Identification of Potential Malicious Web Pages

Van Lam Le, Ian Welch, Xiaoying Gao, Peter Komisarczuk
School of Engineering and Computer Science, Victoria University of Wellington
P.O. Box 600, Wellington 6140, New Zealand
{van.lam.le, ian.welch, peter.komisarczuk, xiaoying.gao}@ecs.vuw.ac.nz

Abstract

Malicious web pages are an emerging security concern on the Internet due to their popularity and their potential serious impact. Detecting and analysing them are very costly because of their qualities and complexities. In this paper, we present a lightweight scoring mechanism that uses static features to identify potential malicious pages. This mechanism is intended as a filter that allows us to reduce the number suspicious web pages requiring more expensive analysis by other mechanisms that require loading and interpretation of the web pages to determine whether they are malicious or benign. Given its role as a filter, our main aim is to reduce false positives while minimising false negatives. The scoring mechanism has been developed by identifying candidate static features of malicious web pages that are evaluate using a feature selection algorithm. This identifies the most appropriate set of features that can be used to efficiently distinguish between benign and malicious web pages. These features are used to construct a scoring algorithm that allows us to calculate a score for a web page's potential maliciousness. The main advantage of this scoring mechanism compared to a binary classifier is the ability to make a trade-off between accuracy and performance. This allows us to adjust the number of web pages passed to the more expensive analysis mechanism in order to tune overall performance .

Keywords: Internet Security, Drive-by-download, malicious web page.

1 Introduction

A "malicious web page" refers to a web page that contains malicious content that can exploit a client-side computer system. This attack is delivered to client's web browser when a malicious web page is requested. This type of attack is termed web-based client-side attack. The attack is delivered as part of the web page itself and is designed to exploit client-side vulnerabilities such as flaws in the implementation of browser functionality, interpreters of active content within webpages or scriptable client-side components such as ActiveX components. The result of an attack is often the installation of malware in the client system without the user's consent and disclosure of user's information. The user's computer is often "owned" by attacker and can take part in generating SPAM and Distributed Denial of Service (DDOS) attacks.

Detection and blacklisting of malicious web pages has been the subject of several research projects. One effective approach is to build virtualised environments like high interaction client honeypots (Seifert 2007a) where suspicious web pages are loaded, executed and monitored to track potential malicious activities or behaviour. The virtualised environment allows this to be done without allowing any malware to be propagated to production systems. While this method shows very efficient results in term of detecting unknown attacks, it is expensive in terms of the resources required to provide a virtualised environment containing a complete operating system and is relatively slow with each visit taking up to 10 seconds. To attempt to reduce the required resources and increase the speed of the detection method, previous work (Seifert 2007a) has proposed using a hybrid approach where web pages are first filtered using a lightweight mechanism before being passed to the more expensive high-interaction mechanism. Our work focuses on improving the efficiency and effectiveness of that lightweight mechanism.

There are three main issues that we have explored in the design of our lightweight mechanism. Firstly, we want our mechanism to be lightweight in terms of its resource requirements. Therefore our mechanism is a data-mining algorithm that uses features derived from the static web page rather than runtime features gathered through the expensive process of loading the web page into a web browser within a virtual environment. This paper proposes a set of features that have been arrived at through analysis of known malicious web pages. These features are then evaluated by feature selection methods in order to find out the most suitable feature set to identify potential malicious web pages. Secondly, we want our lightweight mechanism to be tuneable to allow us to control the number of pages passed through to the more expensive mechanisms such as high interaction honeypots. This allows us to manage overall system performance. This has led us to develop a lightweight mechanism that computes a score rather than a simple binary malicious/benign classifier (Seifert, Welch and Komisarczuk 2008). By choosing the threshold that must be reached before passing on the web page, the overall performance can be tuned to reflect overall performance constraints. Thirdly, we believe that it is worse to miss a potential malicious web page (a false negative) than incorrectly class a web page as malicious (a false positive) and pass it onto the second stage for further analysis. Therefore, our aim has been to design a mechanism that minimises the number of false negatives whilst keeping the false positives at an acceptable level. Note that when taking resource usage into account that

there will most likely be a relationship between our choice of threshold value and the false negative rate and part of our interest is in understanding this relationship.

