# An Evaluation of Lightweight Classification Methods for Identifying Malicious URLs

Shaun Egan, Barry Irwin
Security and Networks Research Group
Department of Computer Science
Rhodes University
Grahamstown, South Africa
Email: g10e4008@campus.ru.ac.za

*Abstract*—**Recent research has shown that it is possible to identify malicious URLs through lexical analysis of their URL structures alone. This paper intends to explore the effectiveness of these lightweight classification algorithms when working with large real world datasets including lists of malicious URLs obtained from Phishtank as well as largely filtered benign URLs obtained from proxy traffic logs. Lightweight algorithms are defined as methods by which URLs are analysed that do not use external sources of information such as WHOIS lookups, blacklist lookups and content analysis. These parameters include URL length, number of delimiters as well as the number of traversals through the directory structure and are used throughout much of the research in the paradigm of lightweight classification. Methods which include external sources of information are often called fully featured classifications and have been shown to be only slightly more effective than a purely lexical analysis when considering both false-positives and false-negatives. This distinction allows these algorithms to be run client side without the introduction of additional latency, but still providing a high level of accuracy through the use of modern techniques in training classifiers. Analysis of this type will also be useful in an incident response analysis where large numbers of URLs need to be filtered for potentially malicious URLs as an initial step in information gathering as well as end user implementations such as browser extensions which could help protect the user from following potentially malicious links. Both AROW and CW classifier update methods will be used as prototype implementations and their effectiveness will be compared to fully featured analysis results. These methods are interesting because they are able to train on any labelled data, including instances in which their prediction is correct, allowing them to build a confidence in specific lexical features. This makes it possible for them to be trained using noisy input data, making them ideal for real world applications such as link filtering and information gathering.**

## I. Introduction

In recent trends the motivation behind malicious websites has moved towards financial gain [1]. This is primarily done through the use of phishing and spam sites that attempt to sell fake goods such as pharmaceuticals or through the use of ”drive-by-downloads”, causing the user to unknowingly install malicious applications [2]. The main outcomes of malicious content can be broadly grouped into the following three categories:

- Phishing
- Fraudelent advertising
- Computer infection for unauthorized use

### A. Phishing

Phishing is an attack whereby an attacker tries to obtain user's personal information by trying to trick the user into entering identifying and account information for a legitimate service. This method predominantly targets financial and payment service sectors as indicated by [3]. Phishing attacks focusing on the financial and payment service sectors account for about 71% of the phishing attacks during that time, as indicated by the statistics.

### B. Fraudulent advertising

This method of attack tries to prompt users to buy counterfeit goods at low cost. The main vector for this attack is through email spam advertising which will often show a price list as well as a URL to the web page where the goods may be purchased.

### C. Computer infection for unauthorized use

Botnets are primarily propagated in a manner whereby a user installs software which exploits their machine. This can be done via many attack vectors, including that of drive-by-downloads. These are executed when a user follows a link to a web page which then exploits the user's browser, installing the software in the background without the user's knowledge or permission.

## II. Structure

The *Obfuscation* section of this paper describes the methods whereby the owners of malicious websites try to obfuscate the URLs with the intention of making them more difficult to detect and the different types of obfuscation employed. The next section entitled *Counter Measures* describes methods employed by the security industry to try to detect these malicious URLs and attempt to warn the user as to their content (or even forcibly stop the user from visiting them). The two broad categories discussed are blacklisting and lexical analysis of the URL itself. The section titled *Lightweight Classification Algorithms* is a discussion around three modern algorithms designed to train perceptrons (using lexical analysis) to identify malicious URLs without the use of external data sources while remaining highly accurate. The algorithms discussed are the *Online Perceptron*, *Confidence Weighted* and

*Adaptive Regularization of Weights* methods. The applications of these lightweight classification applications are discussed in the section *Applications*, as well as the work already achieved in our research and work currently underway. The final section *Conclusion* summarises the topics covered in this paper.

## III. Obfuscation

One of the challenges faced when identifying malicious URLs is that they are often obfuscated using a variety of methods. Doshi et al. [4] list the following four types of obfuscation which are intended to hide the malicious nature of the web site.

- Type I - This type of obfuscation refers to cases where the hostname is replaced with an IP address and in some cases where a port number is used. This can be particularly effective when the site that is being impersonated is somewhere in the URL path and the IP address is represented as hexadecimal value [4].
- Type II - The hostname in the URL has a domain name that appears to be legitimate but usually contains a redirect to another host [4].
- Type III - Again, the host name is obfuscated, but in this form a large string of other valid domains is appended to it. This gives the appearance of a valid hostname [4].
- Type IV - Misspelled or no domain name is given in the URL [4].

