AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Characterizing A Malicious Web Page

Abubakr Khamis, Baharum Baharudin, Low Jung

Computer & Information Science Department, University Technology Pertonas, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia.

ARTICLE INFO

Article history:
Received 26 January 2014
Received in revised form 10 March 2014
Accepted 12 March 2014
Available online 5 April 2014

Keywords:
Malicious, Benign, Feature, Detector, Page Content, Script, Learning, URL lexical.

ABSTRACT

Background: The web page has become a great environment for those with malicious intent to perform some unwanted activities or launch attack. The attackers normally embed the malicious contents in the compromised web page or inject links to fake web pages they have created. Unhappily attacks are becoming more sophisticated by using complicated techniques to evade detection. The victim's system can be infected by just a single visit to a web page and the attack sometimes can be silent and unnoticeable. Often, the attacker focuses on a web site that has become the centre of attention, and then exploits the system's vulnerabilities and launches the attack. **Objective:** To provide a lightweight framework for malicious web page detection in order to reduce the threats of the web-based attacks and mitigate the risk. **Results:** Comparing with different approaches, our experiment with real world data set has shown a significant result, and our approach is able to reach 97% detection**. Conclusion:** We have proposed a framework for identifying a malicious web page as either benign or malicious using supervised machine learning techniques. The malicious web page has been characterized by a vector of discriminative features, and based on HTTP response we have collected two groups of features, the URL string lexical features and the page content features. When comparing with other works, the proposed framework effectively mitigates threats and less execution overhead.

To Cite This Article: Abubakr Khamis, Baharum Baharudin, Low Jung., Characterizing A Malicious Web Page. ***Aust. J. Basic & Appl. Sci.,*** *8(3): 69-76, 2014*

INTRODUCTION

The internet has become an essential part in our daily life. It is the base of banking transactions, shopping, entertainment, resource sharing, news, and social networking. The growth of the web has also rewarded cyber criminals to take the chance and conducts their illegal actions. With this growth, the design and the use of the malware scenario has also changed. Nowadays damaging the machines is not the target any more. The malware tactics are stealthier and polymorphic to avoid the detection mechanism. The majority of the web malware tactics is either to steal the user's private data, such as credit card details and passwords, or force the victim system to join a malware distribution network. One of the most common methods for spreading malware today is through web pages, which exploit the vulnerabilities in web browsers, web applications, and operating systems, and trying to gain control of a victim's machine. Victims' machine is then used as a host to various malicious activities like: heap spray, botnet, key loggers, sending spam emails, or distributed denial of service.

The attack occurs when a user visits a suspected website. Therefore the attacker focuses on a web site that has become the centre of attention, and then exploits the vulnerabilities and launches the attack silently without the user consent, and sometimes with your consent leveraging some social engineering scenarios. Managing/ handling Web page attack is a challenge, and necessitates a careful understanding of the details and the behavior. To enrich the web applications, browser vendors supported active client-side content with many techniques. Among these techniques is JavaScript, which is most widely used as a primary component of active and dynamic web content, and at the same time providing a great chance for the web exploits. Other techniques such as PHP language, adobe flash, and visual basic are also provides opportunities for these exploits. All these techniques have in common the capability of downloading and executing code from the Internet [8]. In addition most browsers have a feature of plug-ins, which allow third parties to extend the functionality of the browser, and this may add a chance for exploit. And all plug-ins commonly have higher privileges to run than the embedded script code unless there are explicit restrictions on that.

Although several solutions have been proposed to fight malicious software such as network intrusions, malware detecting and preventing, malicious email attachments, web site exploits has not received much

Corresponding Author: Abubakr Khamis, Computer & Information Science Department, University Technology Pertonas, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia.

attention so far. This work is continuing to add some values to the malware field combat by mitigating some threats, and improve performance by enhancing the detection rate.

Related Work:

Existing detection approaches can be classified into: signature-based, behavior monitoring, and machine learning. And the feature analysis method followed by them can be grouped in three types: URL and domain analysis, page content analysis, behavior analysis.

