specific purpose and intent of his. Thus, the landing pages are part of the adversary's plan but the user has not idea of the true mask and simply believes it is a new page that will guide him towards that specific subject he has searched. The landing page's role is to redirect the user continuously in different ways until at last he arrives at the core of the MDNs, the *Malware Repository*. Once at the malware repository, an exploit or a social engineering scam will be attempted, both potentially resulting in the download and installation of a malicious executable binary and thus, the attack is complete.
For a better understanding of how the attack takes place an image is displayed in Figure 1. As can be seen, Alice is an average Internet user that is using a search query in a search engine to find information regarding a specific subject. One of the results is clicked by Alice and she is redirected to a compromised landing page. Alice visits the compromised site, which leads her to connected redirections until she has eventually arrived at the malware repository [1].
MDNs are comprised of several components which can be dynamic in nature: the path of redirection from the landing page to the malicious executable(i.e., the malware delivery tree of the malware redirection chain), as well as the malware executables themselves must be frequently updated in order to avoid URL blocklists and anti-virus detections. In order for this attack to be successfully accomplished, the victim has to certainly enter the network. Figure 3 shows how the malware delivery chain might change between two visits to the same MDN.

A point that should be considered is that not all MDNs are similar. They may vary in different aspects such as the number of landing pages, type of malware repository and updating the malware repository. Additional information can be found in Section 3, MDNs Structure. An example of malware distribution network, is Fake Anti-Virus malware distribution networks. Fake AV attacks attempt to convince the users that their computer systems are infected and offer a free download to scan for malware. Fake AVs pretend to scan computers and claim to find infected files(files which may not even exist or be compatible with the computer's OS). Users are forced to register the Fake AV program for a fee in order to make the fake warnings disappears. To increase the injury of the

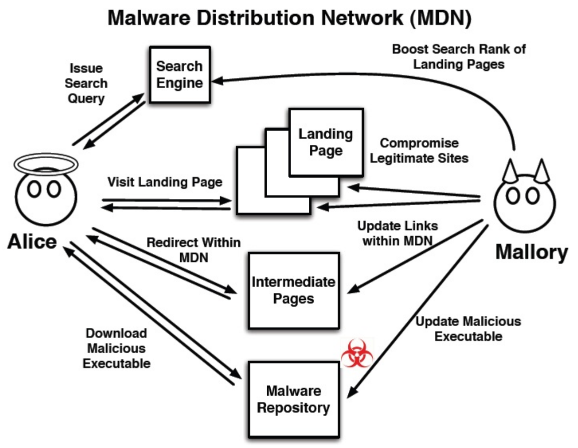
Malware Distribution Network (MDN)
Issue Search Query
Search Engine
Boost Search Rank of Landing Pages
Landing Page
Compromise Legitimate Sites
Visit Landing Page
Alice
Mallory
Redirect Within MDN
Update Links within MDN
Intermediate Pages
Download Malicious Executable
Update Malicious Executable
Malware Repository

Fig. 1. Attack scenario in a Malware Distribution Network (MDN). Alice, an average Internet user, is somehow redirected to the core of MDNs, the malware repository, from the landing pages. Mallory, the malware distributor, is actively maintaining the MDN [6].

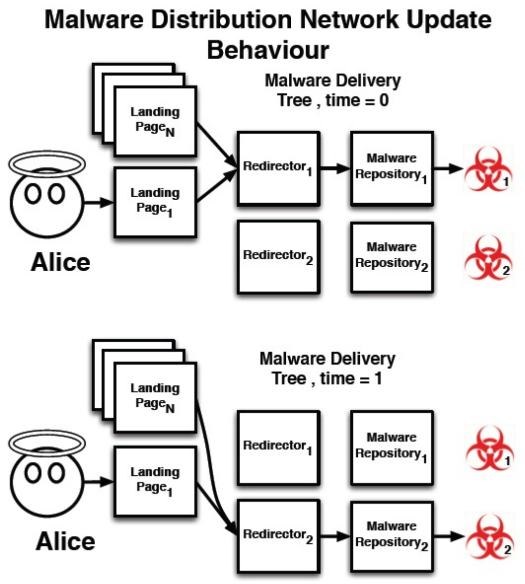

Fig. 2. When Alice, an average Internet user, visits a landing page at time 0, she is redirected through a specific set of servers to a malware repsitory hosting a malicious executable. If Alice visits the same landing page at time 1 she is directed along a different set of servers to a different repository hosting a different malicious executable.[1]

attack, Fake AVs are aggregated with other malware[8].

III. MDNs Structure

As mentioned before, not all the MDNs are exactly the same as each other, differentiating in distinct levels of structure. The main goal of this section is to provide an overview of what the structure of the malware distribution networks is by discussing about each level and also information about the roles played in MDNs. Therefore, the structure of a malware distribution network is separated from the people that play a role in the MDNs.

A. Structure

In Fig.1, an image is displayed indicating an abstract view of the malware distribution networks. As can be seen, all malware distribution networks rely extremely on the landing or intermediate pages and the malware repository. Both of these entities are very important for the attack and if one is unsuccessful in accomplishing its goal, the whole attack will not take place. For explicit understanding of these two entities, they will be discussed separately.