The purpose of this obfuscation is to hide the true nature of the URL and to make it appear to be a legitimate website, tricking users into believing that the URL is safe. Obfuscation also intends to make the URL more difficult to detect through automated testing, but it is this fact however, that makes these URLs look suspicious to advanced users and make them identifiable by machine learning.

## IV. Counter Measures

As mentioned in [2], the user is required to follow a URL to become a target. This is where a large amount of research, generated by the security industry, is focused and tries to find methods of preventing users from following URLs which may be potentially malicious or contain fraudulent content.

### A. Blacklists

There are several blacklists available which are a collection of malicious URLs and can be queried before visiting a page. Phishtank is one such site and provides a blacklist of malicious URLs which is supplied by the general public and then verified as having malicious content [5], [6]. This is one method of generating blacklists and can be very accurate as it uses human verification. A disadvantage with this method of blacklisting is that it can be slow as a direct result of the verification process. Another difficulty is that it requires users to discover malicious pages and report them. As a result, new malicious pages simply may not be on the blacklist or may still be waiting for verification.

Other methods for creating and maintaining blacklists are outlined in [7] and [2] and include honeypot data, web crawlers and heuristic content analysis of the particular webpage. However, these are not always accurate, as indicated by [2], because malicious pages have been known to 'cloak' themselves and display specific content depending on who is requesting it. For example, some pages will show completely benign content to IP addresses originating from known security companies and web crawlers, making them almost impossible to detect by these methods.

These blacklists can be queried in several ways, examples of this include browser plugins such as those offered by the Microsoft Smart Screen service [8] and Google Safe Browsing service [9], firewalls, proxies and search engines. Another form of blacklisting occurs in spam filters which filter out mail according to a blacklist of addresses.

The Google Smart Screen API may be used for the purpose of creating URL blacklist filters and is available from [9]. Another method of implementing blacklisting is offered through websites provided by security companies such as Norton [10] where users may enter the URL of a suspect website before visiting it to determine whether it is in the blacklist. This is not an ideal situation as the user has to enter the URL into the site which costs time and will almost definitely not be done for every link that the user wishes to visit. A more effective solution is to have each link checked as the user is browsing through the use of browser extensions that use blacklists or classifiers.

### B. Classification

A new method of identifying potentially malicious web sites is through the use of machine learning using Artificial Neural Networks (ANN) as classifiers, such as those presented in [2], [7], [6]. There are different approaches to this and vary in terms of features that are analysed an learning algorithms used to update the classifier.

The features analysed by a classifier can be identified as host-based and lexical, the combination of the two is known as full-featured analysis [6].

*1) Host-based features:* Host based features are those that require the use of external sources. The two sources of information used in [6] are WHOIS and Team Cymru which is available from [11]. The following list describes important information outlined in [2] and [7].

- WHOIS data - Details included are registration dates of the domain, registrars and registrants. This allows the classifier to determine how new the domain is and whether or not the domain belongs to an individual already associated with other malicious URLs. Also identified as important information in [2] is the expiration date and whether or not the WHOIS entry is locked.

- IP address information - [2] uses this information to check whether or not an IP address is in a blacklist and checks to see if the IPs of the A, MX and NS records in within the same AS as each other. This feature is described in [7] as including all identifying information regarding the hosting of the website and includes the IP address prefix and the AS number. This allows a specific ISP's IP prefix to be flagged as malicious by the classifier. Also associated with this data is geolocation of the IP address.

- Connection speed - This is cited as being an important factor as malicious content is often hosted on compromised machines, usually private machines with low connection speeds such as DSL [7].

- Domain Name - Included in these properties are the TTL, whether or not the certain keywords exist in the hostname and if it contains an IP address [2].

  These properties are common to almost all research regarding classification using host-based features. The features represent a valuable set of data and are fairly easy to obtain through automated software. The negative aspect of using these external features for classification is that they may incur significant additional latency in the case of using the classifier as a browser extension and as such, may not be appropriate on connections where bandwidth is limited [6].

*2) Lexical features:* Lexical features of a URL refer to the actual text of a URL and include no external information. These features are useful as malicious URLs often ”look” different than benign ones to experts [2]. Features that belong to this group include numerical information regarding lengths of features, numbers of delimiters and directory structure. This information is useful as it is obfuscation resistant [6]. The model outlined by [6] will be used for the purposes of this research and will be described below.

- Automatic features - These features describe the URL in the form of tokens and include items such as the domain name, top-level domain, directory structure, the file being accessed, it's file type and the arguments passed.

