International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

VILEEAR: Detection of Drive by Download attack on Malicious Web Pages

Chirag R. Desai[1], Dr. Narendra M. Shekokar[2]

[1, 2] Mumbai University, D. J. Sanghvi College of Engineering, Vile Parle (West), Mumbai-56, India

Abstract: ***Internet is a platform which is mostly used to spread malicious software and viruses on the network. The drive by download (DBD) is most successful and popular attack invented by web site attacker till date. Drive by download attack enables the victims to click coded malicious links and the browser will be redirected to malicious web sites to exploit vulnerabilities, and also it installs software from internet that can be harmful for victim`s machine. Script code embedded in plug-in are commonly used by attacker to execute drive-by download attack which is capable to exploit victim`s system vulnerabilities. Thus it is most challenging and important to find solutions that will detect and mitigate DBD attack. To Detect Drive by download attack is biggest challenge since there is tremendous growth of number of pages hosted on the web. So it is tedious to find small portion of malware network from web and detect whether the web page is harmful or not. In this paper we have proposed an enhanced mechanism called VILEEAR to detect Drive-by download attack. This system will locate and thoroughly analyze the web pages in malware distribution network (MDN) to detect DBD attack. The system will work like a supervised learning process for finding malicious web page in an MDN and will keep on improving the process to detect DBD attack after each iteration.***

Keywords: Drive-by download, Malware Distribution network, web security, malicious landing page, VILEEAR.

1. Introduction

The world-wide web is the most successful platforms used to spread malware over computer network. The use of internet is increasing daily and it has provided large number of facilities to users such as online communication, e-commerce, e-banking, entertainment, social networking, etc. Thus, cyber-attacks too are increasing along with the growth of web services and web applications. Drive by download [1] is the most significant and popular attack on the Internet, since it can execute malicious code on victim`s machine without victim`s interaction. Drive-by download attack is executed when the attacker embeds malicious script code in the benign web pages and these pages are visited by the user. The encoded web pages are called as the landing page and are hosted on a compromised web server.

If such attack gets executed successfully, the embedded code will force the victim's computer to download and forcefully install malicious software, and attacker will gain complete control of victim`s machine. Drive-by download attack allows the attacker to execute software that can record keystrokes, steal login credentials, and leak sensitive information of the system. Such infected systems can be used as a botnet, which is a collection of infected hosts controlled by the attacker.

The attack scenario to execute drive-by download attack [2] is shown in Figure 1. First the attacker will compromise a genuine web server and uploads malicious script code. Whenever the compromised web site is visited by the victim, the victim's browser will download and execute embedded script code. The encoded script will exploit vulnerability of victim`s machine by installing browser plug-ins and will provide full control to the attacker.

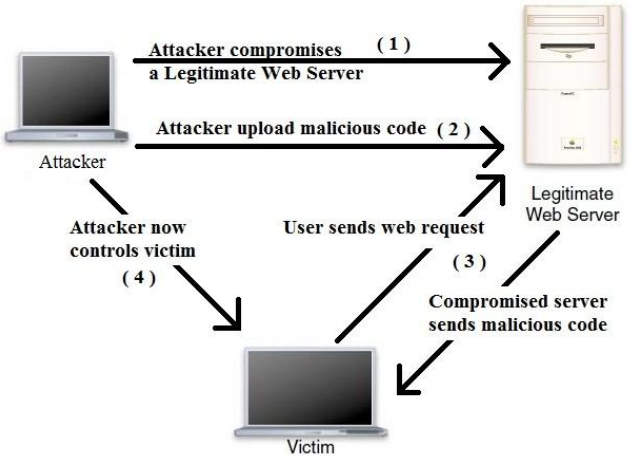

Figure 1: Attack scenario to launch Drive by download

Nowadays drive-by download attacker targets browser plug-ins [1]-[3]. Developers mostly neglect plug-in security when compared to browser security; thus they contain more security vulnerabilities. Plug-ins are mostly coded using unsecured languages C or C++, thus they exploits large set of vulnerabilities in the system. Plug-ins is executed in the browser and thus the attacker can get complete access of browsers and can easily penetrate into the victim`s machine. Hence, it is necessary to come up with more enhanced and foolproof solutions that will detect and prevent Drive by download attack from the users.

