Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 4, April 2013, pg.517 – 522

RESEARCH ARTICLE

Deep Malicious Website Detection

Parvatam Sriram Rohit[1], R. Krishnaveni[2]
[1]Department of Computer Science, Hindustan University, Chennai, India
[2]Department of Computer Science, Hindustan University, Chennai, India

[1] psriramrohit@gmail.com; [2] rskichu10@gmail.com

Abstract— To search for malicious web pages, the first step is typically to use a crawler to collect URLs that are live on the Internet. In this system, Deep Web Crawler is presented to detect the deep malicious web pages, an approach to search the web more efficiently for pages that are likely malicious. Deep Web Crawler uses the crawling infrastructure of search engines to retrieve URLs that are much more likely to be malicious than a random page on the web. In other words this Crawler increases the input to the URL stream. The deep web crawler also focuses on how to prepare appropriate queries to retrieve hidden pages. As vast amount of Web pages lie in the deep or invisible web these pages are typically only accessible by submitting queries to a database, and regular crawlers are unable to find these pages if there are no links that point to them.

Key Terms: - Deep Bot; Deep Web; Qiiiep based deep web crawler; web security; prefilter

I. INTRODUCTION

A. Web Security

Web security is a branch of information security that deals specifically with security of websites, web applications and web services. At a high level, Web security draws on the principles of application security but applies them specifically to Internet and Web systems. The core area of the project deals with the deep malicious websites. The maliciousness [3] is not only injected onto the surface web but also injected onto the deep or invisible web where vast and hidden information is available. There is chance for attackers to attack the users system and their user applications in the case of deep web by injecting malicious code into the deep web. As vast amount of Web pages are lie in the deep or invisible Web. These pages are typically only accessible by submitting queries to a database, and regular crawlers are unable to find these pages if there are no links that point to them. Deep web crawling also multiplies the number of Web links to be crawled. In some cases, such as the Google bot, web crawling is done on all text contained inside the hypertext content, tags, or text. Strategic approaches may be taken to target deep web content. With a technique called screen scraping, specialized software may be customized to automatically and repeatedly query a given Web form with the intention of aggregating the resulting data. Such software can be used to span multiple Web forms across multiple Websites.

B. Crawler

Web crawler [2] is a system for the bulk downloading of web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. The crawler is seeded with trending terms from Google Trends and Twitter terms, which are known to be often abused by attackers. The crawler engines of today cannot reach most of the information contained in the Web. A great amount of valuable information is hidden behind the query forms of online databases, and/or is dynamically generated by technologies such as JavaScript. This portion of the web is usually known as the Deep Web or the Hidden Web. The problem of crawling the hidden web can be divided into two challenges: Crawling the server-side hidden web: Many websites offer query forms to access the contents of an underlying database. Conventional crawlers cannot access these pages because they do not know how to execute queries on those forms. Crawling the client-side hidden web: Many websites use techniques such as client-side scripting languages and session maintenance mechanisms. Most conventional crawlers are unable to handle this kind of pages.

C. Deep Bot

A prototype system for crawling the hidden web a Deep Bot describes in detail the techniques it uses for accessing the content behind web forms. The main features of Deep Bot are for accessing the server side deep web, Deep Bot [2] can of domain definitions, each one describing a certain data-gathering task. Deep Bot automatically detects forms relevant to the defined tasks and executes a set of predefined queries on them. Deep Bot's crawling processes are based on automated mini web browsers, built by using browser APIs. This enables our system to deal with client-side scripting code, session mechanisms, and other complexities related with the client-side hidden web. As well as in conventional crawlers, the functioning of Deep Bot is based on a shared list of routes, which will be accessed by a certain number of concurrent crawling processes, distributed into several machines. The main singularities of our approach are: In conventional crawlers, routes are just URLs. Thus, they have problems with sources using session mechanisms. This system stores, with each route, a session object containing all the required information cookies, etc. to restore the execution environment in which the crawling process was running in the moment of adding the route to the master list.The crawler is seeded with trending terms from Google Trends and Twitter terms, which are known to be often abused by attackers. The crawler engines of today cannot reach most of the information contained in the Web. A great amount of valuable information is "hidden" behind the query forms of online databases, and/or is dynamically generated by technologies such as JavaScript. This portion of the web is usually known as the Deep Web or the Hidden Web. DeepBot is a prototype hidden web crawler able to access such content. DeepBot receives as input a set of domain definitions, each one describing a specific data-collecting task and automatically identifies and learns to execute queries on the forms relevant to them. DeepBot and report the experimental results obtained when testing it with several real world data collection tasks.