2 Background and Related Work

2.1 Web-based Client-side Attacks

As the number of Internet users has increased significantly, web-based attacks that use malicious web pages to exploit users' system have become a primary concern in the Internet security. A web-based client-side attack happens when an Internet user visits malicious web pages which attempt to exploit the user's browser vulnerabilities, plug-in application vulnerabilities or user's operating system vulnerabilities in order to compromise the user's system.

A web application is defined as an network application which is typically interacting with the web browser over the Internet (Mehdi 2007). Information service providers use web applications to deliver their services to users. To do that, they implement their business logic through web applications at a web server with an advertised URL (Gollmann 2008). To enrich their services, the providers can use more than one web server and backend servers and applications which work in cooperation in order to deliver services to the customers. In the client-side, there is the main application – web browser which users use to access information services from the providers. In order to expand their functionalities, almost all web browsers support adding third-party plug-in components such as Adobe Acrobat, Adobe Flash, Apple QuickTime, and Microsoft ActiveX.

To deliver malicious content to the client-side, an adversary first needs to publish malicious contents on the Internet. Compromising a web server is one of the common ways to deliver malicious contents. Various methods are reported to be used to increase attack effectiveness (Websense 2008, Sophos 2009, ScanSafe 2009, Symantic April 2009, ScienceDirect 2008, Websense 2009). Intruders can compromise a website by exploiting some vulnerabilities in the web server, exploiting a vulnerable web application (Symantic April 2009), vulnerable database applications such as SQL injection (Niels, Moheeb Abu and Panayiotis 2009, ScanSafe 2009, Microsoft 2009). The results from this compromising are inserting malicious contents which can be delivered to the client-side system (Niels, Moheeb Abu and Panayiotis 2009, Microsoft 2009). Some vulnerabilities in web server and web applications are reported as a very common issue(Provos, Mavrommatis, Abu and Monrose , Symantic April 2009). Web 2.0 technology, in addition, has become a common environment for attackers to spread their malicious contents (Websense 2008, Adam and Meledath 2008). Visitors are allowed to put arbitrary HTML and they can insert malicious codes into websites, insert links to malicious sites or even upload malicious files (Provos, McNamee, Mavrommatis, Wang and Modadugu 2007, Adam and Meledath 2008, Patsakis, Asthenidis and Chatzidimitriou 2009, Lawton 2007).

After publishing their malicious contents on the Web, attackers must get users to visit the malicious web pages in order to make exploitation (Niels, Moheeb Abu and Panayiotis 2009). Spam is a common technique which intruders use to lure user to their malicious web pages. For instance, spam emails can contain a links to a malicious web page. Web blogs and social networking sites are also abused to get users to visit malicious sites (Garrett, Travis, Micheal, Atul and Kevin 2008). In addition, some legitimate sites have third-party contents like access counters, advertisements which refer to malicious sites (Alme 2008, Provos, McNamee, Mavrommatis, Wang and Modadugu 2007, Websense 2008, Barth, Jackson and Mitchell 2009). Moreover, search engine are also abused by attackers in order to get users to visit their malicious sites. Popular search terms are used to make malicious web pages be displayed in the search results (Keats and Koshy 2009, Alme 2008, Barth, Jackson and Mitchell 2009, Gyongyi and Garcia-Molina 2004, Websense 2009) so there is a very high chance for their malicious sites to be visited.

When a user visits a malicious site, malicious contents are delivered to exploit the user's system. Malicious code is usually used to target a specific vulnerability of the web browser itself or plug-in applications (Jose, Ralf, Helen and Yi-Min 2007, Charles, John, Helen, Opher and Saher 2007). To discover available vulnerabilities in the user's system, adversaries abuse scripting support via JavaScript, Visual Basic or Flash to collect information about the user's computing environment (Provos, McNamee, Mavrommatis, Wang and Modadugu 2007). Moreover, obfuscation is used to hide exploit code in order to make malicious pages hard to be detected (Seifert, Welch and Komisarczuk 2008, Seifert 2007b, Seifert, Steenson, Holz, Yuan and Davis 2007).

In addition, Seifert's study about malicious web servers shows that there are some available web exploitation kits (Seifert 2007b). These web exploitation kits are very powerful in term of compromising web servers and delivering malicious contents. The result from this kind of attacks is usually to redirect users' requests to malware distribution networks. In addition, other related researches also show that malicious web pages are delivered by malware distribution networks (Provos, Mavrommatis, Abu and Monrose , Wang, Beck, Jiang and Roussev 2006, Jianwei, Yonglin, Jinpeng, Minghua, Xulu, Weimin and Yuejin 2007).