In this paper we have proposed an enhanced system VILEEAR that will locate the malware distribution network (MDN) and will thoroughly analyze web page together. Initially, this system will used Arrow [4] mechanism that discovers central server in Malware Distribution Network (MDNs) and leads to malicious landing pages. Using the supportive knowledge provided by Arrow method we will further identify and analyze the malicious content on landing pages using multiclass feature selection of rule-based

approach [5] and will give suspicious web pages as output. These web pages will act as initial ear page to a system called VILEEAR. These pages will help our system to query the search engine and identify more relevant malicious pages which are similar to the initial ear. This result will be given back to Arrow system in form of optimized input and complete system will work like supervised learning process and will keep on improving the process of detecting Drive by Download attack on each iteration.

The paper is designed as follows. We have elaborated in section II the related work on detection of Drive-by download attack. Our enhanced proposed detection system is elaborated in section III. We concluded the paper in section IV with optimized way to defend drive-by attack.

2. Related Work

To investigate and find out Drive-by download attack different methods are proposed by the researchers. The researchers have suggested various approaches which include locating, analyzing, detecting and classifying the web servers that hosts malicious web pages with embedded harmful scripts. The existing approach in which drive-by download attack are categorized is top-down approach and bottom-up approach.

The top-down approach uses a crawler and scanner based architecture. The crawler uses forward direction to traverse the dynamic web graphs and collects the URLs, while simultaneously scanner will identify drive-by download attack. The scanner works like a honeyclient which uses signature [6] and anomaly detection [7] methods.

Wang et al. [8] has proposed top-down approach called HoneyMonkey which is a browser based high-interaction honeyclient. It is fully unpatched system, in which the controller executes a program called as “monkey” which browses previously scheduled Web sites and waits for a given time to check for possible intrusions. If an attack is detected, the Web site is again visited with the next machine in the pipeline and if all chained patched system is exploited, and then the Web site is stated to have zero-day vulnerabilities. The HoneyMonkey system analysis is done by using black-box approach but the performance time to detect the exploit is very high.

Seifert et al. [7]-[9] has proposed another top-down approach called Capture-HPC which is also a high-interaction honeyclient. Capture system contains capture client and a capture server. The honeypot is hosted by client on a virtual machine and the server co-ordinates and controls the clients to provide dynamic execution of the web page content [8]-[9]-[10].

Provos et al. [11] also has used high-interaction client honeypots to measure drive-by downloads. Nazario [12] suggested PhoneyC method which is light weighted low-interaction client honeypot that analyze the web page content to find drive-by download exploits. Cova et al. [13] has suggested a mechanism that will locate the JavaScript embedded code in web pages for DBD detection. Recently, Lu et al. [14] suggested using a detector mechanism that manages user action and the downloading events to identify drive-by download attack.

These approaches have shown promising results but due to lack of a malicious content response from a drive-by download attack it is ineffective and has introduced a large number of false negatives outcome.

To overcome the limitations introduced by classifying and analyzing webpage content, Stokes et al. [15] has suggested WebCop method which is bottom-up method. It finds the web pages that host malicious web content. WebCop uses backward direction to extract the links of the web graph and finds the malicious landing pages for malware distribution network. WebCop uses a perfect match to discover the malware web sites in the web graph, thus it easily introduces false negatives. Web graph uses hyperlinks which are rigid and restricts the detection of WebCop. Therefore, the vision to locate a malicious landing page on malware distribution sites of a static web graph is very limited.

Zhang et al. [4] has proposed method called Arrow which identifies malware distribution networks over internet. This method first collects the suspicious URLs by using client honeypot. These URLs are then mapped with IP address and hostname to eliminate the medley created by hostnames and IP addresses for representing a server. Further the central server is located in each MDN and redundant servers are eliminated to improve the performance. This central server is the main server in Malware Distribution Network (MDN) that leads to drive by download attack. The system generates signature based on regular expression to easily identify additional malicious web pages which are coded to launch drive by download attack. Arrow system`s detection rate of drive-by download attack is very significant and has large rate of low false positive. But it can prune the signature incorrectly and can wrongly mark HTTPTraces [14] as false positives. The attackers can also decentralize the MDNs to eliminate the central servers or hide the MDN structure from the client. This limitation has being significantly rectified by using Arrow system with rule base detection approach.