D. Deep Web Crawler

As vast amount of Web pages lies in the deep or invisible Web. These pages are typically only accessible by submitting queries to a database, and regular crawlers are unable to find these pages if there are no links that point to them. Deep web crawling also multiplies the number of Web links to be crawled. In some cases, such as the Googlebot, Web crawling is done on all text contained inside the hypertext content, tags, or text.

Strategic approaches may be taken to target deep-Web content. With a technique called screen scraping, specialized software may be customized to automatically and repeatedly query a given Web form with the intention of aggregating the resulting data. Such software can be used to span multiple Web forms across multiple Websites. Data extracted from the results of one Web form submission can be taken and applied as input to another Web form thus establishing continuity across the Deep Web in a way not possible with traditional web crawlers.

A traditional crawler picks up a URL, retrieves the corresponding page and extracts various links, adding them to the queue. A deep Web crawler, after adding links to the queue, checks for forms. If forms are present, it processes them and retrieves the required information. Various techniques have been proposed for crawling deep Web information, but much remains undiscovered. To minimize limitations of existing deep Web crawlers, a novel architecture is proposed based on QIIIEP specification. This architecture is cost effective and has features of privatized search and general search for deep Web data hidden behind html forms.

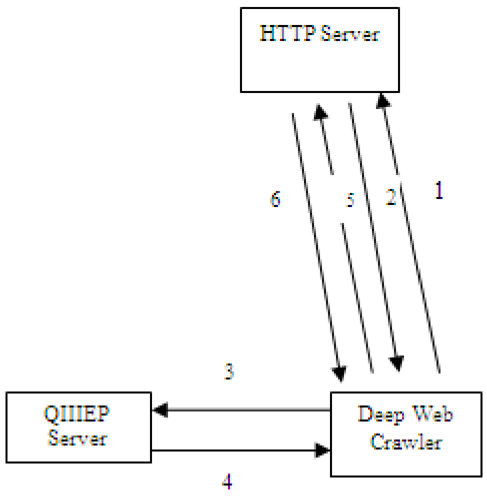


Fig 1 Mechanism of Qiiiep Based Deep Web Crawler

The work process of deep web crawler is given below

1. Request for page
2. Content/Form analysis
3. Request for query word to fill form
4. Received query words
5. Submission of filled form
6. Crawl deep web content

II. Existing System

In existing system, Malicious web pages [4] that use drive-by download attacks or social engineering techniques to install unwanted software on a user's computer have become the main avenue for the propagation of malicious code. To search for malicious web pages, the first step is typically to use a crawler to collect URLs that are live on the Internet. Then, fast pre-filtering techniques are employed to reduce the amount of pages that need to be examined by more precise, but slower, analysis tools such as honey clients.

A. Constraints of Existing System

- The effectiveness of evilseed [2] is dependent on the quality of seed and diversity of the malicious seed as they are used as input.
- There are problems in designing effective oracles, Wepawet and the Google Safe Browsing blacklist for the detection of malicious web pages.

An attacker might try to prevent evilseed from finding and detecting malicious pages. The most radical approach would be to make sure that these pages are not indexed by search engines in the first place. This can easily be achieved by an attacker who has full control of an exploited website.