2.2 Related Work

In this section, we preview some current analysis methods which are used to detect malicious web pages. They are classified into three main approaches: Signature approach, state-change approach and machine learning approach.

2.2.1 Signature technique

In the signature approach, detection systems use known signature to detect malicious web pages. Signatures can be from some well-known Intrusion Detection Systems (IDS) or anti-virus applications. This approach is commonly used in the detecting system using low interaction client honeypot. Snort signature is used to detect malicious web pages in their HoneyC system (Seifert, Welch and Komisarczuk 2007). The HTTP responses from web servers are constructed under XML format, and then analysed against Sport signatures. In

Monkey-Spider system, Ikinci, Holz and Freiling also used signature approach to detect malicious websites. The contents of websites are crawled and stored in files. The crawled contents are then scanned by ClamAV – an anti-virus application (Ikinci, Holz and Freiling 2008).

2.2.2 State-change technique (rule-based technique)

In addition, state-change approach is commonly used in the detecting systems using high interaction client honeypot – one of the efficient instruments to detect malicious web pages. The main idea of this approach is monitoring the state change in the client system during visiting an URL time. If there is any unauthorized state change during visitation, the visit URL is classified as malicious. In the Strider HoneyMonkeys system, a monkey program loads a browser, instruct it to visit each URL and wait for a few minutes for downloading process. The state changes in the system is then detected against unauthorized creating executable files or registry entries in the system (Wang, Beck, Jiang and Roussev 2006). Moreover, to detect drive-by-download attack, Moshchuk, Bragin, Gribble and Levy use event triggers. They create some trigger conditions to track unauthorized activities in process creation, file system and registry system. The trigger conditions also include any event that makes browser or the system crash. During visitation, if an URL make a trigger fire, it is classified as unsafe (Moshchuk, Bragin, Gribble and Levy 2006). The state change approach is also used by Xiaoyan, Yang, Jie, Yuefei and Shengli in their client honeypot system to collect Internet-based malware. A behaviour monitoring module is conducted to track malicious behaviour. It hooks native API, DLL functions and TDI in order to monitor all activities causing buffer overflow, accessing system resources such as process, network, file, and registry (Xiaoyan, Yang, Jie, Yuefei and Shengli 2008).

2.2.3 Machine Learning Approaches

Seifert et al. (Seifert, Welch and Komisarczuk 2008) proposed a novel classification mechanism to detect malicious web pages. This method is based on HTTP responses from potential malicious web servers which are then analysed to extract potential malicious characteristics. The method was used in a hybrid system in which all URLs are classified by static heuristic method and sent to high interaction client honeypot for verification. To classifying URLs by static heuristics method, some common attributes are chosen based on three proposed main elements in malicious web pages: exploit, exploit delivery mechanism and obfuscation. The first step in this method is collecting malicious and benign web pages and then extracting potential attributes from these web pages. In learning step, all attributes extracted from 5,678 instances of malicious and 16,006 instances of benign web pages were fed into Weka with J4.8 decision tree learning algorithm implementation. The outcome classifier from learning step was used to classify 61,000 URLs. This classifier had very good false positive rate (5.88%) but very high false negative rate (46.15%).

Hou et al proposed a machine learning approach to detect malicious web content (Hou, Chang, Chen, Laih and Chen 2009). The key point in this research is the method used to choose features according to the usages of DHML knowledge. The chosen features have to meet the requirement for abilities against obfuscation vs. accuracy. Three groups with 171 features were chosen. There are 154 features used to count the use of native Java functions. Nine features are also used to measure some elements inside a HTML documents. There is 8 advanced features are used to count the use of ActiveX object. In the first step, 965 benign and 176 malicious web pages were collected, analysed and labelled manually. The malicious web pages were then categorized into nine pre-defined types based on the skill used by attackers. In order to study about choosing type of features, the authors took some experiments with different chosen features. Decision tree algorithm is used in these experiments. While using all features cannot get high true positive and low false positive result, the combination of three features can get very good result. The authors also compared the results of different classification algorithms with the use of all the features. Four classification algorithms used in this comparison are decision tree, Naïve Bayes, SVM and boosted decision tree. The result showed that the boosted decision tree got the best performance with high true positive rate and low false positive rate.