Wang et al. [5] has proposed method called Rule-Based Landing Pages Detection that discovers and analyses each landing pages in MDN that causes drive-by downloads. It takes feedback from Arrow system which is capable to identify MDNs. For all central servers in MDN the rule base system will scan and classify each web page to extract and detect malicious web content. Web pages content is analyzed and the feature or strings that have malicious redirection are extracted. These strings are clustered into set of groups which will reduce number of candidate feature from a large set of string to a smaller string cluster. Further using multi-class string selection algorithm we will be able to identify the malicious content on landing pages and also be able to distinguish between MDNs and legitimate web pages to reduce large number of false positive. This system is limited in use since it requires an initial set of MDNs.

Both Arrow and rule base detection method have some limitation, thus we require a more efficient and enhance method to detect more relevant malicious landing pages in an Malware Distribution Network which causes drive by download and lures the victim to install a fake anti-virus program. Therefore, we have proposed an approach called VILEEAR with these methods to detect drive by download attack more efficiently. Our approach differs from existing work where we have suggested combining and inheriting existing two methods that locate malicious landing page web servers in MDNs and examine thoroughly each web page on this network so the outcome can be forwarded to our proposed system for further optimized result to detect Drive-by attack.

3. Proposed System

In this section we will discuss our proposed mechanism VILEEAR to detect Drive-by download attack with the combination of Arrow and rule base detection. We have described the Arrow system and Rule base detection approach in previous sections which provides an organized way to locate MDNs over the internet and analyze web pages to detect malicious activity related to Drive-by download attack.

3.1 System Overview

We have propose an approach called VILEEAR that will search the web more efficiently and produces the pages that are more likely to be malicious as compared to previous system. VILEEAR uses known malicious web pages URLs as initial input. The ear will help our system to query the search engine and identify more relevant malicious pages which are similar to the initial input. As a result more malicious URLs can be detected as compared to random page detection on the web and it also helps to improve the crawling method of search engines.

VILEEAR uses guided search for malicious URLs to enhance random web crawling approach. VILEEAR uses initial pages to starts detection process. These pages are mostly created by cybercriminals and are hosted on malicious server to cause drive by download exploits. VILEEAR considers a page as malicious if that page executes drive-by download script on user browser. VILEEAR will also detect a malicious pages that will lure the victim`s to install a fake anti-virus program.

Using Arrow and Rule base approach with VILEEAR in tandem the system will work like supervised learning process for finding malicious web page in an MDN and will keep on improving the process of detecting Drive by Download attack after each iterations. Thus, by using our approach we will be able to find more relevant web pages that are malicious in a Malware Distribution Network to detect Drive-by attacks.

3.2 System Architecture

The general architecture of system is shown in Figure 2 where Arrow will generate malicious landing pages by finding central server of MDNs. These landing pages are given to rule base detection system as initial input for further selection and detection of malicious feature in landing pages. Rule base finds malicious features in web pages and will give a set of suspicious pages which is forwarded further to our system called VILEEAR.

The components of our system are elaborated in more detail in next sections and we will see how it enhances the mechanism to retrieve and detect malicious landing pages which causes drive by download attacks.

3.2.1. Suspicious Ear Pages

The Suspicious ear pages are a malicious pages set that are already detected by rule base detection system. These pages will form the input to gizmos. Whenever gizmos will discover new pages they will add it to the set of ear pages. There are two main types of ear pages: a) Pages that are set up by cybercriminals. Attacker set up the pages that mostly contain JavaScript code that causes exploit, and has links to malware software, such as fake Anti-Virus programs. b) The second type of pages is the benign pages that have been compromised. Such legitimate landing pages will contain a small HTML or JavaScript code that will redirects the victim to MDN.

3.2.2. Gizmos

Gizmos are the heart of VILEEAR. The suspicious ear pages are given as input to gizmos. The purpose of a gizmo is to find candidate pages that are likely to be malicious and to generate more malicious pages that are similar or relevant to the initial input pages.

All gizmos perform the same processing steps. First, they will analyze the ear pages to search similar characteristics or properties. Second, they will expand the initial ear by querying the external search engines to identify more relevant pages that contain similar properties.

We have proposed four gizmos for our system to detect Drive by download attack and fake Anti-Virus program installation.

