Weedon, Martyn, Tsaptsinos, Dimitris and Denholm-Price, James (2017) Random forest explorations for URL classification. In: ., ., (ed.) 2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA). Institute of Electrical and Electronics Engineers, Inc. ISBN 9781509050604 https://doi.org/10.1109/CyberSA.2017.8073403

Random Forest Explorations for URL Classification

Martyn Weedon
Faculty of SEC
Kingston University
Penrhyn Road
Kingston Upon Thames
United Kingdom

Dimitris Tsaptsinos
Faculty of SEC
Kingston University
Penrhyn Road
Kingston Upon Thames
United Kingdom

James Denholm-Price
Faculty of SEC
Kingston University
Penrhyn Road
Kingston Upon Thames
United Kingdom

***Abstract*—Phishing is a major concern on the Internet today and many users are falling victim because of criminal's deceitful tactics. Blacklisting is still the most common defence users have against such phishing websites, but is failing to cope with the increasing number.**

In recent years, researchers have devised modern ways of detecting such websites using machine learning. One such method is to create machine learnt models of URL features to classify whether URLs are phishing. However, there are varying opinions on what the best approach is for features and algorithms.

In this paper, the objective is to evaluate the performance of the Random Forest algorithm using a lexical only dataset. The performance is benchmarked against other machine learning algorithms and additionally against those reported in the literature. Initial results from experiments indicate that the Random Forest algorithm performs the best yielding an 86.9% accuracy.

Keywords—Phishing, URL, machine learning, Random Forest, lexical features

I. INTRODUCTION

Phishing is a method used by criminals to deceive and trick users into releasing personal and sensitive data, such as their identity or financial information. Criminals can achieve this by employing social engineering tactics to spoof the user. For example, one such tactic is through email. Often a criminal, or phisher, will send emails to vulnerable users that will request personal data and lure them into clicking Uniform Resource Locator (URL) links to malicious web pages [1].

Phishing attacks are growing. The Anti Phishing Work Group reported that between January 2016 and June 2016 the number of unique phishing sites was 466,065, which was 61% higher than the previous quarter [2].

At present, blacklisting is the most common anti-phishing technique used [3]. Blacklists are populated using various techniques, such as honeypots, web crawlers combined with analysis heuristics and user reported URLs [4]. Unfortunately, not all malicious URLs are in blacklists as they could be too new, were never found or were never evaluated properly [4].

To deal with the growing number of malicious phishing URLs, another method of detection is to employ the use of machine learning algorithms and in this paper, the focus is on the Random Forest (RF) [5] algorithm to classify URLs as either malicious or benign.

RF is an ensemble algorithm introduced by Leo Breiman that generates a vector of decision trees. Each tree is generated using a bootstrap sample from the dataset and upon each split in the tree, the algorithm chooses the best feature from a random subset of features. To select the best feature, a measure of purity is computed and a feature found to have the smallest value of purity is selected. When classifying an instance, the RF algorithm uses the most frequently predicted class, also known as voting [6].

For this ongoing work, the classification of URLs will be lexical based, which means features will be extracted directly from the URL itself, for example, a character count of the domain. If URLs "look" malicious, such features can assist the learning process to find the tell-tale signs of what makes those URLs malicious [4]. The lexical based approach is considered lightweight when compared to other external features, which may require the resource of a real-time system. Thus, there is no delay in the classification [4, 7, 8]. In addition, previous studies have shown that using lexical features with additional features has only a slightly better accuracy than just solely using lexical features [4, 9].

The purpose of this paper is to illustrate how effective the RF algorithm is at classifying malicious URLs against three other algorithms, J48, Naïve Bayes and Logistic Regression. The comparison will be based upon how each algorithm does in terms of accuracy, sensitivity (recall) and specificity and will further show that the RF algorithm can produce the lowest false negatives. This is important because if any algorithm, not just RF, that classifies a URL as benign when it is malicious can potentially cause more harm for the user.

II. RELATED WORK

Basnet, Sung and Liu [3] discovered RF to be the best out of 7 other algorithms in terms of accuracy and speed. Additionally, they adopt the lexical approach choosing features believed to be "phishing like" and using solely lexical features they report an 85.38% accuracy with a false positive rate of 8.22% [3].