To detect malicious web pages, Liang (Bin, Jianjun, Fang, Dawei, Daxiang and Zhaohui 2009) proposed the concept of abnormal visibilities. According to their studies, malicious web pages are usually changed in their display modes in order to be invisible or almost invisible. The authors showed three main forms of abnormal visibility. The first one is changing the width and height attributes of iframe in order to make embedded malicious codes invisible or almost invisible. Setting the display style of iframe 'display: none' is the second form of abnormal visibility. The last form is generating iframe tag dynamically in order to make obfuscation. Abnormal visibility fingerprints are created and used to detect malicious web pages. Each web page is scanned to detect any form of abnormal visibility. The detected value in any kind of abnormal visibility is compared with a threshold value. If the detected value is less than the threshold value, the web page has an abnormal visibility and is considered as a possible malicious page. To carry out the experiment, the authors detect 60 websites reported malicious by StopBadWare.org. They scanned 66882 pages from these websites and found 30561 malicious one. They also figured out that their system has low false positive (1.99%) and false negative rates (2.63%).

Ma et al. (Ma, Saul, Savage and Voelker 2009a) pinpointed a new approach to detect malicious web pages named lightweight URL classification. In this approach, they classify web pages based on relationship between URLs, their lexical and host-based features. It does not use contents of web pages in detection. Lexical features include any features which make the page 'look different'. They can be the length of the host-name, length of the entire URL, number of dot in URL and so on. Hosted-base feature include IP address properties, WHOIS properties, Domain name properties and geographic properties. Naive Bayes, SVM and Logistic Regression are used for classification. The authors used two experiments in their study. The first experiment is for

comparing between feature sets. The features were divided into nine feature sets and these sets were fed into the ℓ1-regularized logistic regression (LR) classifiers. The results showed that using more features got better classification accuracy. In addition, their another experiment (Ma, Saul, Savage and Voelker 2009b) was conducted to build online learning algorithm to detect malicious web pages. They used the same feature as the experiment (Ma, Saul, Savage and Voelker 2009a). There were three online algorithms implemented: Perception, Logistic Regression with Stochastic Gradient Descent and Confidence-Weight. They compared their online learning algorithm with Support Vector Machine (SVM). The results showed that SVM needed more training data set in order to get better accuracy but their algorithms did not.

To build an inductive learning model to detect malicious web pages, Liu et al. (Liu and Wang 2009) extracted features from HTTP responses such as iframe, javascript, body redirect, css redirect etc. The inductive learning model consisted of behaviour signatures based on extracted features and the relationship of features. The results from their experiment showed that the inductive learning model missed many malicious web pages (46.15%).

Chia-Mei et al (Chia-Mei, Wan-Yi and Hsiao-Chung 2009) proposed a model to detect malicious web pages based on unusual behaviour features such as encoding, sensitive key word splitting and encoding and some dangerous JavaScript functions. To classify web pages, they created a scoring mechanism which cored based on 9 predictor variable. Moreover, weights for each predictor variable were decided by training phrase. The results from their experiment showed that their model worked very well. However, their dataset was very small with 460 benign and 513 malicious web pages.

Shih-Fen et al. (Shih-Fen, Yung-Tsung, Chia-Mei, Bingchiang and Chi-Sung 2008) proposed a novel semantics-aware reasoning detection algorithm to detect malicious web pages (SeAR) which was based on structures of HTML codes. Firstly, they defined templates for HTML codes. For each tested HTML code, the distance between the tested HTML code and templates were calculated. Secondly, the best match was chosen based on the distance and weight of the template. Finally, threshold was used to make decision whether web pages were classified as malicious or benign. The outcome from this research is very good but their dataset had only 147 malicious instances (no benign one).

Cova et al. (Cova, Kruegel and Vigna 2010) presented a novel approach which used anomaly detection and emulation to identify malicious JavaScript Code. The features were chosen based on sequence of carrying out an attack: redirection and cloaking, de-obfuscation, environment preparation, and exploitation. They argued that not all of the features were necessary for an attack happening and classified the features into two groups: useful features and necessary features. To extract features, they used emulated HTML browser HtmlUnit (Gargoyle). They carried experiments on over 115K web pages and their approach achieves very good outcome in comparison to other approaches such as ClamAV, PhoneyC and Capture-HPC.

While there is a few of works focusing on identifying malicious web pages, this paper presents a mechanism to detect potential malicious one in order to reduce number of suspicious web pages which need to be investigated further by detection instruments or experts.