URL and Domain Analysis:

The universal resource locator (URL) textual form was used in malicious web page detection methods and some distinguishable URL criteria have been used to give a sign of the potential maliciousness based on URL lexical and host-based features using on line learning classifiers, and their experiment has shown some effective results. Yoshiro Fukushima et.al (Fukushima, Hori *et al*. 2011) analyzed characteristics of malicious Web sites by their domain information. They studied the IP address block, IP address, domain, and registrar, and evaluated the reputations of IP address blocks and registrars used by attackers. Then, they proposed a blacklisting scheme derived from the combination of IP address block and registrars with low reputation that is intensively used by attackers. A hierarchical structure graph is constructed using the extracted information. Zhang et.al (Zhang, Seifert *et al*. 2011) provided a model for determining a malware distribution networks from the secondary URLs, and redirect chains recorded by a high-interaction client honeypot. The method also encompasses a drive-by download detection mechanism based on a set of regular expression-based signatures.

Page Content Analysis:

HTML tags and JavaScript have been used a lot to launch attacks, especially the latter which is useful in embedding tasks into a web page. It enhances the capabilities of HTML by some wonderful features such as: Higher-order functions, dynamic typing and flexible object model. Cove et.al (Cova, Kruegel *et al*. 2010) have introduced an approach for detection and analysis of malicious JavaScript code based on assigning probability to a feature that reflects the likelihood that a given feature value occurs. The approach combines anomaly detection with emulation to identify malicious JavaScript code. Byung-Ik Kim et.al (Kim, Im *et al*. 2011) proposed a hybrid JavaScript obfuscation strength examination system to detect malicious contents of a web page. The system used the characteristics of obfuscated JavaScript instead of the analysis that focuses on the meaning of strings. The study selected N-Gram, entropy and word size as the primary items to be scrutinized. The primary criteria that are scrutinized include string length, density, frequency of the particular function, frequency of special characters, and entropy value. These criteria are used to measure the obfuscation strength of an obfuscated website. Yuan-TsungHouet.al (Hou, Chang *et al*. 2010) proposed a malicious web page detection model using the boosted decision tree learning algorithm. The classification algorithm used the dynamic HTML features combined with some Java script native functions. Le et.al (Van Lam Le, Gao *et al*. 2011) presented a scoring mechanism for potential malicious pages, which uses static features. The mechanism is used as a filter that assists in reducing the number of suspicious web pages that might need extra analysis by other mechanisms that require loading and interpretation of the web pages. Van Lam Le et.al (Van Lam Le, Gao *et al*. 2011) have introduced a two-stage classification model to detect malicious web page based on information gain values. In the first stage, the URL is inspected to extract the potential static feature, while in the second one only potential malicious page that has been identified in the first stage is scanned to extract run-time features. They have selected 52 potential features from the malicious contents, and they used four values to measure it at first sight: minimum, maximum, mean and median. Mario Heiderichet.al (Heiderich, Frosch *et al*. 2011) introduced an approach that performed lightweight instrumentation of JavaScript, detecting attacks against the HTML document object model (DOM) tree, and also able to mitigate improved PHoneyC and LibemuShellCode detection to collect malware scenario, and the result outperformed Google's safe browsing. S. Chitraet.al (Chitra, Jayanthan *et al*. 2012) have provided an approach for detecting a malicious web page using genetically evolved fuzzy rules. The rules are filtered by SVM and the result is stored as a symbolic knowledge representation. They used a symbolic and non-symbolic knowledge-base in their approach, and only 21 optimal features of a web page have been investigated in a virtualized environment. The approach has shown good result. Eshete et.al (Eshete, Villafiorita *et al*. 2012) presented of a web page. Ma et.al (Ma, Saul *et al*. 2011) introduced malicious web site detection model based on URL features. The model used three different online classifiers algorithms: Naïve Bayesian, SVM and logistic regression. The model categorized the features of URLs as being either lexical or host-based features. Zhang et.al (Zhang, Ding *et al*. 2011) provided a study on detecting malicious web pages BINSPECT, that uses static analysis and browser emulation and apply supervised learning techniques in detecting malicious web pages by proposing three feature groups, URL, page-source and Social-Reputation Features. BINSPECT achieved about 97% accuracy with low false signals with quite execution overhead.