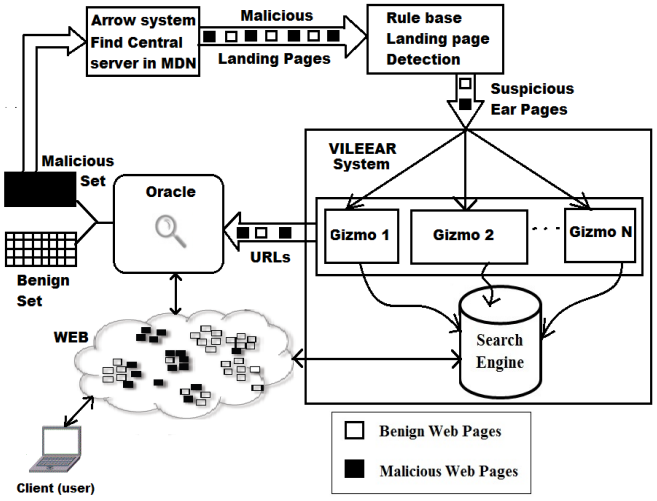

Figure 2: General System Architecture

(a) Link Gizmo

The link gizmo provides great effect on web topology (web graph) to locate pages that redirect to malicious sites. Link gizmo is developed to locate group of pages that redirects to all infected URLs. Such pages are called malware hub, these hubs contains vulnerable sites that are attacked multiple times. This gizmo improves the observation and thus will represent valuable candidate URLs.

(b) Content Dork Gizmo

The content dork gizmo focuses on identifying vulnerable and exploited web applications. The most effective technique to locate vulnerable web sites is to query a search engine with a Google dork. It uses special keywords to detect the links that redirects to vulnerable sites. For example, some search engine is fed with a query to locate web sites that share their details. Similarly, to find a web site that uses a previous version of popular software's can also be exploited by using known queries. Such content dork gizmo is used by cybercriminals to find vulnerable sites, but the benefit of this gizmo is that it has ability to identify suitable dorks automatically.

Advantages of using content dork are that it can detect a wide diversity of less popular but more vulnerable web application. Also, they can quickly react to the attack that exploits previously unknown vulnerability.

(c) Search Engine Optimization Gizmo

Large numbers of vulnerabilities on web sites are exploited by cybercriminals to take control of system. Such web sites is visited by a very small numbers of users, thus drive-by attacks injected into these websites can harm to only a small pool of potential victims. To reach more users, attackers use various techniques to drive all the traffic to the malicious pages which are under their control. One of the popular techniques is using a Search Engine Optimization (SEO) method to increase the ranking of malicious pages in search engine results for popular search terms.

Nowadays attackers started using SEO kits that automatically generates "sets" of pages that optimizes currently popular search topics. SEO kit uses the semantic cloaking [16]-[17] which is the response of the exploited web sites with completely different content.

Detecting semantic cloaking and distinguishing it from syntactic cloaking [18] is possible. To detect cloaking, we visit a URL three times, providing different values for the "User-Agent" and "Referer" HTTP headers.

We will follow HTTP redirections and will consider the final "landing" domain from which the browser receives a successful HTTP response. We will detect cloaking if, during the three visits to a URL, we observe two or more different landing domains.

(d) Domain Registration Gizmo

The most widespread techniques used to protect users against web malware are by using blacklist for suspicious web sites. For domain-based blacklist, a domain will be added to the list as soon as it is discovered to host malicious content. To avoid domain-based blacklisting cybercriminals uses a short-lived domain that frequently switches the domains to maximize the time to remain unlisted. To implement such concepts cybercriminals uses automatic domain generation and registration process. This automation has some drawbacks that provide us some leverage to identify these domains. More precisely, we assume that automatic domain registrations are mostly close in time with the registration of known malicious domains and are also likely to be malicious. Domain Registrations Gizmo stores record of domains that host malicious pages, and domain registration records are easily available online. This gizmo extracts the domain of a malicious ear URL, and marks the domains that are registered before and after as suspicious.

3.2.3. Search Engine Queries

The information provided by the gizmos and initial ear pages is transformed into queries and are sent to search engine. These queries searches for the similar words on the web page resulted by search engine. Gizmo considers the parts of pages that are newly tagged by a search engine. Once a query is issued, the gizmos will gather the set of URLs from search engines, and they forward them to the oracle for further analysis.

3.2.4. Oracles

An oracle is nothing but an client honeypot that will analyze a web page to detect whether it is malicious or not. In our system, the oracle will consist of three components Blacklist for Safe Browsing [18], Wepawet [13], and a tool to detect web pages that host fake antivirus software.

Google creates blacklist of malicious pages which gets regularly updated. These blacklists, is easily accessible through the Safe Browsing API, and are created by analyzing more than million pages daily. The analysis is done by honeyclients, which are capable to detect DBD attacks with a low false positive rate.