Le, Markopoulou and Faloutsos [9] discuss how effective lexical features are over using full features for phishing URLs.

Their full feature set includes lexical, but also external features, where they query the WHOIS and Team Cymru servers for registration and geolocation data, respectively. They find that when experimenting with the online learning algorithms Confidence Weight and Adaptive Regularisation of Weights, the results show that lexical features are comparable to the full set with only a 1% difference in accuracy for both algorithms [9].

Ma, Saul, Savage and Voelker [4] explore the potential of machine learning on lexical and host based features of URLs. They state that lexical features tend to "look different", whilst host based features describe "where", "who" and "how" URLs are managed. Their data are run against 4 machine learning algorithms: Naïve Bayes, Support Vector Machines (one with a linear kernel and the other with an RBF kernel) and Logistic Regression. They report that, when running the lexical features dataset, using Logistic Regression yields a 98.07% accuracy, whereas running full features (lexical and host based) using the same algorithm yields a 98.76% accuracy [4].

Garera, Provos, Chew and Rubin [1] provide a framework for detecting malicious URLs. They explain that there are 4 types of obfuscation to a URL that can be shown to be recognised as phishing. These are Type I (obfuscating the host with an IP address), Type II (obfuscating the host with another domain), Type III (obfuscating with large host names) and Type IV (domain unknown or misspelled). They use this approach with page and domain based features, as well as hand selecting 8 words they believe to be common amongst phishing URLs. In their work, the Logistic Regression algorithm was chosen and following evaluation received an average accuracy of 97.31% with a true positive rate of 95.8% and a false positive rate of 1.2% on their testing dataset [1].

Darling, Heileman, Gressel, Ashok and Poornachandran [8] describe a lexical approach to classifying malicious URLs using the J48 algorithm in Weka. They state that classification needs to be lightweight to reduce the delay to real-time systems. The training dataset used consists of 131,402 URLs with a 50/50 split between benign and malicious classes and consists of 87 lexical features, which are categorised into 6 groups: n-grams, lengths, counts, patterns, binaries and ratios. The results of J48 against the training dataset show a 99.1% accuracy with a false positive rate of 1.7% and a false negative rate of 0.5% [8].

III. Experiments

A. Data

The data are retrieved from two online sources, one for benign URLs and the other for malicious URLs.

The benign URL dataset was retrieved from the DMOZ Open Directory website [10] on the 23rd of February 2016. The DMOZ Open Directory is a free, open source and community based website that allows users to browse the directory and suggest URLs. In addition, other users can volunteer to edit the directory, which allows for some validation of URLs prior to them being added, modified or deleted in the DMOZ [11].

The malicious URL dataset was retrieved from the PhishTank website [12] on the 24th of February 2016 and all URLs were verified as malicious at the time of retrieval. PhishTank is a free community website that allows users to submit, verify, track and share phishing URL data [13].

The online datasets retrieved were both cleansed by removing any duplicates and for the experiments both a training and testing set were created. The training set consists of 4000 URLs, 3000 from the benign set and 1000 from the malicious set. The testing set consists of 7000 URLs, 3000 from the benign set and 4000 from the malicious set. All URLs were selected randomly, except any URLs selected in the testing set do not include those that were present in the training set.

The next step was to extract features from the URLs. The features obtained were those used in the literature [1, 3, 9] and the reason is twofold. The results published, firstly, proves that the features are a good baseline and secondly, provides a better comparison of algorithm results to other researchers. Table I provides a list of the features used and extracted from the URLs, and includes the class label.

To ensure equality between features, all numeric values were normalised, so their values lie between 0 and 1. All features in Table I are counts and binary values of specific entities within the URL.