1- Landing Pages - Intermediate Page

Landing pages are an interface between the user and the target of this attack, the malware repository. The click traffic(when a user's web browser is directed to into a network) will be achieved through the control of these landing pages[1]. With the use of them the adversary can easily take advantage of the exploitation or simply perform a social engineering scam. Landing pages are basically pages modified by the adversary. He inserts the code necessary to carry out his intentions without allowing the main pages to learn of this modification.
Originally, in order to inject the specific code, the adversary or in better words, the malware distributor has two options: to create or to compromise. The definition of creating code is very obvious, the attacker simply creates code(content) that persuades his goals. Writing the code can be either done by himself or he can choose one out of many software that are available and assist him quickly in generating the content that ranks high in web searches[9]. Either way, this option is both very time consuming specially if he is willing to write it himself and also expensive, especially if he is using a software. More important than them, is the fact that search engines are becoming more tuned to detect and punish auto-generated pages[10]. Another common way used for this option is the deployment of Search Engine Optimization(SEO) kits to automatically generate popular content and cross-link their content with compromised content for further increase traffic to the compromised site[11]. For the compromise option, which has become more common than the previous option, the malware distributor takes advantage of the reputation and popularity of a legitimate site. There are different methods in injecting code inside a legitimate site.
There are multiple ways which can aid the adversary

to compromise the legitimate site, such as commercial or open source HTTP servers available and the most common nowadays are Apache and Microsoft IIS[12]. In addition to these software, there are code and content management platforms such as Drupal[13] and WordPress[14]. Any of the components used, might contain bugs, and since they might be vulnerabilities, one can easily exploit these bugs to persuade his own purpose. A malware distributor can also benefit from using exploit kits available today which contain exploits that target specific versions of vulnerable servers[15]. Whenever a server is found, this kit allows the adversary to add or modify the content on the legitimate site by facilitating the execution of the exploit simple.

As mentioned before, the malware distributor must compromise the web hosting software to inject his code, yet full compromise is not necessary. Many sites rely on the content submitted by the users and the advertising networks like forums and blogs. Some of the sites store these data in a Database(DB) and generate the web markup dynamically by querying the DB[1]. The malware distributor injects the ideal code for redirection into the content DB by exploiting one of many known SQL injection vulnerabilities[16]. This kind of injection is indeed very hard for the administrators to detect since it cannot be identified when inspecting the static content of the web site.

Advertising typically is the action of displaying content which is basically controlled by a third-party. On the web, most of the advertisements are carried out by dedicated advertising companies and organizations that provide very small pieces of Javascript to web masters in order to insert into their web pages. Although web master cannot directly monitor the ads themselves, they simply trust the advertisers to show non-malicious content. This is an ideal assumption when advertisers rely on the business from the web masters. Malicious content could very easily harm the advertiser's reputation, that leads to removing ads by web masters. The act of renting out the advertising space to advertisers brings complications in the trust relationship by asking for transitive trust. In other words, the web master needs to trust the ads provided not only by the first advertiser but also from the company that might be trusted by the first advertiser. With all complications, the adversary still gets around them and knows how to inject his code by using Javascript and iFrame into his ads.

The purpose of the injected code is to take either action: Redirection or Embedding. Redirection is the act of redirecting the clients web browser to another site and embedding is the act of downloading content that are comprised of elements within it such as *iframe* or *img*[1].

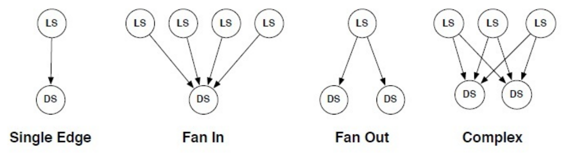

Fig. 3. Simplified MDN configuration. LS resembles the landing sites and DS represents the malware distributor or malware repository.(From[18])

Either way, these two actions directly or indirectly guide the user towards the malware repository that will achieve in the attack.

2- Repository

As mentioned before, the malware repository must avoid being detected by URL blocklists and anti-virus detections and in order to achieve this goal frequent updates of their URL is necessary. The update frequency of the malicious executable distributed by the malware repository varies and basically depends on the MDN. A repository can be classified into three categories based on the sample update behaviours:

Single Sample Repository: A repository that does not update the malicious executable for the lifetime of the repository[1].

Multiple Sample Repository: A repository that performs updates to the malicious executable over time, but is not generating the samples for each request[1].

Polymorphic Repository: A repository that produces a unique malicious executable for every download request[17].

However, several combinations of MDN components have been identified in [18]. This is shown in Fig. 3. In this image, the multiple MDN configurations is displayed, showing the relationship between the landing sites(LS) and the malware repository(MR). However, the intermediate sites are disregarded if any exists.

Once at the malware repository, two techniques can be employed to deliver the malware and are divided into two categories[16]: Social Engineering and Vulnerability Exploition. In social engineering attacks the adversary tries to trick the victim into willfully download a virus and in our case, the malicious binary. The victim believes it is a legitimate software program. At time of writing Fake Anti-Virus scareware is by far the most common and effective social engineering trick used by malware authors[8]. In Fake Anti-Virus, a user is simply redirected to a web server that displays contents resembling the Windows *My Computer page* and informs the user that their computer is infected and he is required to download a software to remove all of the viruses. The vulnerability that is mostly chosen is a memory corruption vulnerability that allows arbitrary code execution[1]. The inserted code causes the browser to download and execute the malicious binary infecting the victim's computer system.

B. Roles

One question that might appear to mind, is that: "Is the person responsible for the malicious binary the same person who compromises the web servers?". Based on the research done, it is seen that different people might participate and each have a one or multiple roles. However, one individual responsible for the executable binary is not necessarily the same person compromising the legitimate sites. There might be affiliates which may receive commission for guiding the victim in redirection or downloading the malicious binary. Fig. 4 resembles a type of malware distribution network. In this scenario, the adversaries or Pay Per Install(PPI) clients wish to persuade the victim into downloading the software they have provided. In other words, clients are entities that want to install a program on a number of target hosts. However, they might not be able to directly reach this goal, thus they will use PPI services. The PPI clients pay the PPI provider to aid them in road of downloading and installing their program on the target host's system. The PPI provider uses a program called the *downloader* that retrieves and runs the client's executables upon installation[18]. The PPI provider may conduct the installation itself or use third parties, the affiliates. When a provider has an affiliate, he acts as the man in the middle, while the affiliate simply facilitates the distribution of the installation.
Based on this scenario, it can be concluded that their might be several parties interacting with one another in the malware distribution network and all benefit from downloading and installing that malicious binary. The real adversary, can benefit much by using other parties such as not requiring the necessary knowledge and experience in distributing the malicious binary and he will not be wasting a lot of time for this step and thus, the executable binary can be downloaded and distributed in very little time.
Affiliate web marketing has become very common today. The affiliate networks have focused on the promotion of illegal products and have become the multi-million dollar "industry". It has also become the main force behind recent explosion in malware, web site infections, email spam and general web pollution[9]. The most dangerous side of the affiliate business such as scareware are being forced to close or go underground which impacts their operational costs[9]. Although, this is good news yet we know that it will not completely eliminate the illegal practices in Internet.