- Domain name - Length, number of tokens and hyphens are collected. Also, whether the domain name uses and IP address or port number used as binary features that are extracted from the domain name.

- Directory - Among the directory structure considerations are how many directory traversals are used, the largest number of delimiters used in a particular directory, and the longest directory name.

- Filename - Extracted filename features include the length of the filename and the number of delimiters

- Arguments - A very important factor in the feature extraction is that of the arguments passed to the file. These features include the number of variables passed, the longest variable value passed and the highest number of delimiters used.

### C. Conclusion

These features, be it host-based, lexical or a combination of both, are then run through a classifier which will then present a prediction as to whether the URL is malicious or benign. This is usually represented as a 1 for malicious or a 0 as benign.

## V. Lightweight Classification Algorithms

Classifiers range from state of the art machine learning techniques to simple perceptrons. All classifying algorithms share a common structure based on the structure of the perceptron, but differ in feed forward propagation and back-propagation learning algorithms. While the structure of these simple artificial neural networks is shared, the learning mechanism represents a significant difference and greatly affects the effectiveness of the algorithm.

Classification algorithms which only use the lexical features are known as lightweight classification algorithms and have a major advantage over fully featured classification algorithms as they do not require the use of external information sources to make a prediction about the safety of a URL. This is a noteworthy factor, especially when due consideration is given to performance on the client's side of the process, as no additional latency is introduced through the execution of lookups. Another positive factor in this regard is the use of a feed forward mechanism of a neural network, which may be executed efficiently, thereby speeding up the process substantially.

### A. Online Perceptron

The structure of a perceptron is known as a single neuron artificial neural network. It consists of a number of input neurons in a layer which take on the values of the input fields that they represent. They also have a single neuron in a second layer which is connected to the first via a series of weighted connections and acts as a single output layer. The input layer is not considered a layer in neural networks, so the single neuron is considered the only layer and is thus known as a single layer neural network and works as a binary linear classifier.

The neural network mechanism can be described as two steps. The first step is used by the network to create a prediction given the input supplied and is known as the ”Feed forward” mechanism Firstly, the inputs are multiplied by their respective weights and summed. This process is known as a

Linear Combiner and is given by:

$$\text{Linear Combination} = \sum_{i=1}^{n} x_i w_i \tag{1}$$

The result of this is then passed through a hard limiter which simply adds a bias (or threshold) to the value. This is then compared to 0, and if it is greater than 0, the output is set to 1, else it is set to false. This is known as a step activation function.

The entrire feed forward mechanism is given by:

$$f(x) = \left\{ \begin{array}{ll} 1 & \text{if } x \cdot w + b > 0 \\ 0 & \text{if } x \cdot w + b \leq 0 \end{array} \right\} \tag{2}$$

Where $f(x)$ represents the prediction, $x$ represents the input vector, $w$ represents the weight vector assigned to the inputs and $b$ represents the bias (or threshold). This activation function is known as a step activation function. In multilayer neural networks, this step is repeated for each neuron on each layer until the all neurons are activated. The final layer represents the output and gives the entire network's prediction based on the supplied input.

The second step is known as the learning mechanism and is used in conjunction with the feed forward mechanism to facilitate the learning process. Learning requires that the input data is "labeled". This means that a correct answer is supplied with the input so that when the neural network generates a prediction, the correct label can be used to create an error value ($\gamma$). This value is then used by the back-propagation (or learning) mechanism to update weights in the neural network. This mechanism can be said to move through the neurons in the same way that the feed forward mechanism does, only in reverse.

The weight updates for the perceptron are calculated via the following method:

$$\gamma = Y_d - Y \tag{3}$$

Where $\gamma$ represents the error, $Y_d$ represents the desired output, or the labeled output, and $Y$ represents the actual output of the neuron. The following formula shows how the change in any particular weight may be calculated given the input associated with that weight:

$$\Delta w = \alpha \times x_i \times \gamma \tag{4}$$

Where $\Delta w$ represents the change to be made to the weight $w$ and $\alpha$ is the learning rate for the network. This learning rate is a customizable value that must be larger than 0.

Finally the weights are adjusted via the following formula:

$$w_{t+1} = w_t + \Delta w_t \tag{5}$$

This process of using labeled training data is repeated until the neural network's error reaches an acceptable level. It is important to note at this point that a perceptron using these rules for weight training will only update values if an incorrect prediction is made. Also, the perceptron uses a linear update method which can be detrimental to the performance of the network as it has the ability to over compensate when an error is made.