TABLE I. Extracted Features

L_Host_Len	L_Semicolon_In_Path	L_Hyphen_Count_In_Host
L_Dot_In_Host	L_Comma_In_Path	L_Host_LongestTokenLen
L_Dot_In_Path	L_Has_Query	L_Path_NumOfSubDirs
L_Dot_In_URL	L_Equals_In_Query	L_Path_LongestSubDirTokenLen
L_Path_Len	L_Has_Fragment	L_Path_NumOfDelimitersNotDots
L_URL_Len	L_AtSign_In_URL	L_Filename_Len
L_Hyphen_In_Host	L_Username_In_URL	L_Filename_NumOfDots
L_Digit_In_Host	L_Password_In_URL	L_Filename_NumOfHyphenAndUnderscores
L_IP_Host	L_NonStdPort	L_Arguments_Len
L_Hex_Host	L_Underscore_In_Path	L_Arguments_Count
L_Hyphen_In_Path	L_URL_BlacklistedWordAppears	L_Arguments_LenOfLongestArgVal
L_ForwardSlash_In_Path	L_Host_HasPortNum	L_Arguments_NumOfOtherDelimitersInValues
L_Equals_In_Path	L_Host_NumOfTokens	*Class*

B. Methodology

For the experiments, 4 machine learning algorithms were run in Weka [14], these are Naïve Bayes (NB), Logistic Regression (LR), J48 (also known as the C4.5 algorithm) and Random Forest (RF). Apart from LR, all algorithms were run with the default settings. The ridge parameter for LR was set to prevent the algorithm from placing too much weight on coefficients. The reason being is that if too much weight is placed on certain features, the algorithm becomes prone to overfitting the data.

All models were trained first using 10-fold cross validation. The results in the next section are for the testing dataset and are presented in terms of the accuracy, sensitivity and specificity using the definition in the work by [15]. The measure in each definition is calculated using values that demonstrate the performance of a classifier. A contingency table has been used to define these values in this work (Table II), where TP is the true positive, TN is the true negative, FP is the false positive and FN is the false negative.

Accuracy is a measure of how many instances were correctly classified out of all instances classified, it is defined in (1) [15].

$$(TP + TN) / (TP + TN + FP + FN) \quad (1)$$

Sensitivity (also known as recall or the true positive rate) is a measure of the *positive* instances that have been correctly classified as *positive* over all instances declared *positive* in the dataset, it is defined in (2) [15].

$$TP / (TP + FN) \quad (2)$$

Specificity (also known as the true negative rate) is a measure of the *negative* instances that have been correctly classified as *negative* over all instances declared *negative* in the dataset, it is defined in (3) [15].

$$TN / (TN + FP) \quad (3)$$

TABLE II. CONTINGENCY TABLE EXAMPLE

		Classification	
		Malicious	***Benign***
Actual	***Malicious***	TP	FN
	Benign	FP	TN

C. Results

The initial results (Table III) prove that the RF algorithm is a better choice for the data as it can more accurately classify the data whilst keeping the false negatives moderately low. Furthermore, our results agree with the published work of [3].

The main aim of this work is to get the false negatives as low as possible, as these instances are classified as benign when in fact they are malicious. RF was the only algorithm to get this as low as possible when compared to the other three algorithms. In contrast to this, RF obtained the highest number of false positives when compared to the other three algorithms. False positives are instances that are classified as malicious when in fact they are benign. As such, false positives are an important measure in classification, but they are not emphasised in this work because if a benign URL has been classed as malicious, the level of risk to the user will remain unchanged. In other words, the user remains protected.

After training the RF model, the feature importance was obtained to determine what features the algorithm considered the most important. In this work, the feature importance is computed by counting the number of nodes used by each feature listed in Table I, excluding the class. The top 6 important features have been listed in Table IV. In addition, using the URL in Fig. 1 as an example, the table also illustrates what text would be extracted and what the computed value would be from the extraction for each feature.

TABLE III. EXPERIMENT RESULTS (TESTING DATASET)

Algorithm	**Accuracy**	**Sensitivity**	**Specificity**	**False Positives**	**False Negatives**
NB	64.6%	39.7%	97.7%	69	2412
LR	81.5%	70.5%	96.1%	118	1180
J48	83.9%	75.0%	95.8%	126	1002
RF	**86.9%**	**80.5%**	**95.4%**	**138**	**782**

`http://www.kingston.ac.uk/aboutkingstonuniversity/`

Fig. 1. Example URL

TABLE IV. FEATURE IMPORTANCE (TOP 6)

Rank	Feature	Node Count	Extraction	Example Value
1	L_Host_LongestTokenLen	5032	`www.kingston.ac.uk`	8
2	L_Host_Len	4945	`www.kingston.ac.uk`	18
3	L_URL_Len	4944	(See Fig. 1)	50
4	L_Path_Len	3879	`aboutkingstonuniversity/`	24
5	L_Path_LongestSubDirTokenLen	3052	`aboutkingstonuniversity/`	23
6	L_Path_NumOfSubDirs	1584	`aboutkingstonuniversity/`	2

