Expert Systems with Applications 42 (2015) 1166–1177

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Comparisons of machine learning techniques for detecting malicious webpages

H.B. Kazemian *, S. Ahmed

Intelligent Systems Research Centre, School of Computing, London Metropolitan University, United Kingdom

ARTICLE INFO

Article history:
Available online 16 September 2014

Keywords:
K-Nearest Neighbor
Support Vector Machine
Naive Bayes
Affinity Propagation
K-Means
Supervised and unsupervised learning

ABSTRACT

This paper compares machine learning techniques for detecting malicious webpages. The conventional method of detecting malicious webpages is going through the black list and checking whether the webpages are listed. Black list is a list of webpages which are classified as malicious from a user's point of view. These black lists are created by trusted organizations and volunteers. They are then used by modern web browsers such as Chrome, Firefox, Internet Explorer, etc. However, black list is ineffective because of the frequent-changing nature of webpages, growing numbers of webpages that pose scalability issues and the crawlers' inability to visit intranet webpages that require computer operators to log in as authenticated users. In this paper therefore alternative and novel approaches are used by applying machine learning algorithms to detect malicious webpages. In this paper three supervised machine learning techniques such as K-Nearest Neighbor, Support Vector Machine and Naive Bayes Classifier, and two unsupervised machine learning techniques such as K-Means and Affinity Propagation are employed. Please note that K-Means and Affinity Propagation have not been applied to detection of malicious webpages by other researchers. All these machine learning techniques have been used to build predictive models to analyze large number of malicious and safe webpages. These webpages were downloaded by a concurrent crawler taking advantage of gevent. The webpages were parsed and various features such as content, URL and screenshot of webpages were extracted to feed into the machine learning models. Computer simulation results have produced an accuracy of up to 98% for the supervised techniques and silhouette coefficient of close to 0.96 for the unsupervised techniques. These predictive models have been applied in a practical context whereby Google Chrome can harness the predictive capabilities of the classifiers that have the advantages of both the lightweight and the heavyweight classifiers.

1. Introduction

Web security threats are increasing day by day (Facebook, 2010; Malware, 2011; Sood & Enbody, 2011). The open nature of the Internet allows malicious webpages to pose as 'safe webpages' and consequently some users are misled to think that these webpages are safe.

As the use and the speed of the Internet increased over the last two decades, web developers have increased the usage of images, JavaScript and other elements. The Google search engine is a clear example. At the beginning, it had very few elements. There are now more elements, graphics, stylesheets and the HTML specifications which have been added as time went on. Initially, the only way to create a webpage was by static HTML. JavaScript was then added for user interactivity. ActiveX, Silverlight, Java Applets, etc. were further added to include features. For example, ActiveX allowed browsers to host various executables which enabled users to read PDF and various other file formats such as Flash, DivX, etc. Web developers started using the integrated development environments that generated a considerable HTML markup language and this increased the HTML payload. The number of browsers increased and some of these browsers, especially Internet Explorer had their own quirks and needed more work from the developers. These factors raised the complexity of the webpages that led to potential increase in how webpages are 'adversely affected' and have become malicious.

Cross Site Scripting (XSS) injects malicious code from an unexpected source. These malicious codes can get hold of the cookies, browsing history and then send them over to the malicious webpage. Thus the user's privacy is jeopardized. There have been many attempts to prevent this sort of attacks (Lucca, Fasolino, Mastoianni, & Tramontana, 2004). XSS not only affects the user but also it affects the server. The webpage is used as the vehicle to transfer infections to multiple users. The malicious

* Corresponding author.

http://dx.doi.org/10.1016/j.eswa.2014.08.046
0957-4174/

code then executes in the user's browser. The problem has been intensified with the addition of scripting capabilities that did not exist at the beginning of the history of web browsing. With the addition of scripting capabilities, the users are benefitting with a better user experience but have become prone to these additional security problems. These scripts run on users' browsers. The web developer may build a webpage only using HTML, but an attacker can still inject scripts to make it susceptible to scripts. These scripts can then access the cookies that are used for authentication. The XSS vulnerability therefore affects the users and the webpages. Take for example, a user visits a webpage and decides to purchase a product. The user adds the items to the basket and would like to checkout. Then he fills in a form to register. Each of these users is uniquely identifiable by the webpage through the use of cookies. The criminal will be able to look at the cookies and impersonate the user and buy the products, without the knowledge of the user. By the time the user has realized the problem, the money has already been dispatched from the user's account.