It is shown in [6] that the online perceptron achieves an accuracy of 93% to 96% when trained on data from phishtank [5] and malwarepatrol [12]. It is indicated, however, that different classifiers should be used for the two different sources as they differ in the nature of the malicious URLs [6].

The basic model structure of the perceptron is shared by all of the following lightweight classifiers that are described here. As already mentioned, they only differ in their method of training, which greatly affects the model's ability to classify URLs correctly.

## *B. Confidence Weighted Algorithm*

The problem highlighted with the online perceptron is that it has the tendency to over or under compensate for an error on any particular feature. An example of such an error would occur if a domain is registered with URL features that resembles a legitimate web site. When the classifier predicts (incorrectly) that the URL is benign, it tries to correct itself with the same linear update method for other, more identifiable, errors. The Confidence Weighted (CW) algorithm tries to overcome this by maintaining a confidence level in each feature (or input) of the classifier [13]. This is done by keeping a record of the mean $\mu$ of the input weights and a covariance matrix $\Sigma$. The mean value for the input weight $i$ is represented by $\mu_i$ while $\Sigma_i$ represents the algorithm's confidence in the feature $i$. These two features alow the algorithm to update itself in proportion to the confidence that it has in any particular feature (smaller changes for features with high confidence and larger changes for ones with low confidence).

As indicated by [6], the weight of a feature $i$ at a timestep may be taken as $\mu$ for that feature. Classification of that URL is done in the same way as that of the perceptron: through the step activation function over the inputs and their respective weights. The learning method of the CW is shown below:

$$(\mu_{t+1}, \Sigma_{t+1}) = \arg\min_{\mu\Sigma} D_{KL}(\mathcal{N}(\mu, \Sigma) \| \mathcal{N}(\mu_t, \Sigma_t)), \tag{6}$$

$$\mathsf{s.t.Pr}_{w \sim \mathcal{N}(\mu,\Sigma)}[y_t(w \cdot x_t] \geq \eta \tag{7}$$

This equation is used to calculate the mean vector of the weights and the covariance matrix for the next time step. $D_{KL}$ represents the KL divergence between the normal distributions $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu_t, \Sigma_t)$ and is a standard measure of difference between distributions. It is shown in [6] that the CW algoritm is accurate in about 98% of cases, only 1% less accurate than a fully featured classification. This improvement over the online perceptron is due to method in which CW learns by updating through confidence levels in features.

## *C. Adaptive Regularization of Weights*

One problem with the CW algorithm, highlighted in [6], is that of noisy training data in terms of labels. If a URL is labeled incorrectly, the CW may label that kind of site as malicious in future. While it does not suffer highly from this problem due to its confidence in features, it can still cause enough of an error to start missing malicious sites. The idea behind Adaptive Regularization of Weights (AROW) is to use the CW's method of confidence in features, but to modify it in such a way as to be more tolerant of miss-labeled data [14]. The adapted learning method follows:

$$(\mu_{t+1}, \Sigma_{t+1}) = \arg\min_{\mu\Sigma} D_{KL}(\mathcal{N}(\mu, \Sigma) \| \mathcal{N}(\mu_t, \Sigma_t)) \quad (8)$$

$$+\lambda_1 l_{h^2}(y_t, \mu \cdot x_t) + \lambda x_t^T \Sigma x_t, \quad (9)$$

$$\mathsf{s.t.Pr}_{w\sim\mathcal{N}(\mu,\Sigma)}[y_t(w \cdot x_t] \geq \eta \quad (10)$$

The extra fields include $\lambda_n$ which are an adjustable parameters according to [6] and $y_t$ which is the desired prediction of the URL. This update to the CW makes AROW an accurate and more robust classifier, able to handle noisy (incorrectly labeled) training data. The authors of [6] achieved an accuracy of 96% to 97%.

## VI. Applications

One of the largest advantages of the lightweight classification of URLs is that it introduces little overhead in terms of latency and processing. While this fact has been mentioned several times in much of the work presented on the topic and the example use of client-side browser plugins has been mentioned, there are other uses to which this work can be put.

Firstly, a web proxy within an organisation may use the classifier to filter malicious URLs from being visited. This may be used on a standalone basis or it can be used to augment the proxy's existing blacklist of URLs. If the proxy receives a request which the classifier predicts to be malicious, it may deny the request and add the URL to the blacklist or another database system for further analysis. Another use for these lightweight classifiers is that the may be used as an initial first pass of large URL logs when trying to analyse a network incident. The URLs may then be flagged for further analysis such as information gathering from external sources, such as WHOIS data.

