Summer 8-1-2019

Phishing Websites Detection using Machine Learning

Arun Kulkarni

Phishing Websites Detection using Machine Learning

Arun Kulkarni[1], Leonard L. Brown, III[2]
Department of Computer Science
The University of Texas at Tyler
Tyler, TX, 75799

***Abstract*—Tremendous resources are spent by organizations guarding against and recovering from cybersecurity attacks by online hackers who gain access to sensitive and valuable user data. Many cyber infiltrations are accomplished through phishing attacks where users are tricked into interacting with web pages that appear to be legitimate. In order to successfully fool a human user, these pages are designed to look like legitimate ones. Since humans are so susceptible to being tricked, automated methods of differentiating between phishing websites and their authentic counterparts are needed as an extra line of defense. The aim of this research is to develop these methods of defense utilizing various approaches to categorize websites. Specifically, we have developed a system that uses machine learning techniques to classify websites based on their URL. We used four classifiers: the decision tree, Naïve Bayesian classifier, support vector machine (SVM), and neural network. The classifiers were tested with a data set containing 1,353 real world URLs where each could be categorized as a legitimate site, suspicious site, or phishing site. The results of the experiments show that the classifiers were successful in distinguishing real websites from fake ones over 90% of the time.**

Keywords*—*Phishing websites; classification; features; machine learning

I. Introduction

While cybersecurity attacks continue to escalate in both scale and sophistication, social engineering approaches are still some of the simplest and most effective ways to gain access to sensitive or confidential information. The United States Computer Emergency Readiness Team (US-CERT) defines phishing as a form of social engineering that uses e-mails or malicious websites to solicit personal information from an individual or company by posing as a trustworthy organization or entity [1]. While organizations should educate employees about how to recognize phishing e-mails or links to help protect against the above types of attacks, software such as HTTrack is readily available for users to duplicate entire websites for their own purposes. As a result, even trained users can still be tricked into revealing private or sensitive information by interacting with a malicious website that they believe to be legitimate.

The above problem implies that computer-based solutions for guarding against phishing attacks are needed along with user education. Such a solution would enable a computer to have the ability to identify malicious websites in order to prevent users from interacting with them. One general approach to recognizing illegitimate phishing websites relies on their Uniform Resource Locators (URLs). A URL is a global address of a document in the World Wide Web, and it serves as the primary means to locate a document on the Internet. Even in cases where the content of websites are duplicated, the URLs could still be used to distinguish real sites from imposters.

One solution approach is to use a blacklist of malicious URLs developed by anti-virus groups. The problem with this approach is that the blacklist cannot be exhaustive because new malicious URLs keep cropping up continuously. Thus, approaches are needed that can automatically classify a new, previously unseen URL as either a phishing site or a legitimate one. Such solutions are typically machine-learning based approaches where a system can categorize new phishing sites through a model developed using training sets of known attacks.

One of the main problems with developing machine-learning based approaches for this problem is that very few training data sets containing phishing URLs are available in the public domain. As a result, studies are needed that evaluate the effectiveness of machine-learning approaches based on the data sets that do exist. This work aims to contribute to this need. Specifically, the goal of this research is to compare the performance of the commonly used machine learning algorithms on the same phishing data set. In this work, we use a data set, where features from the data URLs have already been extracted, and the class labels are available. We have tested common machine learning algorithms for the purpose of classifying URLs such as SVM, Naïve Bayes' classifier, decision tree, and neural network.

The remainder of this paper is structured as follows. Section II describes the related work in classifying phishing URLs. Section III provides the details of the data set and methodology, Section IV describes the results of the tests and provides discussion. Section V describes limitations of the present work and directions for the future work.

II. Related Work

Machine learning techniques that identify phishing URLs typically evaluate a URL based on some feature or set of features extracted from it. There are two general types of features that can be extracted from URLs, namely host-based features and lexical features. Host based features describe characteristics of the website, such as where it is located, who manages it, and when was the site installed. Alternatively, lexical features describe textual properties of the URL. Since URLs are simply text strings that can be divided into subparts including the protocol, hostname, and path, a system can assess a site's legitimacy based on any combination of those components.

Many machine learning techniques have been used for detection of malicious URLs. Sadeh et al. [2] proposed a system called PILFER for classifying phishing URLs. They extracted a set of ten features that are specifically designed to highlight deceptive methods used to fool users. The data set consists of approximately 860 phishing e-mails and 6950 non-phishing emails. They used a Support Vector Machine (SVM) as a classifier in the implementation. They trained and tested the classifier using 10-fold cross validation and obtained 92 percent accuracy.

Ma et al. [3] considered the URL classification problem as a binary classification problem and built a URL classification system that processes a live feed of labeled URLs. It also collects URL features in real time from a large Web mail provider. They used both lexical and host-based features. from the gathered features and labels, they were able to train an online classifier using a Confidence Weighted (CW) algorithm. Parkait et al. [4] provide a comprehensive literature review after analyzing 358 research papers in the area of phishing counter measures and their effectiveness. They classified anti-phishing approaches into eight groups and highlighted advanced anti-phishing methods.

Abdelhamid et al. [5] built a system for detecting phishing URLs called Multi-label Classifier based on Associative Classification (MCAC). They used sixteen features and classified URLs into three classes: phishing, legitimate, and suspicious. The MCAC is a rule-based algorithm where multiple label rules are extracted from the phishing data set. Patil and Patil [6] provided a brief overview of various forms of web-page attacks in their survey on malicious webpages detection techniques.