Almost all HTML tags are wrapped by 'greater than' and 'less than'. To write the script tag, these two characters are needed. There are several combinations of characters that can be generated (Braganza, 2006). The combinations are quite vast and will likely to increase. The combinations of letters that generate the letters are dependent on browser version and the default language. Braganza (2006) states that the browser cannot be trusted because of these extensive possibilities and some precautions are required. To cleanse, data entered are encoded and data displayed are both decoded, this process is known as 'sanitization'. In terms of how the webpage is deployed to the user, the operations team have to make sure that the firewall or any other forms of preventative measures are kept up to date. Another security threat that is very difficult to detect is clickjacking. This is a relatively new threat that has become more prevalent through the advancement of modern browsers. The interesting thing about clickjacking is that it does not use security vulnerabilities, rather uses the browsers most common feature such as hyperlinks. The user is encouraged to click a link to a webpage. But this webpage has two webpages one is displayed to the user and the other one the malicious webpage which is hidden from the user (Hansen & Grossman, 2008). The hidden webpage executes the malicious code even though the user thinks that the information is on the right webpage. This technique is very hard to detect by inspecting the source code and there have not been many successful ways to prevent it from happening.

Drive-by-download occurs without the knowledge of a user and the downloaded file is used for malicious purposes. This malicious executable installs itself on users' computer. This is a very popular method that has been used by Harley and Bureau (2008) to spread malware infection on the Internet. There are three components in the attack, the web server, the browser and the malware. An attacker finds a web server to serve the malware. The user who visits a webpage hosted in this web server is then exploited by the webpage, and some code utilizes software loopholes to execute commands on the user's browser are injected. The web server subsequently provides the malware that is downloaded by the browser. As a result, the browser that is targeted will have a known vulnerability that the attacker will try to exploit. Internet Explorer had many instances of ActiveX loopholes that the attackers had used and are still using and Harley and Bureau (2008) have provided potential solutions. The first solution is to completely isolate the browser from the operating system so that the arbitrary codes are not at all executed on the browser. Another solution is for web crawlers to visit webpage and see whether they are hosting any malware content. But the attackers can avoid by using a URL that does not have a corresponding hyperlink. Crawlers by its nature only visit URLs that have a corresponding hyperlink.

Browsers these days use publicly available blacklists of malicious webpages. These blacklists are updated after a few days or even a month. These gaps allow for webpages to be affected while being unnoticed to the crawler. At this point, the users will also get affected, because the browser thinks the webpage to be secure, as it has never been in the blacklist. Take another scenario where a regular webpage may be hacked and injected with malicious code visible only to some users or a group of users of an organization or a country. The blacklists will not be able to 'blacklist' those either. Some crawlers do not validate the JavaScript code because the code is executed on the server and not in a browser. This allows client vulnerabilities to pass through easily. Even though some of the scripts which are assumed to be safe, these scripts can load malicious scripts remotely and then execute them on the computer. Some scripts create iFrames and then load external webpages that are malicious (Provos, Mavrommatis, Rajab, & Monrose, 2008). These external webpages then gets hold of the cookies and steal the identity. The users then browse this malicious webpage and get infected and are then easily tracked by remote users from somewhere else. The users also may run malicious executables without even knowing that the executables have already access to the system and are monitored from somewhere else. Webpages are the common victims to all these threats that have been described above. The features in a webpage can indicate whether it is malicious or not. Researchers have studied and analyzed a large number of features with or without machine learning techniques described below.

Kan and Thi (2005) carried out one of the first research work that utilized machine learning to detect malicious webpages. This work ignored webpage content and looked at URLs using a bag-of-words representation of tokens with annotations about the tokens' positions within the URL. A noteworthy result from Kan and Thi's research is that lexical features can achieve 95% accuracy of the page content features. Garera, Provos, Chew, and Rubin (2007)s work used logistic regression over 18 hand selected features to classify phishing URLs. The features include the presence of red flag key words in the URL, features based on Google's page ranking, and Google's webpage quality guidelines. Garera et al. achieved a classification accuracy of 97.3% over a set of 2500 URLs. Although this paper has similar motivation and methodology, it differs by trying to detect all types of malicious activities. This paper also uses more data for training and testing, as described in the subsequent sections. Spertus (1997) suggested an alternative approach and endeavored to identify malicious webpages, Cohen (1996) employed the decision trees for detection and Dumais, Platt, Heckerman, and Sahami (1998) utilized inductive learning algorithms and representations for text categorization. Guan, Chen, and Lin (2009) focused on classifying URLs that appear in webpages. Several URL-based features were used such as webpage timing and content. This paper has used similar techniques but applied them to webpages which have much more complex structures with better accuracies. Mcgrath and Gupta (2008) did not construct a classifier but performed a comparative analysis of phishing and non-phishing URLs. With respect to data sets, they compared non-phishing URLs drawn from the DMOZ Open Directory Project to phishing URLs from Phishtank (2013) and a non-public source. The features they analyzed included IP addresses, WHOIS thin records (containing date and registrar provided information only), geographic information, and lexical features of the URL (length, character distribution and presence of predefined brand names). The difference is that this paper utilizes different types of features to add to the novelty. Provos et al. (2008) carried out a study of drive-by exploit URLs and used a patented machine learning algorithm as a pre-filter for virtual machine (VM) based analysis. This approach is based on heavyweight classifiers and is time consuming. Provos et al. (2008) used the following features in computer simulation,