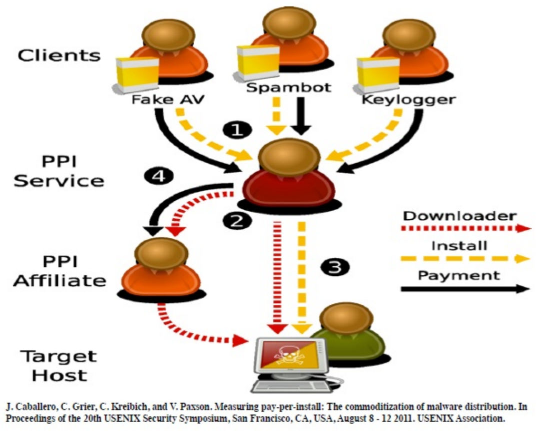
Clients
Fake AV
Spambot
Keylogger
1
PPI Service
4
2
3
PPI Affiliate
Target Host
Downloader
Install
Payment
J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In Proceedings of the 20th USENIX Security Symposium, San Francisco, CA, USA, August 8 - 12 2011. USENIX Association.

Fig. 4. The typical transactions in the Pay Per Install market. PPI clients provide software they want to have installed, and pay a PPI service to distribute their software(1). The PPI service conducts downloader infections itself or employs affiliates that install the PPI's downloader on victim machine(2). The PPI service pushes out the client's executables(3). Affiliates receive commission for any successful installations they facilitated(4). Image and caption from [18].

IV. MDNs Features

In this section we will provide a total overview of the MDNs features since they are very crucial in creating an MDN and each of them must be considered carefully. For this purpose, each feature will be discussed separately.

A. Dynamic Nature

One of the most important features of the malware distribution network is the dynamic nature it obtains. The dynamic nature of the MDNs is very much related to the frequent updates of the landing sites and the malicous binary. However, two factors contribute to this aspect:
1. URL Blocklists and Anti-Virus Detection: An MDN must constantly change the domains of their URL in order to avoid URL blocklists and takedown requests. Also, the executables are constantly modified to evade the AV detection[1].
2. Pricing Differentials: A singe MDN can serve as an affiliate for multiple PPI providers(see Fig. 4 for list of PPI roles). Each provider has a different price payment scheme, and certain traffic will be more valuable to one PPI provider[1].
For this reason, several entities of the malware

distributor network requires change over time:
1. Landing Pages
The adversary tries to keep the landing pages active as long as it is possible. Yet as time goes by and the detection techniques and tools improve, the landing pages can be very easily identified. In order to reduce the chance of being detected by them, they should also be modified or simply updated. In addition, the adversary might want to use new compromised sites, and thus, he simply modifies the existing landing pages to redirect the user to this new site.
2. Malware Repository Since the malware repository is the very most important entity in the attack as it plays as the heart of the network, thus, it requires full attention. Without the malware repository, the attack will not be successful and the malicious binary will not be downloaded by the victim. For this reason, the malware repository also requires frequent updates, especially to avoid the URL blocklists and detection. Therefore, the full URL or the domain of the repository needs to be rotated constantly.
Fake AV MDNs, also update the malware repositories and malicious payloads on a frequent basis and there is a strong fan in factor from the landing pages to the heart of the attack, the repository. One question that appears is: "What is the basis for updating?" This question cannot be easily answered since it depends on many things. The adversary should consider many things for reach a conclusion of the time to become as the baseline. One of them is the techniques used to detect the landing sites and the malware repository such as the blocklist detection techniques.

B. Size

One other important aspect of the malware distribution networks is the size of it. This is very much related to the landing sites and the intermediate sites, generally the delivery chain or delivery tree. A malware delivery tree consists of the landing sites as the leaf node and all nodes that the browser visits until it contacts the malware distribution site which is the root of the tree[6]. As mentioned before, not all MDNs are the same in structure. Some might only have one landing site while others might inlcude several landing pages. In the case of several landing pages, these pages can redirect the web browser to a new compromised site and redirect it to another landing page. In one of the research done to learn of the size of malware distribution network, the results are: 45% of all detected malware distributions sites belong to networks that only have one landing site as the interface and about 40% of those networks, the malware is hosed directly on the landing site[6]. When there are several landing sites in a delivery chain, one thing that can be noticed is that they can also overlap. For example, the first landing site redirects the browser to a new site, and after several redirection, one can redirect the browser to the very first landing site. In Fig. 2, you can see that after the second visit, the web browser can be directed to another site. It should be noticed that downloading in the first visit is not always necessary.

C. Distribution of Malware binaries

In order to prevent detection, distribution of the malware binaries should also be considered by the attacker. If the malware repository always distributes only one unique malicious binary, it will be very simple for the blocklists and anti-virus tools to learn of this binary. Yet, if the adversary wants to prevent detection and still continue with the spread of his binary, it is considered to distribute multiple malicious binaries by one malware repository. Thus, the adversary will distribute the malware across different domains. In [6], approximately 42% of the distribution network sites delivered a single malware binary and the remaining distribution sites hosted multiple distinct binaries over their observation, with 3% of the servers hosting more than 100 binaries. In conclusion, employing multiple payloads reflect deliberate obfuscation attempts to evade detection[6].