The first and most important feature is *L_Host_LongestTokenLen*, which contains the longest token length from the hostname. This feature is used by [9] and they split the hostname into tokens using these characters as delimiters: a forward slash (/), a question mark (?), a full stop (.), an equals sign (=), an underscore (_), an ampersand (&) and a hyphen (-). After splitting the hostname, a search is performed to find the length of the longest token [9], which in the example above is 8 because `kingston` is the longest token.

The second, third and fourth features are *L_Host_Len*, *L_URL_Len* and *L_Path_Len*, which are the total lengths of the hostname, URL and path, respectively.

Similarly, to the first feature, the fifth feature is *L_Path_LongestSubDirTokenLen*, also used by [9], which removes the filename (if present) and the beginning forward slash (/) from the path. The path is then split into tokens using the forward slash (/) character. After splitting, a search is performed to find the length of the longest token [9], which in the example above is 23 because `aboutkingstonuniversity` is the longest and only token.

The last feature is *L_Path_NumOfSubDirs*, this attribute uses the path of the URL, which ignores the filename (if present) and the beginning forward slash (/). The result is split into tokens using the forward slash (/) character and a count of tokens is retrieved. In the example, the path `aboutkingstonuniversity/` has 2 tokens: `aboutkingstonuniversity` and a blank value, which is where the filename would normally reside.

IV. FUTURE WORK

The main aim of this work is to lower the false negative value and the initial results indicate that this could be possible by introducing a cost matrix, which would penalise the false negatives, however, this needs further investigation. A comprehensive review of different levels of penalisation will need to be performed to evaluate what effect these have on the RF algorithm in terms of accuracy, false positives and false negatives.

In addition, a bag-of-words approach will be explored to evaluate how the RF algorithm will perform when adding more features.

REFERENCES

[1] S. Garera, N. Provos, M. Chew and A. Rubin D., "A framework for detection and measurement of phishing attacks," Worm '07, pp. 1-8, 2007.

[2] Anti Phishing Work Group, "Phishing Attack Trends Report - 2Q 2016," vol. 2016, 2016.

[3] R.B. Basnet, A.H. Sung and Q. Liu, "Learning to detect phishing URLs," IJRET: International Journal of Research in Engineering and Technology, vol. 3, pp. 11-24, 2014.

[4] J. Ma, L. Saul K., S. Savage and G. Voelker M., "Beyond blacklists: learning to detect malicious web sites from suspicious URLs," Kdd '09, pp. 1245-1254, 2009.

[5] L. Breiman, "Random forests," Mach.Learning, vol. 45, pp. 5-32, 2001.

[6] A. Cutler, D.R. Cutler and J.R. Stevens, "Ensemble Machine Learning," C. Zhang and Y. Ma, Dordrecht: Dordrecht : Springer, 2012, pp. 157-175.

[7] S. Egan and B. Irwin, "An evaluation of lightweight classification methods for identifying malicious URLs," in 2011 Information Security for South Africa, pp. 1-6, 2011.

[8] M. Darling, G. Heileman, G. Gressel, A. Ashok and P. Poornachandran, "A lexical approach for classifying malicious URLs," in High Performance Computing & Simulation (HPCS), 2015 International Conference on, pp. 195-202, 2015.

[9] A. Le, A. Markopoulou and M. Faloutsos, "Phishdef: Url names say it all," in INFOCOM, 2011 Proceedings IEEE, pp. 191-195, 2011.

[10] AOL, "DMOZ - RDF Data," vol. 2016, 2016.

[11] AOL, "DMOZ Open Directory - About Us," vol. 2016, 2016.

[12] PhishTank, "PhishTank > Developer Information," vol. 2016, 2016.

[13] PhishTank, "PhishTank - FAQ Page," vol. 2016, 2016.

[14] F. Eibe, M.A. Hall and I.H. Witten, The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2016, .

[15] M. Bramer, "Measuring the Performance of a Classifier," in Principles of Data Mining, SpringerLink (Online service), London : Springer London : Imprint: Springer, 2016, pp. 178-179.