Behavior Analysis:

Xuet.al (Xu, Yao *et al*. 2011) proposed a model for detecting the infection that is delivered through vulnerable applications and web browser. The detection engine depended on the dependency between a user's action and system's event. Two components were used to record user behavior; one at kernel level, and another as a Firefox extension. Hsu, F.H., *et al* (Hsu, Tso *et al*. 2011) have proposed a runtime, behavior-based solution, to protect the system against drive-by-download attacks. The approach utilized the browser helper object (BHO) mechanism of Windows operating system to implement the framework on internet explorer 7.0. The experimental results have shown low performance overhead without false positives and false negative rates.

As a conclusion, the URL and domain analysis method is a very useful idea, but the limitation of this method is that the malicious URL does not mean that the corresponding page holds malicious contents and vice versa. On the other hand the drawback of the JavaScript-Based method is that the attack might be launched through Obfuscated mechanism, while the page content-based method faces the challenge of the continuous changing of the attack features, which require continuous update.

MATERIALS AND METHODS

The model mainly composed of three components, Feature Extractor, Learning and Model Selection and the Detector as shown in Fig. 1. It starts with the feature extraction process via feature extractor component, and then the URL lexical features and page content features are collected together as one group and send to the detector. The model selector is developed for the training purpose to produce the final classification model.

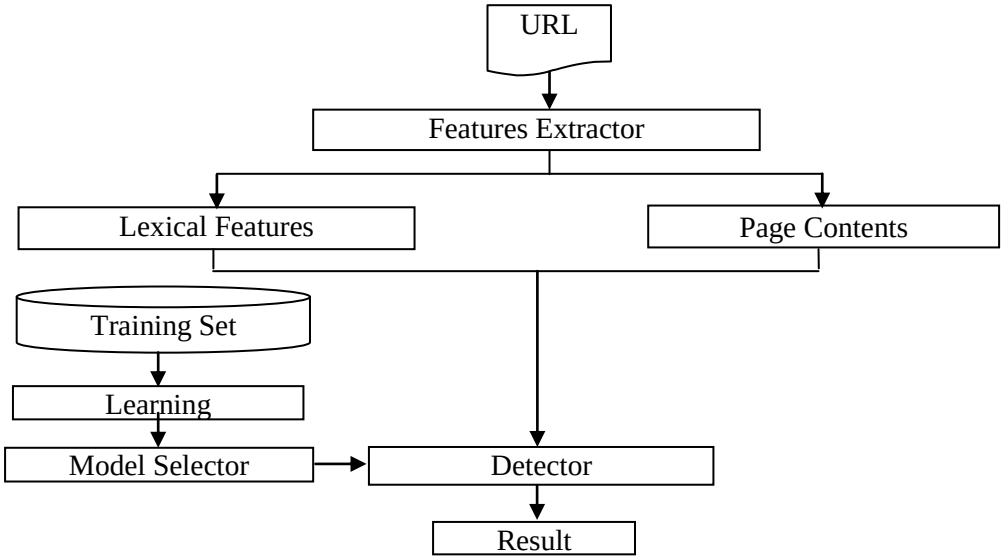

Fig. 1: the general structure of the approach

Feature Extractor:

This is the first component, which is responsible for capturing the web page features. It consists of: Jericho3.3 HTML parser, JavaScript engine and URL lexical class using Java. This part is important mainly because, it is essential to collect the candidate features and prepare the data set.

Learning and Model Selection:

It necessitates the data set for learning to build the model, and according to the performance, the model selector selects the learning model of the best result among different machine learning algorithms.

Detector:

The detector is using the final classification model generated from the Learning and Model Selector component.

Malicious Web Page Characterization:

There are a lot of candidate features that can support the degree of maliciousness of a web page and considered as the target of intruders. In our study we have selected 39 features.

URL Lexical Features:

The uniform resource locator (URL) textual form was used in malicious web page detection methods (Ma, Saul *et al*. 2011)-(Zhang, Ding *et al*. 2011) , and some distinguishable URL criteria have been used to give a sign of the potential maliciousness of a web page. The lexical features of the URL are not the page content that it refers to, but they are only the partitions of the URL string. We have studied the case deeply and determined some characteristics which differentiate the good reputation web pages from the bad ones. We have explored around 18 different features and only 10 most significant features have been selected as shown in Table 1.