D. Malware Hosting Infrastructure

In a malware distribution network, the malware repository can be hosted on a single IP address or on multiple IP addresses. If an attacker is very naive or inexperienced, he might use an IP address to host his malware repository that has another malware distribution network already hosted by it. If so, the chance of detecting the malware repository will become very high, if one malware distribution network is found, then it will be considered that IP address can contain other malware distribution networks as well. However, if each malware repository is hosted by a single IP address, then it will probably take more time to learn that malware distribution network. In [6], 90% of the cases identified, each site was hosted on a single IP address and the remaining 10% sites were hosted on IP addresses that host multiple malware distribution sites.

E. Countermeasures

The study of malware on the web is complicated and difficult due to the several countermeasures the malware distributor employs. These include anti-crawler content and blacklisting which are discussed separated here.

1) Anti-Crawler Content: Any technique that can be used towards the goal of complicating the task of honey clients is considered to be an anti-crawling technique. There are many content-based techniques that should be thought about and the most general of all is the use of obfuscated JavaScript.

This technique is widely applied to increase the complication of content interpretation. Simple JavaScript functions (e.g., 'Window.location') are resided in multiple layers of encoding in order to prevent the simple crawlers from interpreting their intent[1]. Both sides, the crawler and the adversary rapidly update and improve their techniques so they cannot be defeated by the other competitor. In addition, the techniques used to prevent the crawlers are evolving with a high speed in order to include multiple cooperating scripts, use of the DOM content in decryption loops, checking for the presence or absence of cookies and other tricks.

2) Blacklisting: There is little published information on the use of blacklisting by malicious networks[18]-[21], yet it is generally believed in the security community that certain malware networks engage in the act of blacklisting. Blacklisting the context of malicious networks can be broken into two simple steps: Identification of Honey Clients and Altering responses to Honey Clients.

1. Identifying the Honey Clients

In order to collect instances and samples of malicious web content and binaries, security researchers have deployed custom HTTP clients that automatically gathers content for analysis. These systems are referred to *client-side honeypots* or *honey clients*. The honey client is responsible for retrieving the malicious content and simply monitor the system's state after downloading and installing the malicious binary.

Identifying the honey clients might not require very much effort since they visit the web servers repeatedly and they also visit many web servers controlled by the same organization. The repetition of the downloading operation produces a track of the honey client which becomes suspicious to the adversary and thus, be easily identified. The identifiable features of the honey client are: the IP address, the TCP/IP characteristics, HTTP content, the frequency of downloading and the number of downloads.

2. Altering Responses to Honey Clients

Once a honey client has been identified by the malware distributor, he has a few options to choose from:

HTTP 500 In this scenario, the malware network simply refuse to deliver the content to the honey client.

Benign Content In this case, the malware network will deliver the contents of a benign web site or redirect the honey client towards the benign web site[11].

Old Content In this scenario the malware network will continue to present the honey client with the content, yet the content is an old version of the malware.

Tarpit The use of tarpitting was originally used to reduce the effects of network worms[22]. Here, the malware distribution network intentionally holds the TCP/IP connection open as long as it can be, in order to send the content in a very low rate or by not delivering the content at all.

For clarification, a blocklist is a list of resources to block while blacklist is the act of using a blocklist to change a response. In other words, a blacklist is a control mechanism that will allow all except for the members of the blacklist to have access to that web site. Many organizations keep a blacklist of their own based on different kind of attacks that were employed like email spam, cloaking, software copying and many more. One of the web spam techniques named cloaking was very much interesting for the search engine crawlers in year 2005. Wu and Davison[23],[24] conducted many studies regarding semantic cloaking. They impersonate regulate Internet users as a baseline in addition to the automated crawlers by simply varying the user agent. Nui et al.[25] performed a similar study, focusing on the problem of forum-based spamming as black SEO technique. They detected a new kind of cloaking known as click through cloaking that separates the user from the crawler based on the value of the HTTP referrer. By changing the HTTP referrer, the cloak would be triggered and could easily use the presence of the cloak as a span indicator to aid in the URL categorization.

F. Attractiveness and Persuasion

As mentioned before, the adversary can create a new content or simply compromise a legitimate site for his purpose. One of the really important aspects of the landing pages is to easily convince the user that it is useful for him and without that, no other solution is found. If the attacker wants to create his own content, the appearance, structure and content should be very similar to high ranked web sites in order to attract the user and not to allow the user to learn it is an intermediate page. In order to reach this goal, a lot of time and experience is necessary and if the attacker wills to distribute his malicious binary, it will not be profitable for him. Thus, he will certainly benefit from using legitimate sites and there is nothing to worry about.

V. Adversarial Information Retrieval Systems

With the distribution of malware over the web, information security researchers regularly download the web content and system executables from the Internet looking for threats. Studying malware distribution networks is adversarial in nature: behind the networks and the binaries is a group of humans that change their behaviour to counter the efforts of security researchers[1]. The systems that are responsible for identifying the malware distribution networks and harvest their content are called the *Adversarial Information Retrieval Systems.*
In this section, the challenges for these systems are discussed, followed by *why* evaluation and re-evaluation of the URLs are necessary, *what* URLs should be evaluated and finally, *how* to evaluate these URLs.

A. Challenges

One of the challenges for retrieving information regarding the malware distribution networks is the dynamic nature of the MDNs which is explained in previous section. These networks are comprised of different levels of servers where each plays a distinct role in the malware delivery chain. The links between the servers and the content served by each server which includes the malicious binary are frequently updated to evade any efforts of the security researchers. The second challenge is the adversarial counter measure of blacklisting, where an MDN identifies the components of a security system and alters its behaviour while studied by the system[1]. These challenges lead to two main problems:
Scalability: There are a lot of URLs to evaluate. The number of new URLs are extremely large that the retrieval systems of adversarial information are always working, creating a backlog of URLs that are waiting to be or have been already evaluated. Any new suspicious URLs are discovered and pre-evaluated malicious URLs need to be re-evaluated for content updates due to the dynamic nature of the malware distribution network.
Accuracy of Content: URL evaluation can be tainted by blacklisting. Blacklisting occurs when the retrieval system revisits the same servers repeatedly and they also visit many different servers controlled by the same organization. The repetitive downloading leaves patterns in HTTP server logs that can be identified by the malicious adversaries. If a client is identified, the MDN will alter its content which reduces the performance of the retrieval systems.