3 Scoring Mechanism

This work focuses on how to reduce number of suspicious web pages but minimize missing attacks. A scoring mechanism is proposed to work as a filter which classifies suspicious web pages into classes: benign web pages and potential malicious web pages. Only potential malicious web pages are forwarded to detection devices or experts for further investigations (Fig. 1).

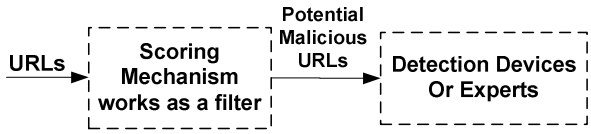

Figure 1: Scoring Mechanism

We propose scoring mechanism because of three reasons. Firstly, it works as a filter, not a final classifier so it just makes an estimate by scoring maliciousness of web pages. Secondly, it uses static features which can be obtained without rendering fully or executing web pages. However, they are less valuable than run-time features which are extracted by rending fully and executing web pages. Therefore, static features are likely good for detecting potential malicious web pages. Finally, scoring algorithm can make a trade-off between number of detected potential malicious web pages and false negative rate (missing attack). The key idea to propose scoring mechanism is to reduce number of suspicious web pages which need to be inspected by detection devices or experts, but not missing any attack.

3.1 Feature Selection

The first step on feature selection is to identify potential malicious features which can distinguish between benign web pages and malicious one. By analysing the selected common malicious web pages, we find that there are three main groups of malicious contents of web pages as follows:

- Foreign contents are malicious contents which are loaded from outside along with suspicious web pages. These contents can be loaded with suspicious web pages by some of malicious HTML tags such as frame, iframe, image source… Iframe is especially known as very common method to load outside malicious web pages along with suspicious one (Provos, Mavrommatis, Abu and Monrose). In almost all of cases, foreign malicious contents are resulted from compromises or uncontrolled third-party contents such as advertising and site hit counters.
- Script contents are known as the most common malicious contents of malicious web pages. In almost all of cases, script codes are used for two main purposes: delivering and hiding malicious codes by obfuscations. We identify some of potential malicious features from scripts which could distinguish between benign web pages and malicious web pages, such as script size, string

size, word size, argument size, character distribution…

- Exploit code contents are the core contents of malicious web pages. They are target specific vulnerabilities in web browsers, plug-ins or operating systems. Some of HTML tags known as delivery of potential malicious codes are applet, object, embed… However, there are rarely malicious codes found in this direct form. In almost cases, exploit codes are encoded in scripts with obfuscations to hide from detection devices.

Feature	Group 1: Foreign Contents
1	Number of redirection
2	Number of iframe and frame tag
3	Number of external link in iframe and frame tag
4	Iframe and frame link length: Median
5	Ratio of vowel character in iframe and frame link: Minimum
6	Ratio of special character in iframe and frame link: Minimum
7	Number of external links (except iframe and frame)
8	Other link length: Minimum
	Group 2: Script Contents
9	Number of scripts
10	Number of script lines
11	Number of script word
12	Ratio of special character in scripts
13	Script length: Minimum
14	Script line length: Minimum
15	Script string length: Maximum
16	Script word length: Minimum
17	Script function argument length: Minimum
	Group 3: Exploit Contents
18	Number of objects
19	Number of applets
20	Object link length: Maximum
21	Ratio of special character in object links
22	Ratio of vowel character in object links
23	Number of object attributes: Median
24	Applet link length: Minimum
25	Ratio of special character in applet link
26	Ratio of vowel character in applet link

Table 1: Appropriate Features for Identifying Potential Malicious Web Pages

According to our analysis, we select 52 potential features from these main malicious contents. If a feature appears more than once, we use four values to measure it at the first sight: minimum, maximum, mean and median. However, only one measured value for each feature is chosen for scoring algorithm.

Secondly, we use information gain as a measurement method to choose high valuable features only. Information gain for an attribute a is defined as follows:

$$\mathrm{IG}(S, a) = \mathrm{Entropy}(S) - \sum_{v \in a} \frac{|S_v|}{|S|} * \mathrm{Entropy}(S_v)$$

Where S is collection of instances, S_v is a subset of S with relevant value v of attribute a. The greater information gain an observed attribute obtains, the higher value it contributes to the process to identify malicious web pages. The training dataset which is used to calculate information gain must have both malicious and benign instances. There are 26 potential features selected based on information gain (Table 1).