Wepawet is a kind of oracle which uses a browser that captures the web pages which are executed by JavaScript. Based on this recorded execution traces, the system detects drive-by download attack using anomaly based technique. Detection rate of this oracle is very high with no false positives [14].

The third component, we use is a detector of web pages that host fake antivirus software. This detector uses signatures and frequency analysis to detect if a web page threats the users with security issues in their computers and lures them to download fake security software.

This Oracles is the final output of our system that provide us with large sets of malicious web pages that are hosted to launch Drive-by download attack on victims machine. Oracle is also a client honeypot and their results will be given back to system called Arrow as an optimized input for future enhancement. Thus our system will work like supervised learning process for finding malicious web page in an MDN and will keep on improving the process of detecting Drive-by Download attack after each iteration. Thus, by using our approach we will be able to find more relevant web pages

that are malicious in a Malware Distribution Network. Also it will improve the identification of content and feature selection of rule base approach. This system will also improve Arrow system to find central servers for MDNs that host malicious landing pages.

4. Conclusion

Detecting drive-by download attacks is an extremely important and challenging problem. We conclude that by collecting data from a large number of drive-by download attempts, the system will be capable to locate Malware Distribution Networks including one or more central servers. Our system will enhance the efficiency of the search process for malicious web pages. It also improves a ear of known, malicious web pages and extracts features and similarities from these pages that cause Drive by download attack. Using the combination of Arrow and rule base system with VILEEAR our system can retrieve a set of candidate web pages that contains a much higher rate of relevant malicious web pages, when compared to random crawling infrastructure. Our system uses supervised learning concept to improve detection rate of drive-by attacks and therefore by using such VILEEAR system, it is possible to improve the effectiveness of the malicious page discovery process and thereby detecting and protecting Drive by download attack more efficiently.

References

[1] N. Provos, P. Mavrommatis, M.A. Rajab, and F. Monrose, "All Your Iframes Point to Us," in *USENIX Security Symposium*, 2008.
[2] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, " Defending Browsers against Drive-by Downloads: Mitigating Heap-Spraying Code Injection Attacks," in *Secure Systems Lab, Technical University Vienna*, Austria, 2009
[3] Provos, N., McNamee, D., Mavrommatis, P., Wang, K., Modadugu, N, "The Ghost In The Browser Analysis of Web-based Malware," in *First Workshop on Hot Topics in Understanding Botnets, HotBots 2007*, 2007
[4] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee, "Arrow: Generating signatures to detect drive-by downloads," in *Proc. WWW*, 2011.
[5] G.Wang, "Detecting Malicious Landing Pages in Malware Distribution Networks," in *Computer Science UC Santa Barbara Santa Barbara*, 2013.
[6] C. Seifert, I. Welch and P. Komisarczuk, "Honeyc – the low-interaction client honeypot," in *Proc. NZCSRCS*, 2007.
[7] C. Seifert and R. Steenson, "Capture - honeypot client (capture-hpc)," https://projects.honeynet.org/capture-hpc, 2006.
[8] 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 *Proc. NDSS*, 2006.
[9] C. Seifert, R. Steenson, T. Holz, B. Yuan and M. A. Davis, "Know your enemy: Malicious web servers," http://www.honeynet.org/papers/mws/, 2007.
[10] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy, "A crawler-based study of spyware on the web," in *Proc. NDSS*, 2006.
[11] N. Provos, P. Mavrommatis, M. Abu Rajab and F. Monrose, "All your iframes points to us," in *Proc. USENIX SECURITY*, 2008.
[12] J. Nazario, "Phoneyc: A virtual client honeypot," in *Proc. LEET*, 2009
[13] M. Cova, C. Kruegel and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proc. WWW*, 2010.
[14] L. Lu, V. Yegneswaran, P. Porras and W. Lee, "Blade: An attack-agnostic approach for preventing drive-by malware infections," in *Proc. ACM CCS*, 2010.
[15] J. W. Stokes, R. Andersen, C. Seifert and K. Chellapilla, "Webcop: Locating neighborhoods of malware on the web," in *Proc. USENIX LEET*, 2010.
[16] B. Wu and B. D. Davison, "Detecting semantic cloaking on the web," 2006.
[17] K. Chellapilla and D. M. Chickering, "Improving cloaking detection using search query popularity and monetizability," in *2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2006.
[18] Google, "Safe Browsing API," http://goo.gl/vIYfk, 2011