B. Why evaluate URLs?

The evaluation of URLs is very important for the retrieval system in order to support multiple objectives which are as below:
1. Collect any new malicious binary to update detection strategies.
As mentioned before, the landing pages, the malware repository and the malicious binary needs to be updated. Most important of all, is the malicious binaries that are regularly updated in order to avoid anti-virus detections. In some cases, new URLs are used to host the updated binary, and in other cases the same URL is used to host a new binary. If the latter case is used, a re-evaluation of the malicious URLs in order to harvest new binaries.
2. Collect new malicious web content to update detection strategies.
Many anti-virus products include detection of malicious web content in addition to the malicious binary content. This presents another layer of protection for all users by stopping the attack when a malicious web content is identified before the user's web browser is compromised and forced him to download the malicious binary. This content, like the malicious binaries, is updated to counter anti-virus detection strategies[26]. Thus, it is essential to collect and analyse new malicious web content.
3. Maintain a blocklist of malicious URLs.
URL blocklists provides the user with another layer of protection against the malware binary. If an update of malicious content by the malware distributor defeats the anti-virus detection, yet the malicious content is hosted on a blocklist domain or IP, then the user will be protected by the anti-virus. Populating and keeping records of the blocklists of network elements such as domains, name servers, IPs, DNS patterns, requires collection and examination of the content from those respective network components.
4. Avoid false positives.
Classifying the benign content of a web site as malicious content is referred to *False Positive*. The risk of indicating any URL as malicious content is seen in all systems. The use of blocklists results to a new false positive concern: the listing of a legitimate but compromised site for longer than the necessary time. The initial decision to add the legitimate content to the list of compromised ones is contentious. Regardless of this contention side, when a legitimate site has been added to the blocklist, this site will be monitored for any signs or indications of clean up so that the site could be removed from the blocklist afterwards.

C. What URLs should be evaluated?

Assuming there is unlimited source for evaluation from the protection point of view, the best strategy is to crawl everything on the Internet. For all but the largest technology organizations and firms, this is however not feasible. With this fact, the question that appears to mind is: “What URLs should an anti-virus firm harvest?”. To answer this question, the usual sources of distrustful URLs that are evaluated in security labs are discussed below:

Following Trending Search Terms
The use of Search Engine Optimization (SEO) poisoning in order to generate traffic to the malicious network is a very ordinary technique among web sites[6]. Search engine optimization poisoning is an attack towards the search engines which the goal is to boost their rankings among major the search engines, such as Google and Bing by using XSS or cross server scripting. The attackers inject common search terms and an iframe script designed to send victims to other sites hosting malicious code. The search term and iframe redirect get cached in search engines. Attackers can also upload and implement scripts basically in PhP, JavaScript languages on the clients web browser which become infectious and the results to a threat. It is possible to identify trending search terms and then execute searches for these terms. The results of these searches, leads to landing pages of the malware distribution network.

Searching for Vulnerable Strings
Vulnerable web hosting platforms can be easily identified by the footers containing the version string. Locating for the footers, guides to the compromised site and these sites can be used as anchors in subsequent searches (using search operators such as *inurl*[27]), yielding potential landing pages [1].

Searching for known Kit Patterns
Many exploit kits will generate and produce URLs that have identical patterns. For this regard, search modifiers can be easily used to identify these URL matching the patterns which also results to the compromised pages or the landing pages.

Static and Dynamic Analysis of the Executable Files
URLs can be retrieved by learning the static strings included in the malicious binary. In order to analyse the dynamics of the executable binary, the runtime behaviour of a malicious binary should be monitored and observed.

Product Feedback
A great real-time source of malicious URLs come from the modern security products itself that report the detection of the malicious binary back to the vendor of the security firm. The feedback strategy is not only used for this purpose, yet other systems have benefited from this strategy as well. However, many feedback strategies can be applied that suites the purpose well enough. In [1], the researcher uses feedback from web content filters that report URLs classified as malicious based on content inspection.

Strategic Relations
The are and can be symbiotic relationships where a non-security firm with unique visibility (e.g., search engines, social networks, telecommunication firms) use augmented security products to identify malicious networks affecting the partner[1]. In exchange, the partner simply delivers intelligent threats that were collected when using the security product.

Cooperative Industry Exchanges In industry, it has become common to share any information regarding a malicious binary that has been detected such as the URLs and the malicious binary itself between the anti-virus labs and security researchers. The information are highly important and must be typically verified upon occurring one.

D. How to evaluate the URLs?

In order to gather the samples of the landing pages content and binaries, security researchers have developed custom HTTP clients that automatically collects these information for analysis. These systems, are called honey clients. Honey client’s duty is to retrieve and observe the system’s state after downloading the malicious binary. Honey clients may vary in the means of their implementation and features. Similar to honeypots, honey clients are classified intro high and low interaction varieties[28]. A honeypot is a trap that is set to detect and counteract attempts at unauthorized use of information systems. A high interaction honeypot uses the actual vulnerable software (in this case the browser and the software), whereas a low interaction honeypot will mimic the vulnerable software.
Low Interaction Honey Clients (e.g., GNU Wget[29], Heterix[30]) implement the HTTP protocol and have the ability to download the content and follow the basic HTTP redirections. Yet, they do not interpret the downloaded content in the same manner as the web browser. In other words, JavaScript or HTML redirections will not be followed and the embedded content cannot be retrieved. An advantage of a low interaction honey client is that they are light weight in terms of resource use and the state of the downloader does not require to be reset between fetch attempts (as in the case of high interaction honey clients)[1]. Thus, a low interaction honey client can download at a higher rate than the high interaction

honey client.