3.2 Scoring Mechanism

Our scoring algorithm works based on the concept of standard score which measure how many standard deviations a value of observed attribute is far from the mean (Carroll and Carroll 2002). Each instance has three types of scores based on three groups of contents of web pages: Foreign content score, script content score and exploit content score.

A group score of instance x is calculated as follows:

$$\mathbf{GS}_{g \in G}(x) = \sum_{a \in g} \frac{|x_a - \mu_a|}{\delta_a}$$

Where g is an attribute group which can be foreign content group, script content group or exploit content group; a is an attribute of g; x_a is value of attribute a of instance x; $\delta_a$is a standard deviation of attribute a which is estimated during training a set of benign instances; μ_a is mean of attribute a which is estimated during training a set of benign instances.

The greater score an instance x has in each group, the more likely it is classified as potential malicious class. If $T_g$is chosen as a threshold for content group g in order to identify potential malicious instances, the rule of classification is as follows:

$$x = \begin{cases} \text{potentialy malicious if } \exists g \in G: GS_g(x) > T_g \\ \text{otherwise,} \quad x \text{ is benign} \end{cases}$$

Any page will be classified as potential malicious that has a group score greater than the threshold value for that group.

4 Data Collection

To get dataset for our experiments, we firstly collect candidate web pages which include both malicious and benign one. To collect benign web pages, we collect hot search terms from Google Search Engine (Google 2010) and then feed these search terms to Yahoo API websearch (Yahoo 2010) to get top 10 URLs from the search results. In addition, we collect malicious web pages from some of common public announced malware and exploit websites like Blade-defender.org, Clean-mx.de, Paretologic.com, Malwaredomainlist.com. These selected web pages are verified by our Capture-HPC, a high interaction client honeypot (Seifert and Steenson 2009).

Secondly, we create a low interaction client honeypot which interacts with web servers to request for the selected web pages. The HTTP responses from web servers are extracted based on the attributes and their potential values described on Table 1. We totally collect 33646 instances of web pages, including 33422 instances of benign web pages and 224 instances of malicious one.

5 Experiments

To evaluate our scoring mechanism, we divide dataset into two subsets as follows:

- Training dataset consists of 20,000 benign instances and it is used for training scoring algorithm to calculate mean and standard deviation for each attribute.

- Testing dataset contains 13,646 instances with 13,422 benign instances and 224 malicious one. This dataset is used to test the scoring mechanism.

The experiment is carried in three steps. Firstly, training dataset is fed into our scoring mechanism in order to calculate some statistic values such as mean, standard deviation. Secondly, we calculate group scores for each instance in the testing dataset. Each instance has three types of scores: foreign content score, script content score and exploit content score. Finally, we adjust threshold score values in each group in order to find the relationship between false negative rate and the number of identified potential web pages.

6 Results

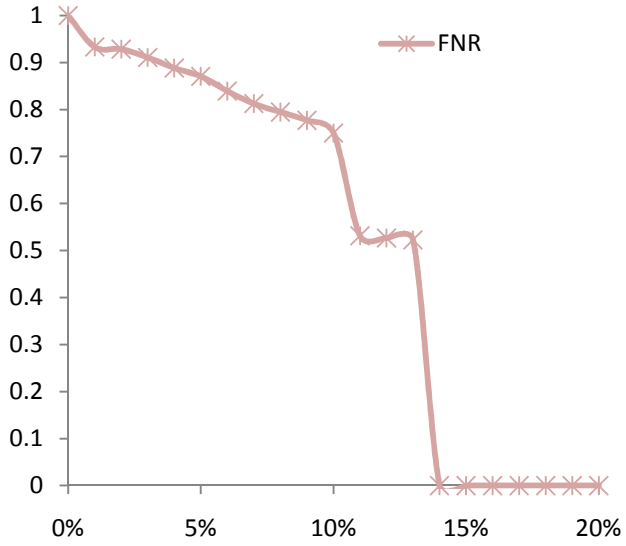

Figure 2: The relationship between false negative rate and number of potential malicious web pages.

We use 20,000 instances of benign web pages to train our scoring algorithm and 13646 instances of malicious and benign web pages for testing. To find out the relationship between false negative rate and the number of identified potential malicious web pages, we adjust the value of score threshold in each group and calculate number of negative. The threshold start from the maximum value of each group score, and then reduce to the value corresponded to the percentage of potential malicious web pages. Figure 1 shows the relationship between the number of identified potential malicious web pages and false negative rate. When number of potential malicious increases, false negative rate decreases. Our aim is to minimize the false negative rate, in Figure 1 this is achieved when number of potential malicious web pages reaches 14% of the total number of instances in the testing dataset. In the other word, we can reduce 86% number of suspicious web pages without missing attacks.