Table 1: The URL lexical Features

No	Features	Type
1	The length of host name	integer
2	The number of delimiters in host name (-,_)	integer
3	The length of file name	integer
4	The number of digits in host name	integer
5	The total length of the URL	integer
6	The number of arguments	integer
7	The number of directories	integer
8	The number of digits in path	integer
9	The length of directories	integer
10	The maximum length of tokens	integer

Page-contents Features:

Some HTML tags in general have been known as a common method used to load the malware from the outside. These tags include: body, iframe, img, meta, applet, frameset, style, layer, ilayer, embed, script, form, object, link, and normal hyperlink. And some tags are commonly used and targeted by intruders to deliver foreign malicious contents. On the other hand JavaScript is a useful language in embedding tasks into a web page, but when some functions are misused it's so harmful such as: escape, unscape, eval, exec, and unbound, including its ActiveX objects. In addition, the code obfuscation plays a great role in hiding the malware. Therefore we have taken under consideration some features like: obfuscation attempt including encoding and mathematical functions, length and number of scripts, harmful keyword, internal and external scripts, and number of methods. On the other hand implementing security features in browsers using cryptographic algorithm is important for protecting a user credential from man in the middle attack, but still there is a resistance from browser JavaScript to cryptography. The total number of features that we have selected is 29.

Table 2 shows the selected page content features.

Table 2: The Page Content Features

No	Features	Type
1	The number of total links	integer
2	The number of external links	integer
3	The number of encoded links	integer
4	The number of Iframe	integer
5	The number external Iframe source	integer
6	The abnormal visibility fingerprint	Boolean
7	The number Applets	integer
8	The number of Objects	integer
9	The number of Scripts	integer
10	The number of PHP dangerous functions	integer
11	The number of harmful links	integer
12	The number of forms	integer
13	The number of external form action	integer
14	The length of the script	integer
16	The number of native functions	integer
17	The number of redirects	integer
18	The number of methods	integer
19	The number harmful keywords	integer
20	The number of interaction events	integer
21	The use of encoding methods	Boolean
22	The number of encoded links	integer
23	The existence of obfuscation signs	Boolean
24	The number of external Script source	integer
25	The number of forms predefined values	integer
26	The number of documents declaration	integer
27	The number of XML declaration	integer
28	The number of new TAGs declaration	integer
29	The maximum length of links	integer

The Classification Model:

Support Vector Machine (SVM):

It is a classification method introduced in 1992 by Boser, Guyon, and Vapnik (Lee and Huang 2007). The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create non-linear classifiers by applying the kernel trick to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be non-linear in the original input space. In dealing with large data sets, the support vector machine (SVM) was proposed for the practical objective to overcome some computational difficulties as well as to reduce the complexity (Lee and Huang 2007). The SVM is widely used due to its ability to deal with high-dimensional, flexibility, and high accuracy.

Decision Tree (DT):

It is a machine learning classifier based on the tree structure. Each node in the tree is associated with a particular feature, and the edges from the node separate the data based on the value of the feature. Each leaf node binds to a class in the classifier model. By tracking the nodes from the root of the tree based on the feature values, one can get the predicted class of it. The training data is the key point for the information gain (IG) of the feature and feature selection policy (Kohavi and Quinlan).

K nearest-neighbor (KNN):

It is a simple algorithm for predicting a class of an example. This classifier is supervised learning based on the distance of the example. The training stage simply stores all training examples with their labels. To predict the class for a new test example, first it computes its distance to every training example and then, keeps the k closest training examples, where $k \geq 1$. Finally, it looks for the label that is most common among these examples. This label is assigned to this test example as the predicted result (Elkan 2007).

Artificial Neural Network (ANN):

It is a solution inspired by biological neural networks. An artificial neuron is a computational model receives signals through synapses located on the dendrites. When the signals received are exceeded a certain threshold value, the neuron is activated and emits a signal though the axon. The emitted signal might be sent to another synapse, or activate other neurons (Gershenson 2003). ANN can be viewed as directed graphs with weights, which artificial neurons are nodes and directed edges represent the connections between neuron outputs and neuron inputs. According to the connection pattern, ANNs can be put into two groups: feed-forward networks, and feedback networks (Jain, Mao *et al*. 1996).

RESULT AND DISCUSSION

In this section, first we detail data source and sets, and experiments scenario. Second we evaluate our approach targeting the accuracy and feature significance, and lastly we compare the method with the other previous work and discuss the method.