High Interaction Honey Clients, such as Microsoft's HoneyMonkey[31], use an automated web browser in a sand-box environment to conduct the URL evaluation. The operating system is monitored for any unexpected state changes, like file downloads, that indicate the system has been infected with the malicious binary. The upside of high interaction honey clients is the full fidelity of the environment[1], a real browser is used to apply the sequence of the HTTP requests, and all of the interpretations is done by the web browser. However, some honey clients control the post infectious behaviour of the malware, resulting to provide more information and confidence in any of the subsequent malicious classification. One of the disadvantages of the high interaction honey clients is the design and maintenance which is very complex, time consuming and expensive due to the amount of components involved and the time necessary to restore the state between the download attempts. The high interaction honey client can be compromised during the download attempt, thus it will be required to be reset and restored to the previous state where is was not compromised. Without the restoration, the results of a download can taint other downloads.

In addition to the low and high interaction honey client, a third kind is also discussed here: The Medium Interaction Honey Client[1]. Medium interaction honey clients add additional functionality to interpret the web page and mimic the browser's certain features in order to identify the malicious behaviour. For example jsunpack[32], emulated ActiveX plugins which enables the software to detect exploits targeting specific plugins. This additional feature of the medium interaction honey clients increases the cost compared to low interaction honey clients, yet they do not require to be reset to the previous state and thus, less time is necessary compared to high interaction honey clients.

An important question that the adversarial information retrieval systems must answer before their design, is: "Out of these kind of honey clients, which type is more suitable to be used to download a URL?". To answer this question, the three types of honey clients will be compared with one another. High interaction honey clients can handle a broader range of URLs than the low interaction honey clients, yet they are very slower and more expensive. If the resources are all available, it is preferred to use the high interaction honey clients but when not all of the resources are available, a combination of the high and low interaction honey clients can improve the performance and thus, leads to better results[33].

However, a high interaction honey client is necessary when a MDN uses a browser or a plugin exploit, and also when the network uses crawler evasion techniques which simply prevent the study by low interaction honey clients. The URL stream that are received by the external sources usually do not have enough provenance to determine the expected content a *priori*[1]. By using the low interaction honey client one can identify URLs that need analysis by the high interaction honey client. In other words, it is recommended to use low interaction honey clients to first identify the URLs and for better learning, use high interaction honey clients.

VI. Identifying the MDNs

The studies of malware distribution network has become very common research area in the passed few years. In order to identify malware distribution networks in general and the landing sites, malware repository, affiliates and other components, two approach for this purpose are carried out: The *top down approach* and the *bottom up approach.* In Fig. 5 , the different methods used to detect *drive-by downloads* can be seen. Drive-by download is an attack where a malware is downloaded to the victim's computer when visiting a web site without his or hers consent. Since drive-by download is very much related to the malware distribution network subject, it is preferred to provide additional information regarding drive-by downloads here before discussing about the two approaches used for identifying the malware distributor of these malware in the drive-by download attacks.

The structure of drive-by download attack is very much similar to the malware distribution network. Multiple steps should be taken so the drive-by download attack can be successful. The first step is to embedded the malicious content into a web site, which is firstly rendered by the web browser and does not exploit the browser directly. Instead, it redirects the browser to other pages which some might be in the same server of the previous page or simply be in a different server. After visiting one or more redirection servers, the browser will eventually lead to a malicious redirection server that directly redirects the user to the downloadable malware. The malicious redirection server, can be used to manage the attacks and decide the exploit server to use which has the best matching set of exploits[34]. A set of different drive-by downloads can be managed by the same adversary for a specific goal such as forming a malware distribution network. For a better understanding, Fig. 6 displays how this attack is applied.

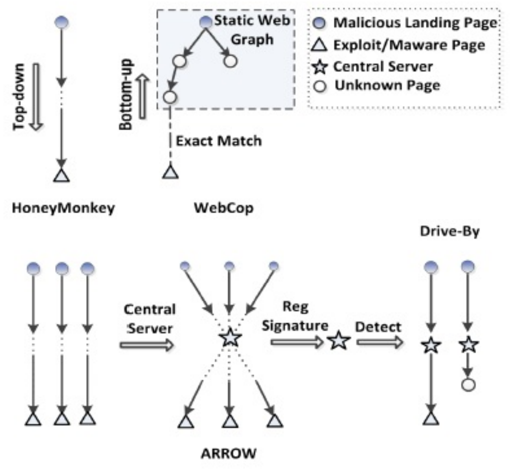

Fig. 5. Different methods for detecting Drive-by-Downloads. Image from [34]

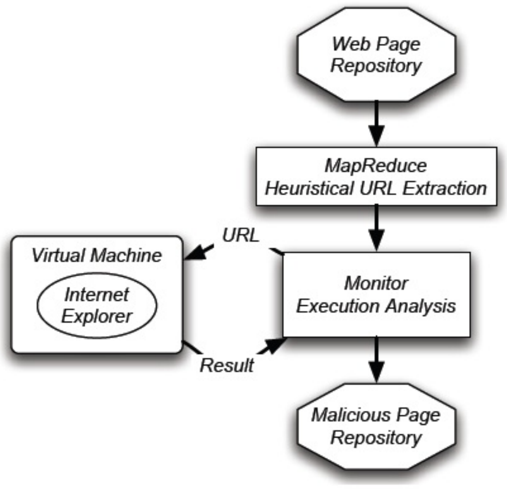
Web Page Repository
MapReduce Heuristical URL Extraction
URL
Virtual Machine
Internet Explorer
Monitor Execution Analysis
Result
Malicious Page Repository

Fig. 7. This diagram shows an overview of the detection architecture. The candidate URLs are heuristically selected and determined via an execution in a virtual machine if the URL behaves maliciously. Image and caption from [35].