7 Conclusion

This paper presents a scoring mechanism to estimate maliciousness of web pages in order to reduce the number of suspicious web pages which need to be analysed by a secondary mechanism such as high-interaction honeypot. The advantages of this scoring mechanism are discussed as using lightweight static features, capability to make trade-off between number of potential malicious web pages and false negative rate (that is, missing an attack).

Three main groups of malicious contents are identified in this paper. Based on these contents groups, we extracted 52 potential features from both malicious and benign web pages. Information gain is used in order to identify 26 potential features. Each web page has three scores corresponded to three contents groups. Thresholds are chosen for each content group. A web page is classified as potential malicious web pages if it has at least one group score higher than threshold.

The proposed scoring mechanism is initially tested on 13,646 instances with 224 malicious web pages. The result shows that there is capability to make trade-off between number of potential malicious web pages and missing attacks.

This work however has some limitations, which are identified and required for future works. Firstly, a limited number of malicious samples (224 instances) may not present all statistical characteristics of malicious web pages. Secondly, only information gain feature selection method is used in the feature selection process. Other feature selection methods could be investigated in order to have a good comparison. Thirdly, there are three contents groups with three thresholds but the relationship between them in order to form the overall score with only one overall threshold has not identified yet.

References

Adam, A. N. & Meledath, D. (2008): Security in web 2.0 application development. *Proceedings of the 10th International Conference on Information Integration and Web-based Applications \& Services, Linz, Austria*,ACM.

Alme, C. (2008) Web Browsers: An Emerging Platform Under Attack. MCAfee.

Barth, A., Jackson, C. & Mitchell, J. (2009): Securing frame communication in browsers. *Commun. ACM*, 52**:** 83-91.

Bin, L., Jianjun, H., Fang, L., Dawei, W., Daxiang, D. & Zhaohui, L. (2009): Malicious Web Pages Detection Based on Abnormal Visibility Recognition. *Proc. E-Business and Information System Security, 2009. EBISS '09. International Conference on***:** 1-5.

Carroll, S. R. & Carroll, D. J. (2002): *Statistics made simple for school leaders: data-driven decision making*, Scarecrow Press.

Charles, R., John, D., Helen, J. W., Opher, D. & Saher, E. (2007): BrowserShield: Vulnerability-driven filtering of dynamic HTML. *ACM Trans. Web*, 1**:** 11.

Chia-Mei, C., Wan-Yi, T. & Hsiao-Chung, L. (2009): Anomaly Behavior Analysis for Web Page Inspection. *Proc. Networks and Communications, 2009. NETCOM '09. First International Conference on***:** 358-363.

Cova, M., Kruegel, C. & Vigna, G. (2010): Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code. *Proc. WWW2010, Raleigh NC, USA.*

Gargoyle Html Unit, http://htmlunit.sourceforge.net/, Accessed 02/05/2010.

Garrett, B., Travis, H., Micheal, I., Atul, P. & Kevin, B. (2008): Social networks and context-aware spam. *Proceedings of the ACM 2008 conference on Computer supported cooperative work, San Diego, CA, USA*,ACM.

Gollmann, D. (2008): Securing Web applications. *Information Security Technical Report,* 13**:** 1-9.

Google (2010): Google Trends, http://www.google.com/trends/hottrends, Accessed 08/03/2010.

Gyongyi, Z. & Garcia-Molina, H. (2004) Web spam taxonomy. California, Stanford University.

Hou, Y.-T., Chang, Y., Chen, T., Laih, C.-S. & Chen, C.-M. (2009): Malicious web content detection by machine learning. *Expert Systems with Applications,* In Press, Corrected Proof.

Ikinci, A., Holz, T. & Freiling, F. (2008): Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients. *Proc. Sicherheit, Saarbruecken*.

Jianwei, Z., Yonglin, Z., Jinpeng, G., Minghua, W., Xulu, J., Weimin, S. & Yuejin, D. (2007) Malicious websites on the Chinese web: overview and case study. Beijing, Peking University.

Jose, M., Ralf, S., Helen, J. W. & Yi-Min, W. (2007): A Systematic Approach to Uncover Security Flaws in GUI Logic. *Proceedings of the 2007 IEEE Symposium on Security and Privacy*,IEEE Computer Society.