Data Set and Data Source:

To collect our experimental dataset, we have set up a virtual machine with a browser emulator. The dataset is composed of benign and malicious instances; the benign set is collected via web search and Alexa website ranking verified by Google safe browsing (Van Lam Le, Gao *et al*. 2011). The malicious set is collected from some of the common public announced malware and exploited websites (Tao, Shunzheng *et al*. 2010)-(Zhang, Ding *et al*. 2011) such as malwaredomainlist.com, StopBadWare, alwaredomains.com, PhishTank.com and malwareurl.com.

The collected dataset is divided into two groups training and test. The training set consists of 44000instances (32000 benign, 12000 malicious), the test set consists of 3765 instances (1386 benign, 2379 malicious).

Experimental Procedure:

The experiment is carried out in four steps. Firstly, we prepared the lists of malicious and benign URL separately. Secondly, we extracted the page content and the URL lexical features and put them into two groups training and test group. Thirdly, the training set is send to the classification algorithm to generate the model. And finally, the generated model is tested using the test data set.

The Evaluation:

The accuracy is defined as the ratio of correctly identified examples over all examples in the test set. The detector was tested using 3765 instances consisting of 2379 malicious and 1386 benign instances. Where benign instances are negative example and malicious ones are positive. The evaluation involved four different classifiers based on two features groups and the performance is shown in Table 3.

Table 3: The Accuracy with the new features

Algorithm	Recall (%)	Specificity (%)	Accuracy (%)
SVM	96.85	94.95	96.15
KNN	85.16	97.19	89.59
DT	93.91	98.85	95.72
ANN	95.75	98.05	96.60

The ROC curve Fig. 2 shows that the artificial neural network achieved the best performance in both true positive rate and false positive rate. The desired false positive rate is above0.01 and below 0.1.

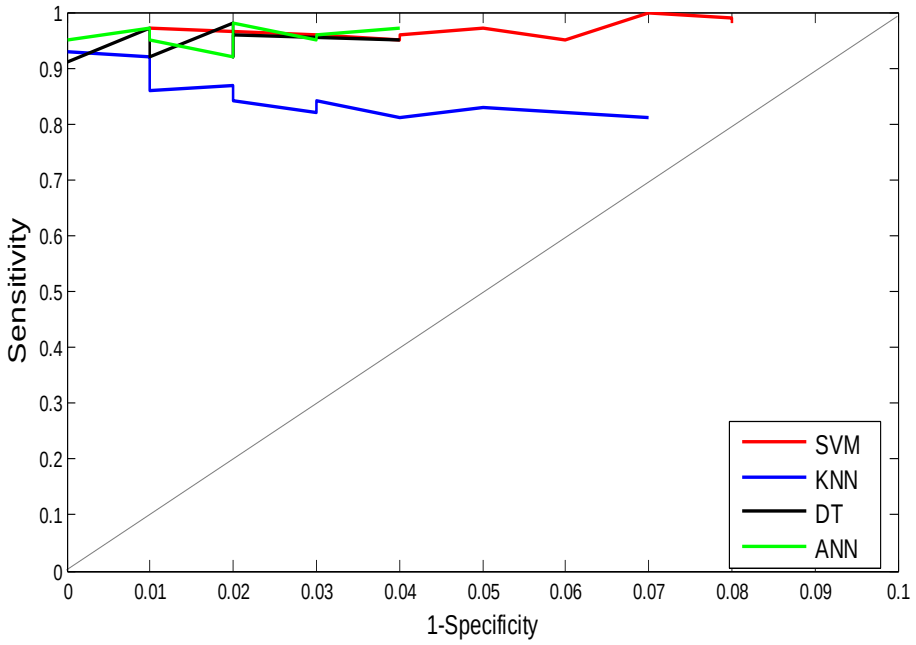

Fig. 2: the ROC Curve

The Features Significance:

In this study we have analyzed the characteristics of the malicious web pages and presented the most relevant features for the attack scenario, in addition to the effectiveness of these features, also they remain resilient against possible anticipated future attack. The experimental results show that the URL lexical features can give distinguishing values for the malicious web pages and raise up the true positive rate. As well as the JavaScript features, it has a significant effect on the true positive rate, which leads to an improvement in the accuracy. The HTML features are easier to be used by the attacker and easier to be captured by detection engines, therefore the attackers usually encode the target code to circumvent the signature-based detection engine, but if most of the code is encoded, some native functions are still needed in a plain text without encoding to render the page. Moreover the HTML features can also reduce the false positive rate. Based on the results, the combination of these feature groups has shown the higher true positive rate and lower false positive, which play a great role in the accuracy improvement. Table 4 shows how the accuracy went down when we selected the reused features.