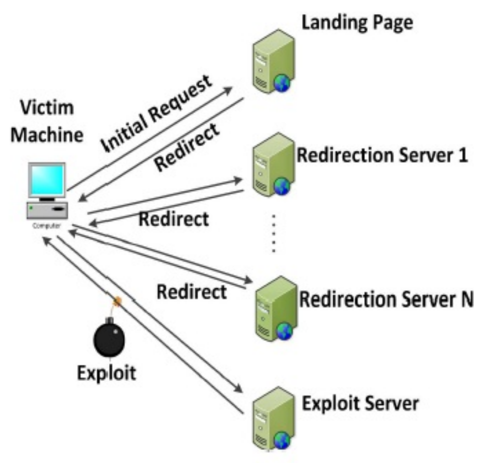

Fig. 6. Different methods for detecting Drive-by-Downloads. Image from [34]

A. Top Down Approach

In the top down approach, the goal is to detect the malware repository and the malicious binary through detecting the landing pages first. This means, the suspicious URLs (landing pages) are traversed in the delivery chain until the malware is detected and in some cases, it is executed by the honey client of the tool used.
As can be seen in Fig. 7, this is a representative of the top down approach. To provide an idea of how the top down approach works, this image will be explained. The description of their method provided here is based on their studies.

There are three phases for analysing the results: identification of candidate URLs, in-depth verification of URLs and aggregation of malicious URLs into site level ratings. For the first phase, *MapReduce* is employed to process the crawled web pages in order to learn of exploits. MapReduce is a programming model that operates in two stages: the *Map* stage takes a sequence of key-value pairs as input and produces a sequence of intermediate key-value pairs as output. The *Reduce* stage merges all intermediate values associated with the same intermediate key and outputs the final sequence of key-valued pairs[35]. The use of MapReduce is to prune several billion URLs into a few million. To verify that a URL is really the cause of a web browser exploit, a web browser and in this case Internet Explorer is used in a virtual machine. The browser ill navigate to each candidate URL. All of the HTTP are recorded as well as the state changes to the virtual machine such as registry and file system changes. For each URL, the analysis is scored by assigning individual scores to each recorded component. If new processes are running by the machine when it is visiting a web page, it can be concluded that a drive-by download attack has occurred.
In order to learn which search results are harmful, they will be tagged as harmful by aggregating the URL analysis on a site basis. If by any change, the majority of the URLs on a site are malicious, the whole site or the path of the site might be tagged as harmful.

B. Bottom Up Approach

In the bottom up approach, data from many HTTP traces are aggregated in an offline process to discover a larger percentage of the malware distribution network components. As can be seen in Fig. 5, one of the detection methods is to use the bottom up approaches. One novel method that is used by J. Zhang et al and is called *Arrow* is discussed here. The method is explained based on their studies.

This method bootstraps from the drive-by download samples detected using existing methods, where they aggregate drive-by download samples into MDNs based on the malware information or the URL of the exploit server. For any MDN, the central servers are identified, if they exist. The central server is the common server that is shared by a large percentage of the drive-by download samples in the same MDN. A central server typically provides important information to achieve in this attack and is not necessarily the server used to exploit attempts or malware distribution. The central server can also be a legitimate server where specific information is retrieved in order to calculate the location of the exploit servers dynamically[34].

The next step is to generate signatures in the form of regular expressions based on the URLs for the central servers. These signatures can be distributed to a search engine or a browser in order to detect the drive-by downloads. The lower half of Fig. 5, displays their method.

The reason they have used signatures is to increase the detection coverage in three ways:

1. If a drive-by download attempt reaches the central server without hitting the servers for exploit attempts or malware distribution, their signature can still detect the attack.

2. For a drive-by download attempt, if only one URL request is sent to the central server but no malicious web page content is returned, the signature can again detect it because the signature is independent of the web page content.

3. The signatures are in the form of regular expressions which can capture the structure patterns of a central server's URL and therefore outperform exact string matching used by WebCop.

WebCop is another bottom up method used, with the same goal of identifying the malware repository or distributor first. Fig. 8 illustrates the architecture of this method which is explained.

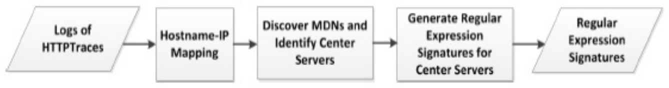
Logs of HTTPTraces
Hostname-IP Mapping
Discover MDNs and Identify Center Servers
Generate Regular Expression Signatures for Center Servers
Regular Expression Signatures

Fig. 8. The architecture for this method. Image from [35]

VII. Conclusion

Malware distribution networks has become a very common attack in the network society especially when people are interacting more with the World Wide Web. This attack exploits from the network structure used nowadays. The goal of malware distribution networks is to redirect the user to the heart of the attack which is the malware repository in order to download the malicious binary to the victim's computer system. In some cases such as the drive-by download the malicious binary is not downloaded directly by the user and in some cases specifically in malware distribution networks the executable binary is downloaded by the user willingly since the attacker or better to say the malware distributor has persuaded the user that the malicious binary is the software or program he is searching for. Fake Anti-Viruses are an example of malware distribution networks where, the user is tricked to download a software that is written by the adversary himself which will relieve the user from the warning that his computer has been infected by different malwares and in some cases he has to pay a fee to download that software.

Many different aspects of the malware distribution networks is discussed in this paper. The structure of a malware distribution network contains important components that are landing pages and the malware repository. The adversary should consider many different features before creating a malware distribution network such as the size of the malware distribution network, the distribution of the malicious binary, the hosting infrastructure and many more. The countermeasures are also explained. In addition, information regarding the adversarial information retrieval systems are also discussed. The challenges that they might encounter, how, why and what URLs they should evaluate are also mentioned in Section 5. In Section 6, information the two approaches to identify the malware distribution network, the top down approach and the bottom up approach with a description of each example is provided.

It can be concluded that malware distribution network has become the focus of many security researchers, since it is indeed very important due to the daily interactions of people with the Internet.

References

[1] K. Zeeuwen, “Optimizing re-evaluation of malware distribution networks,” *Master Thesis*, The University of British Columbia, Oct. 2011.