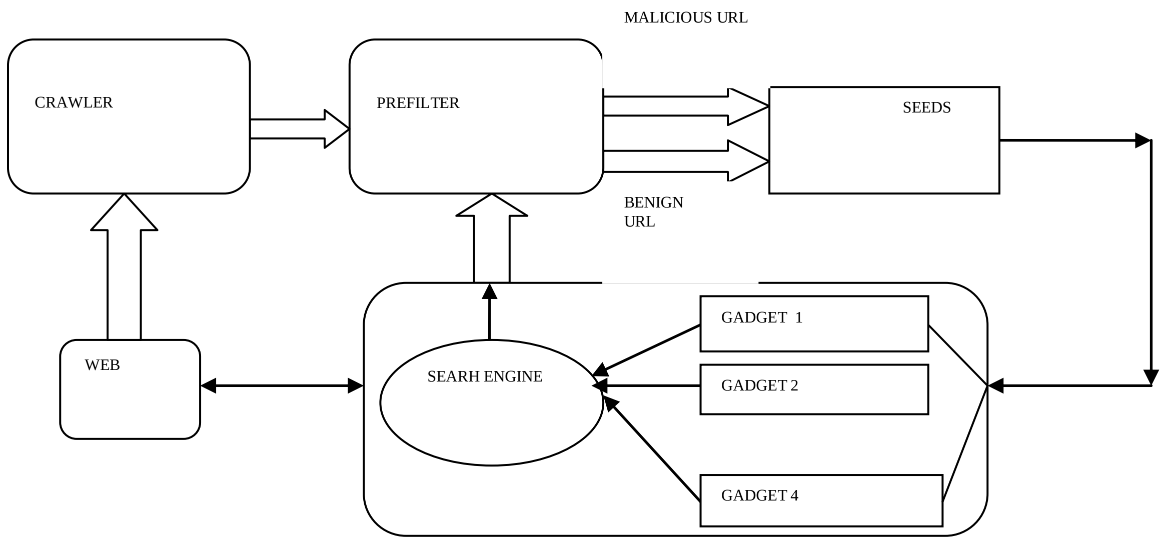
III. Proposed System

The proposed system implements the aspect of security analysis where attackers could try to perform evasion attacks against the detection techniques and the Safe Browsing system. Therefore addition of any gadget with any other oracle for the detection of malicious web pages can increase the difficulty of evasion attempts.

Performance is another aspect to be implemented where the bottleneck of evilseed is the cost of performing in-depth analysis with an oracle.

IV. System Design

The overall architecture below defines the system in order to detect the deep malicious web sites where the vast and hidden information is available. It uses the deep web crawler for the detection of deep malicious websites as the normal crawler does not crawl into hidden websites.



The main part of the existing system lies in the set of gadgets. These gadgets consume a feed of web pages that have been previously identified as malicious as well as other data feeds, such as domain registration feeds. Based on their input, the gadgets generate queries to search engines. The results returned by the search engines are then forwarded to an existing analysis infrastructure.

The architecture consists of the following:
Crawler
Prefilter
Link and Web content analyzation
Search engine optimization
Domain registration
DNS queries

A. Crawler
A web crawler is a system for the bulk downloading of web pages. Web crawlers [2] are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries.

B. Prefilter
Prefitering techniques are employed to reduce the amount of pages that need to be examined by more precise. The fast filter uses static characteristics of a page to quickly identify and discard pages that are likely to be benign, in a vein similar to Prophiler. This setup allows us to compare evilseed with a traditional, crawler-based infrastructure for finding malicious web pages.

C. Seed and Gadgets
The seed is a set of pages that have been previously found to be malicious. These pages form the input to gadgets. Whenever gadgets discover new pages that are confirmed to be malicious, they can be added to the set of seed pages. These are the heart of evil seed. The purpose of a gadget is to find candidate pages i.e,.URLs that are likely malicious based on the pages contained in the seed.

D. Link and Web Content Analyzation
The Link analyzation is designed to locate malware hubs on the web. Malware hubs are pages that contain links to several malicious URLs [6]. The owner of the website page will contain some data i.e., text or image, based on it the page will be developed by the website owner. This page can easily be taken and misuse in malicious site website. Detection of the malicious code in the websites is also being done by the help of this

gadget. The web content analyzation is also done by the help of this gadget in such a way that this gadget will be able to analyze the whole web content. By this, the website is malicious or non-malicious can be known.