Table 4: The Accuracy with existing features

Algorithm	Recall (%)	Specificity (%)	Accuracy (%)
SVM	91.38	90.98	91.24
KNN	71.54	97.69	81.17
DT	87.73	98.70	91.77
ANN	85.62	97.11	89.85

Comparison with Related Work:

In the table below study, we compare with other related work using three points, the number of features, the methods, and the detection accuracy as shown in **Table 5**.

Table 5: The Accuracy of related work

Work	Number Features	Method	Accuracy (%)
(Chitra, Jayanthan *et al*. 2012)	21	Dynamic	95.6
(Tao, Shunzheng *et al*. 2010)	23	Static	92.2
(ESHETE, VILLAFIORITA *ET AL*. 2012)	39	Hybrid	97.0
(Van Lam Le, Gao *et al*. 2011)	26	Hybrid	TP=86.0
This work	39	Static	96.6

In addition, we compare with other related work using three points, the used features, feature analysis methods applied, and the detection effectiveness.

Ma et.al (Ma, Saul *et al*. 2011) introduced a static technique based on URL lexical and host features using supervised learning techniques, the model was able to achieve 99% accuracy with a low false positive signal. However, they depend only on URL and host-based feature without further investigation on the page content features. In our model, we applied further analysis on page content feature by using HTTP response, which added some capabilities to detect more attack attempts.

Zhang et.al (Zhang, Ding *et al*. 2011) provided study how detect malicious web pages based on URL lexical and host-based features using on line learning classifiers, their experiment has shown some effective results. In our case we used page content in addition to URL lexical in order to collect more attack features.

Wang Tao et.al (Tao, Shunzheng *et al*. 2010) proposed a framework that identified the malicious web page based on HTTP session header including both domains of request and response. The model studied the performance of three different classifiers; Naïve Bayesian, decision tree (C4.5) and support vector machine with 92.2% detection rate. Unlike our case, they used only session header instead of page contents. In our case, the page content feature is used, which added accurate performance and better result.

Eshete et.al (Eshete, Villafiorita *et al*. 2012) presented BINSPECT, based on three feature groups: URL, page-source and Social-Reputation Features, using hybrid static and dynamic analysis. Using supervised learning techniques BINSPECT achieved about 97% accuracy with low false signals with quite execution overhead, unlike our case, which is lightweight with no more execution complexity.

Chitra et.al (Chitra, Jayanthan *et al*. 2012) provided a predicate based algorithm for malicious web page based on 21 of page content features using genetically evolved fuzzy rules. The rules are filtered by SVM and the result is stored as a symbolic knowledge representation. The approach has shown good result and the detection accuracy reached 95.6%.unlike our case, we used more features with learning techniques and we have got better result.

Discussion:

The experiment has been conducted and ran without unusual inputs that would have introduced errors, all features were positive integer values. All abnormal cases due to the unavailable page of uncaptured redirection have been traced manfully from the log file. Because we were using mostly static technique some misclassification errors might arise up, we do not expect our algorithm to be as accurate as the dynamic collection of a web page features.

Although our algorithm provided useful features that can successfully distinguish malicious from benign web page. If a benign page has been incorrectly classified as malicious by our algorithm, that might be of the use of some JavaScript functions or key words considered as harmful during the development of the page and our algorithm might applied some penalties on it. But, if a malicious page has been classified as benign, that might be some dynamic characteristics this page have been missed. Therefore such dynamic pages are target for further analysis technique such as, dynamic mechanism or client honey client systems.

Furthermore the highly obfuscated JavaScript client-side code might trick our algorithm because obfuscated content has less consideration in our system. In addition, the malicious link within the benign content, attackers might mimic the benign content of some good reputable web pages to pass as benign, and then luring the user follow their malicious link, but even this malicious link might not pass when landing separately. In this context, user must be careful and cautious when browsing.