### A. Implementation

The primary goal of this research is to implement all three of these lightweight classifiers and to test them on data collected from real world traffic. Finally, once they have been tested, the classifiers will be put to work within a framework designed for use in incident analysis, a browser plugin for use within organisations and a tool designed to work with a proxy to filter requests for malicious sites.

Another possible outcome of this implementation would be to create a plugin for use within email clients that could scan emails for URLs that link to malicious pages. This would be especially useful in trying to combat fraudulent advertising and would not be any different in terms of classifier than any of the other tools mentioned here.

### B. Training data

Le et al [6] trained classifiers on grouped data such as the pairing of Phishtank and Yahoo random benign URLs. They showed, through their results, that seperate classifiers should be maintained for these different pairs as the data from Phishtank and MalwarePatrol indicate different types of malicious sites (phishing vs. malware) and that these different classifiers results in a higher accuracy than the grouping of all malicious URLs and all benign URLs.

## VII. Future Work

Training data has already been collected from Phishtank, MalwarePatrol and a live proxy, serving thousands of requests a day. In excess of 4000 URLs have been gathered from Phishtank as well as another 4000 from malware patrol. It is important to note that the proxy data is already largely filtered through standard means such as blacklisting. Also, benign URLs have been gathered from Yahoo's random URL generator (available from [15]).

Software has already been written to handle formatting and output of standardized labeled URLs from different sources. A prototype implementation of the Online Perceptron has been written in Matlab. Implementations of the CW and AROW algorithms are nearing completion at the time of writing, and will also be implemented in Matblab. The final implementation for testing of these three algorithms will be written in python, using the numpy package for integration with Matlab. Training the classifiers will be done in the method suggested by [6], where the data sources will be grouped by the type of malicious content that they list. For this reason, two versions of each classifier will be maintained, allowing these tools to classify a URL in terms of malicious content in addition to phishing content.

Once the these alogrithms have been tested, the tools mentioned in *Applications* will be implemented. The first will be a browser plugin for the Firefox web browser. The second will be a proxy plugin followed by a mail server addon. All of these implementations will be updated through a central server that will query external data sources periodically for new training data with which to update its classification models. Finally, a tool will be developed that will enable these classifiers to be deployed as plugins for a forensic incident analysis framework.

## VIII. Conclusion

The work presented by [2], [7], [5] has shown that the effectiveness of lightweight classifier is only slightly lower than that of fully-featured classification. Also, the ability to classify a URL with extremely high accuracy and low overhead means that it is ideal for use in browsers and as a first pass analysis of large traffic logs. It is the intention of this research to build these tools and to test their effectiveness in real world situations using real world traffic data, primarily using a proxy server that serves several thousand requests a day and as modular tool built for use in a forensic data gathering framework. These implementations will provide a good understanding of how well lightweight classification algorithms work in real time on real world networks.

## References

[1] Microsoft security intelligence report. [Online]. Available: http://www.microsoft.com/presspass/press/2008/apr08/04-22sirv2pr.mspx; Last accessed: 27/04/2011

[2] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious urls," in *Proceedings of theSIGKDD Conference. Paris,France*, 2009.

[3] Anti phishing workgroup. [Online]. Available: http://antiphishing.org; Last accessed: 27/04/2011

[4] S. Doshi, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," Johns Hopkins University, Tech. Rep., 2006.

[5] Phishtank. [Online]. Available: http://www.phishtank.com/; Last accessed: 27/04/2011

[6] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: Url names say it all," September 2010.

[7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: An application of large-scale online learning," in *In Proc. of the International Conference on Machine Learning (ICML*, 2009.

[8] Microsoft smart screen. [Online]. Available: http://windows.microsoft.com/en-US/windows-vista/SmartScreen-Filter-frequently-asked-questions; Last accessed: 27/04/2011

[9] Google safe browsing api. [Online]. Available: http://code.google.com/apis/safebrowsing/; Last accessed: 12/05/2011

[10] Norton safe web. [Online]. Available: http://safeweb.norton.com/; Last accessed: 27/04/2011

[11] Team cymru. [Online]. Available: http://www.team-cymru.org/; Last accessed: 27/04/2011

[12] Malwarepatrol. [Online]. Available: http://www.malwarepatrol.net/; Last accessed: 27/04/2011

[13] K. Crammer, M. D. Fern, and O. Pereira, "Exact convex confidence-weighted learning," in *In Advances in Neural Information Processing Systems 22*, 2008.

[14] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 414–422.

[15] Yahoo random url generator. [Online]. Available: http://random.yahoo.com/bin/ryl; Last accessed: 27/04/2011