E. Search Engine Optimization

The main server is having all the information of original & malicious web sites. Each web site has who is information along with the IP address. WHOIS, is all about the website registration, name to whom the web site is registered, along with the company details. Every Web site has a IP address, which will be used for authentication. Malicious Database is always updated with the malicious website's details for verification.

F. Domain Registration

Doman registration gadget is a database of all domain names registered in the domain. A registry operator is the part of the Domain Name System of the Internet and keeps the database of domain names. The DNS server will also have the complete details regarding the Domain Name & Inter domain in the web address. Each & every web site will have Domain name (.Com, .Co.in, .Edu,. Tech, .Co.uk, .in, Org & etc). Interdomian is all about any two domain names in the same link, www.123.com/456.com. Malicious Database is always updated with the malicious website's details for verification.

G. DNS Queries

Analyzation of recursive DNS traces will be done by this gadget. It identifies the domain names of the web pages that are likely to lead to malicious pages. The location of the referral sites of malicious web pages will be analyzed by the DNS Query gadget.

V. RELATED WORK

N. Provos et al.[4] proposed the analysis of Web based Malware system. As more users are connected to the Internet and conduct their daily activities electronically, computer users have become the target of an underground economy that infects hosts with malware or adware for financial gain. Unfortunately, even a single visit to an infected web site enables the attacker to detect vulnerabilities in the user's applications and force the download a multitude of malware binaries. Frequently, this malware allows the adversary to gain full control of the compromised systems leading to the ex-filtration of sensitive information or installation of utilities that facilitate remote control of the host. We believe that such behavior is similar to our traditional understanding of botnets. However, the main difference is that web-based malware infections are pull based and that the resulting command feedback loop is looser. To characterize the nature of this rising thread, we identify the four prevalent mechanisms used to inject malicious content on popular web sites: web server security, user contributed content, advertising and third-party widgets. For each of these areas, we present examples of abuse found on the Internet. Our aim is to present the state of malware on the Web and emphasize the importance of this rising threat.Single visit to an infected web site enables the attacker to detect vulnerabilities in the user's applications and force to download a multitude of malware binaries which further leads to ex filtration of sensitive information or installation of utilities that facilitate remote control of the host. The deep websites which are malicious cannot be detected.

M. Cova et al.[1] proposed the Analysis of Rogue AV Campaigns Rogue antivirus software has recently received extensive attention, justified by the discussion of its propagation. A longitudinal analysis of the rogue antivirus threat ecosystem, focusing on the structure and dynamics of this threat and its economics. To that end, the compiled and mined a large dataset of characteristics of rogue antivirus domains and of the servers that host them. The contributions of this paper are threefold. Firstly, to our knowledge, broad analysis of the infrastructure underpinning the distribution of rogue security software by tracking 6,500 malicious domains. Secondly, to apply attack attribution methodologies to correlate campaigns likely to be associated to the same individuals or groups. By using these techniques, we identify 127 rogue security software campaigns comprising 4,549 domains. Finally, contextualize by comparing them to a different threat ecosystem that of browser exploits. Underline the profound difference in the structure of the two threats, and investigate the root causes of this difference by analyzing the economic balance of the rogue antivirus ecosystem.

Disadvantage: Rouge AV campaigns not able to detect the malicious websites which leads to the injection of malicious code into the system. Information regarding the deep malicious websites is not given.

M. Polychronakis et al.[3] proposed the system of exploring the Life Cycle of Web-based Malware While the web provides information and services that enrich our lives in many ways, it has also become the primary vehicle for delivering malware. Once infected with web based malware, an unsuspecting user's machine is converted into a productive member of the Internet underground. In this work, we explore the life cycle of web based malware by employing light-weight responders to capture the network profile of infected machines. Our results indicate that web-based malware provides a cornerstone for large scale electronic fraud. It is used to exfiltrate address books of compromised machines creating databases of hundred millions of email addresses, to form spamming botnets responsible for a significant fraction of spam currently seen on the Internet, and also to steal login credentials that can be directly monetized or leveraged to turn more web servers into malware delivery vectors.