[2] CERT Advisory CA-2001-19 Code Red: Worm Exploiting Buffer Overflow In IIS Indexing Service DLL, 2001. [Online] http://www.cert.org/advisories/ CA-2001-19.html.

[3] CERT Advisory CA-2001-26 Nimda Worm, 2001. [Online] http://www.cert.org/advisories/CA-2001-26.html.

[4] CERT Advisory CA-2003-04 MS-SQL Server Worm, 2003. [Online] http://www.cert.org/advisories/CA-2003-04.html.

[5] CERT Advisory CA-2003-20 W32/Blaster Worm, 2003. [Online] http://www.cert.org/advisories/CA-2003-20.html.

[6] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose, “All your iFRAMEs point to us,” In *USENIX Security Symposium*, pp. 1-16, USENIX Association, 2008.

[7] European Computer Manufacturers Association. *ECMAScript Languate Specification* 3rd Edition, Dec 1999.

[8] M. R. Rajab, L. Ballard, P. Marvrommatic, N. Provos, X. Zhao. “The Nocebo effect on the web: An analysis of fake anti-virus distribution.” In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms and more.* LEET 10, pp 17-25, USENIX Association, Aprl. 2010.

[9] D. Samossseiko. “The partnerka- what is it, and why should you care?”, In *Virus Bulletiin* Virus Bulletin, 23-25, Sept. 2009.

[10] M. Cutts, Google search and search engine spam. [Online] http://googlebot.blogspot.com/2011/01/google-search-and-search-engine-spam.html, Jan. 2011.

[11] F. Howard, O. Komili. “Poisoned search results: How hackers have automated search engine poisoning attacks to distribute malware.”, *Sophos Technical Papers*, Mar. 2010.

[12] Netcraft web server survey. [Online] http://news.netcraft.com/archives/category/webserver-survey/, June 2011.

[13] Drupal. [Online] http://drupal.org/, Sept 2011.

[14] WordPress. [Online] http://wordpress.org/, Sept. 2011.

[15] J. John, F. Yu, Y. Xie, A. Krishnamurthy, M. Abadi, “Heat-seeking honeypots: design and experience”, In *Proceedings of the 20th Annual World Wide Web Conference(WWW)*, Hyderabad, India, pp. 207-216, Mar. 28 - Aprl. 1, 2011.

[16] MITRE, “CWE: 89: Improper Neutralization of Special Elements used in an SQL Command.”, [Online] http://cwe.mitre.org/data/definitions/89.html/, 2011

[17] R. Sherstobitoff. “Server-side polymorphism: Crime-ware as a service model(Caas).”, *Information Systems Security Association Journal*, 6(5), 2008.

[18] J. Caballero, C. Grier, C. Kreibich, V. Paxson, “Measuring pay-per-install: The commodization of malware distribution”, In *Proceedings of the 20th USENIX Security Symposium*, San Francisco, CA, USA, pp. 281-290, ACM, Aprl. 2010

[19] K. Zeeuwen, M. Ripeanu, K. Beznosov, “Improving Malicious URL Re-Evaluation Scheduling Through Empirical Study of the Malware Download Centers.”, In *WebQuality 2011*, ACM, Aprl. 28 2011

[20] M. Wood, “Turning scareware devious distribution tactics into practical protection mechanisms.”, [Online] http://nakedsecurity.sophos.com/2011/02/14/scareware-distribution-tactics-practical-protection-mechanisms/, Feb. 2011.

[21] J. Sobrier. “Fake AV vs. zscalar”, [Online] http://research.zscalar.com/2011/04/fake-av-vs-zscaler.html/, Aprl. 2011.

[22] Tarpit. [Online] http://en.wikipedia.org/wiki/Tarpit_(networking), Aprl. 2011.

[23] B. Wu, B. D. Davison, “Cloaking and redirection: A preliminary study.”, In *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web(AIRWeb)*, Chiba, Japan, May 2005.

[24] B. Wu, B. D. Davison, “Detecting semantic cloaking on the web”, In *Proceedings of the 15th Annual World Wide Web Conferences(WWW)*, Edinbourogh, Scotland, May 2006.

[25] Y. Niu, Y. Wang, H. Chen, M. Ma, F. Hsu, “A quantitative study of forum spamming using context-based analysis.”, In *Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, pp. 79-92, Mar. 2007.

[26] F. Howard, “Malware with your mocha? obfuscation and anti-emulation tricks in malicious javascript.”, *Sophos Technical Papers*, Sept. 2010.

[27] [Online] http://www.google.com/intl/gn/help/operators.html/, June 2011.

[28] N. Provos, T. Holz, “Virtual Honeypots From Botnet Tracking to Intrusion Detection.”, Addison-Wesley, 2008.

[29] GNU Wget. [Online] http://www.gnu.org/software/wget/, Sept. 2011.

[30] Heretrix. [Online] http://crawler.archive.org/, Sept.2011.

[31] Y. M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. T. King, “Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities,” In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pp. 35-49, 2006.

[32] jsunpack-n: a generic javascript unpacker. [Online] http://code.google.com/p/jsunpack-n/, Sept. 2011.

[33] D. Canali, M. Cova, G. Vigna, C. Kruegel. “Prophiler: A fast filter for the large-scale detection of malicious web pages.”, In *Proceeding of the 20th Annual World Wide Web Conferences (WWW)*, Hyderabad, India, pp. 197-206, Mar. 28- 1 Aprl.2011.

[34] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee., “ARROW: Generating signatures to detect drive-by downloads,” In *Proceedings of the 20th Annual World Wide Web Conference (WWW)*, pp. 187-196, Hyderabad, India, March 28 - Apr. 1, 2011.

[35] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, “The ghost in the browser analysis of web-based malware,” In *Proceeding of the first conference on First Workshop on Hot Topics in Understanding Botnets*, Berkeley, CA, USA, 2007. USENIX Association.