Keats, S. & Koshy, E. (2009) The Web's Most Dangerous Search Term. McAfee.

Lawton, G. (2007): Web 2.0 Creates Security Challenges. *Computer,* 40**:** 13-16.

Liu, P. & Wang, X. (2009): Identification of Malicious Web Pages by Inductive Learning. *Proceedings of the International Conference on Web Information Systems and Mining, Shanghai, China*,Springer-Verlag.

Ma, J., Saul, L. K., Savage, S. & Voelker, G. M. (2009a): Beyond blacklists: learning to detect malicious web sites from suspicious URLs. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France*,ACM.

Ma, J., Saul, L. K., Savage, S. & Voelker, G. M. (2009b): Identifying suspicious URLs: an application of large-scale online learning. *Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, Quebec, Canada*,ACM.

Mehdi, J. (2007): Some Trends in Web Application Development. *2007 Future of Software Engineering*,IEEE Computer Society.

Microsoft (2009) Microsoft Security Intelligence Report. *January through June 2009.*

Moshchuk, E., Bragin, T., Gribble, S. D. & Levy, H. M. (2006): A crawler-based study of spyware on the Web.

Niels, P., Moheeb Abu, R. & Panayiotis, M. (2009): Cybercrime 2.0: When the Cloud Turns Dark. *Queue,* 7**:** 46-47.

Patsakis, C., Asthenidis, A. & Chatzidimitriou, A. (2009): Social Networks as an Attack Platform: Facebook Case Study. *Proc. Networks, 2009. ICN '09. Eighth International Conference on***:** 245-247.

Provos, N., Mavrommatis, P., Abu, M. & Monrose, R. F. All your iframes point to us. *Google Inc,* 2008.

Provos, N., McNamee, D., Mavrommatis, P., Wang, K. & Modadugu, A. (2007): The Ghost In The Browser: Analysis of Web-based Malware. *Proc. Proceedings of the first USENIX workshop on hot topics in Botnets*.

ScanSafe (2009) Annual Global Threat Report. *Trends for January 2008 - December 2008.*

ScienceDirect (2008): Most malicious web sites are hacked. *Network Security,* 2008**:** 1-2.

Seifert, C. (2007a) Improving Detection Accuracy and Speed with Hybrid Client Honeypots,. Wellington, Victoria University of Wellington.

Seifert, C. (2007b): Know Your Enemy: Behind the Scenes of Malicious Web Servers. *The Honeynet Project*.

Seifert, C. & Steenson, R. (2009): Capture-HPC, https://projects.honeynet.org/capture-hpc/, Accessed 22/02/2010.

Seifert, C., Steenson, R., Holz, T., Yuan, B. & Davis, M. A. (2007): Know Your Enemy: Malicious Web Servers. *The Honeynet Project.*

Seifert, C., Welch, I. & Komisarczuk, P. (2007): HoneyC - The Low-Interaction Client Honeypot. *Proc. NZCSRSC, Hamilton*.

Seifert, C., Welch, I. & Komisarczuk, P. (2008): Identification of Malicious Web Pages with Static Heuristics. *Proc. Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian***:** 91-96.

Shih-Fen, L., Yung-Tsung, H., Chia-Mei, C., Bingchiang, J. & Chi-Sung, L. (2008): Malicious Webpage Detection by Semantics-Aware Reasoning. *Proc. Intelligent Systems Design and Applications, 2008. ISDA '08. Eighth International Conference on,* 1**:** 115-120.

Sophos (2009) Security threat report: 2009. Sophos.

Symantic (April 2009) Security Threat Report - Trend for 2008. *Volume XIV.*

Wang, Y.-M., Beck, D., Jiang, X. & Roussev, R. (2006): Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites that Exploit Browser Vulnerabilities. *IN NDSS*.

Websense (2008) State of Internet Security. *Q1-Q2.* Websense Security Labs.

Websense (2009) State of Internet Security. *Q3-Q4.* Websense Security Labs.

Xiaoyan, S., Yang, W., Jie, R., Yuefei, Z. & Shengli, L. (2008): Collecting Internet Malware Based on Client-side Honeypot. *Proc. Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for***:** 1493-1498.

Yahoo (2010): Web search document for Yahoo!, http://developer.yahoo.com/search/web/V1/webSearch.html, Accessed 08/03/2010.