Conclusion and Future Work:

The idea behind doing this work is to provide a lightweight early assessment model in order to reduce the threats of the web-based attacks. In this paper we provide a mostly static identification mechanism for potential malicious web page using machine learning algorithms. The identification algorithm is based on two features group, the URL lexical features and the page content. Based on the HTTP response and without fully rendering the page we have collected the candidate lexical and page content features. And then we trained the classifiers and test the model. The experiment has shown the expected result, and as a comparison to similar work our model is able to reach 97% accuracy.

The drawback of this work is that we have used the partial rendering method based on HTTP response to identify the maliciousness of a web page. One of the solutions is to integrate a dynamic behavior monitoring unit to trace the client-server interactions. Moreover, the updated model could be integrated with the internet browser.

Our initial result in identifying malicious web page is efficient, but so some future endeavors' can be done to improve the general performance of the model. Moreover the model can be extended to include run-time feature collection unit. There are some challenges on the way such as: the change of the attack feature, the efficient algorithms, and effective comparative measures, therefore the future research may focus on tracing the attack vector features and paying a lot of attention on feature selecting methods, which might incorporate several disciplines to achieve the best performance.

REFRENCES

Chitra, S., K. Jayanthan, S. Preetha and R.N.U. Shankar, 2012. Predicate based Algorithm for Malicious Web Page Detection using Genetic Fuzzy Systems and Support Vector Machine. International Journal of Computer Applications., 40(10).

Cova, M., C. Kruegel and G. Vigna, 2010. Detection and analysis of drive-by-download attacks and malicious JavaScript code, ACM.

Elkan, C., 2007. Nearest neighbor classification. University of California–San Diego.

Eshete, B., A. Villafiorita and K. Weldemariam, 2012. BINSPECT: Holistic Analysis and Detection of Malicious Web Pages.

Fukushima, Y., Y. Hori and K. Sakurai, 2011. Proactive Blacklisting for Malicious Web Sites by Reputation Evaluation Based on Domain and IP Address Registration, IEEE.

Gershenson, C., 2003. Artificial neural networks for beginners. arXiv preprint cs/0308031.

Heiderich, M., T. Frosch and T. Holz, 2011. Iceshield: Detection and mitigation of malicious websites with a frozen dom, Springer.

Hou, Y.T., Y. Chang, T. Chen, C.S. Laih and C.M. Chen, 2010. Malicious web content detection by machine learning. Expert Systems with Applications., 37(1): 55-60.

Hsu, F.H., C.K. Tso, Y.C. Yeh, W.J. Wang and L.H. Chen, 2011. BrowserGuard: A Behavior-Based Solution to Drive-by-Download Attacks. Selected Areas in Communications, IEEE Journal on., 29(7): 1461-1468.

Jain, A.K., J. Mao and K.M. Mohiuddin, 1996. Artificial neural networks: A tutorial. Computer., 29(3): 31-44.

Kim, H.C.J.B.I., C.T. Im and H.C. Jung, 2011. Suspicious malicious web site detection with strength analysis of a javascript obfuscation. International Journal of Advanced Science and Technology., pp: 19-32.

Kohavi, R. and R. Quinlan, C5. 1.3 Decision Tree Discovery.

Lee, Y.J. and S.Y. Huang, 2007. Reduced support vector machines: A statistical theory. Neural Networks, IEEE Transactions on., 18(1): 1-13.

Ma, J., L.K. Saul S. Savage and G.M. Voelker, 2011. Learning to detect malicious URLs. ACM Transactions on Intelligent Systems and Technology (TIST) 2(3): 30.

Tao, W., Y. Shunzheng and X. Bailin, 2010. A Novel Framework for Learning to Detect Malicious Web Pages, Ieee.

Van Lam Le, I.W., X. Gao and P. Komisarczuk, 2011. Identification of Potential Malicious Web Pages.

Van Lam Le, I.W., X. Gao and P. Komisarczuk, 2011. Two-Stage Classification Model to Detect Malicious Web Pages, IEEE.

Xu, K., D. Yao, Q. Ma and A. Crowell, 2011. Detecting infection onset with behavior-based policies, IEEE.

Zhang, J., C. Seifert, J.W. Stokes and W. Lee, 2011. ARROW: GenerAting SignatuRes to Detect DRive-By DOWnloads, ACM.

Zhang, W., Y.X. Ding, Y. Tang and B. Zhao, 2011. Malicious web page detection based on on-line learning algorithm, IEEE.