Disadvantage: Although the information regarding the maliciousness is given, the insights on the network activities of malware once installed on a system were not given. Detection of malicious websites has not been discussed.

VI. CONCLUSION

A novel approach is developed to improve the effectiveness of the search process for malicious web pages. A seed is used for detecting malicious web pages and extract characterizing similarities that these pages share. The infrastructure of search engines is developed so that they collect and quickly identify other pages that show the same pages that are likely malicious. This has been implemented in a tool called evilseed by the help of gadgets. As a future work more number of gadgets can be added in the aspect of security and performance where attackers could try to perform evasion attacks against the detection techniques.

REFERENCES

[1] M. Cova, C. Leita, O. Thonnard, A. Keromytis, and M. Dacier, “An Analysis of Rogue AV Campaigns,” in Symposium on Recent Advances in Intrusion Detection (RAID), Vol. 162, 2010,pp.234-236

[2] Luca.Invernizzi, Paolo Milani Comparetti, “Evil Seed : A Guided Approach to find Malicious Web Pages”, International Conference on Web Security, Vol. 195, 2012, pp.42-55

[3] M. Polychronakis, P. Mavrommatis, and N. Provos, “Ghost Turns Zombie: Exploring the Life Cycle of Web-based Malware,” in USENIX Workshop on Large-Scale Exploits and Emergent Threats, Vol. 191, 2008, pp.167-176

[4] N.Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, “The Ghost in the Browser: Analysis of Webbased Malware,” in USENIX Workshop on Hot Topics in Understanding Botnet, Vol.142, 2007, pp.122-136

[5] M.A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao, “The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution,” in USENIX Workshop on Large-Scale Exploits and Emergent Threats, Vol. 210, 2010, pp.154-162.

[6] Van Lam Le, Ian Welch, Xiaoying Gao, “Two Stage Classification Model to Detect Malicious Web Pages”, International Conference on Advanced Information Networking and Applications, Vol. 177, 2011, pp.68-77

[7] Wen Zhang, Yu-Xin Ding and Yan Tang, “Malicious Web Page Detection based on online learning algorithm”, International Conference on Machine Learning and Cybernatics, Vol. 181, 2011, pp.95-111

[8] P. Likarish, E. Jung, and I. Jo, “Obfuscated Malicious Javascript Detection using Classification Techniques,” in Conference on Malicious and Unwanted Software (Malware), 2009.

[9] D. Canali, M. Cova, C. Kruegel, and G. Vigna, “A fast filter for the large-scale detection of malicious web pages,” in International World Wide Web Conference (WWW), 2011.

[10] P.Likarish, E. Jung, and I. Jo, “Obfuscated Malicious Javascript Detection using Classification Techniques,” in Conference on Malicious and Unwanted Software (Malware), 2009.

[11] L. Lu, V. Yegneswaran, P. Porras, and W. Lee, “BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections,” in ACM Conference on Computer and Communications Security (CCS), 2010.

[12] K. Rieck, T. Krueger, and A. Dewald, “Cujo: Efficient Detection and Prevention of Drive-by-Download Attacks,” in Annual Computer Security Applications Conference (ACSAC), 2010.

[13] Sharma Dilip Kumar and Sharma A.K. “Query Intensive Interface Information Extraction Protocol for Deep Web” In Proceedings of IEEE International Conference on Intelligent Agent & Multi-Agent Systems, PP. 1-5, 2009

[14] D. K. Sharma and A. K. Sharma, “A QIIIEP Based Domain Specific Hidden Web Crawler”, In International Conference & Workshop on Emerging Trends and Technology (ICWET 2011),Mumbai, ACM Digital Library, 2011

[15] Dilip Kumar Sharma and A.K. Sharma, “A Novel Architecture of Deep web crawler” In International Journal of Information Technology & Web Engineering, USA, Vol. 6, No. 1,pp. 25-48, 2011