Hadi et al. [7] used the Fast-Associative Classification Algorithm (FACA) for classifying phishing URLs. FACA works by discovering all frequent rule item sets and building a model for classification. They investigated a data set consisting of 11,055 websites with two classes, legitimate and phishing. The data set contained thirty features. They used the minimum support and the minimum confidence threshold values as two percent and fifty percent, respectively.

Nepali and Wang [8] proposed a novel approach to detect malicious URLs using only visible features from social networks. Kuyama et al [9] proposed a method for identifying the Command and Control server (C&C server) by using supervised learning and features points obtained from WHOIS and DNS information. They evaluated domain names and e-mail addresses from the WHOIS as input values for machine learning.

In addition to the above solutions, several researchers have surveyed the field of malicious URL detection. Sahoo et al. [10] provide a comprehensive survey and structural understanding of malicious URL detection techniques using machine learning.

III. Methodology

A. Dataset

The data set used in this paper was downloaded from the University of California, Irvine Machine Learning Repository, Center for Machine Learning and Intelligent Systems [11]. It contains features from 1353 URLs. Out of these, 548 are legitimate, 702 are phishing, and 103 are suspicious. The data set also contains nine features that were extracted from each URL. The attributes provide information such as the URL anchor, popup window, age of the domain, URL length, IP address, web traffic, etc. Each feature value holds categorical values, either binary or ternary. Binary values indicate that the existence or the lack of existence of the feature within the URL determines the value assigned to that feature. For ternary features, the existence of the feature in a specific ratio determines the value assigned to that feature. The features that we used in this research work are described in the following paragraphs.

1) Server Form Handler (SFH): Usually information is processed in the same domain where the webpage is loaded. In phishing websites, the server form handler is either empty or is transformed to another domain that is not legitimate.

2) Secure Socket Layer (SSL) final state: Phishing websites may use HTTPs protocol. This is a warning to end users letting them know that the site is not secured by SSL.

3) Popup windows: Usually, legitimate sites do not ask users their credentials via popup windows.

4) Request URL: Often, in legitimate websites, objects are loaded from the same domain where the webpage is loaded.

5) URL of the anchor: The hypertext reference is used to specify a target for the anchor element. If the anchor points to a different domain rather than the domain where the webpage is loaded, then the website is suspicious or phishing.

6) Web traffic: High web traffic indicates that website is used regularly and is likely to be legitimate.

7) URL length: Phishing websites often use long URLs so that they can hide the suspicious part of the URL.

8) Age of the domain: Domains that are in service for a longer period of time are likely to be legitimate.

9) Having IP address in the URL: The usage of an IP address in the domain name is an indicator of a non-legitimate website.

10)Class: In this data set, the URLs are categorized into three classes: phishing, suspicious, and legitimate.

B. Classifiers

This work used the above data set to compare the performance of four classifiers. Specifically, we used the decision tree, Naïve Bayes' classifier, SVM, and the Neural Network to classify the URLs in the data set, and then we compared the results using confusion matrices.

1) Decision tree: Decision trees are non-parametric classifiers. As its name indicates, a decision tree is a tree structure, where each non-terminal node denotes a test on an attribute, each branch represents an outcome of the test, and the leaf nodes denote classes. The basic algorithm for decision tree induction is a greedy algorithm that constructs the decision tree in top-down recursive divide-and-conquer manner [12]. At each non-terminal node, one of attributes is chosen for the split. The attribute that gives the maximum information gain is

chosen for the split. A well-known algorithm for decision trees is the C4.5 algorithm where entropy is used as a criterion to calculate the information gain. The information gain is defined as the difference between the entropy before the split and the entropy after the split. Equations to calculate information gain are below.

$$H(T) = -\sum_j p_j \log_2(p_j)$$
$$Hs(T) = -\sum_i p_i Hs(T_i)$$
$$Gain(s) = H(T) - Hs(T) \quad (1)$$

Where $H(T)$ is the entropy before the split, $Hs(T)$ is the entropy after the split, and p_j is probability of class j. One of the main concerns with the decision tree classifier is that it over fits the training data.

2) Naïve bayes' classifier: This classifier calculates the posterior probability for each class and assigns the sample to the class with the maximum probability [13]. The posterior probability for class *i* is given by Equation (2) and can be calculated from the training set data.

$$P(C_i/\mathbf{x}) = P(\mathbf{x}/C_i)P(C_i)$$
$$\text{where } P(\mathbf{x}/C_i) = \prod_{k=1}^{n} P(x_k/C_i) \quad (2)$$

In Equation (2), $P(\mathbf{x}/C_i)$ is a conditional probability.

3) Support vector machine: This classifier uses a nonlinear mapping to transform original training data into a higher dimension and finds hyper planes that partition data samples in the higher dimensional feature space. The separating hyper planes are defined as

$$Wx + b = 0 \quad (3)$$

where W is a weight matrix, and b is a constant. The SVM algorithms find the weight matrix such that it maximizes the distance between the hyper planes separating two classes. Tuples that fall on the hyper planes are called as support vectors [14].

4) Neural network: Neural networks are non-parametric classifiers. Neural networks provide a powerful alternative to statistical classifiers. Neural networks can learn with a training set data and make decisions. We built the neural network using MATLAB script. In particular, we implemented a three layer neural network with a back propagation algorithm [15, 16]. The three layers are the input layer, hidden layer, and the output layer. The number of units in the input layer is equal to the number of features, and the number of units in the output layer is equal to the number of classes. During the learning process, weights in the network are set to small random values. For each training sample, input values are propagated, and the output values at the last layer is compared with the target values to calculate the error. The backpropagation algorithm is a well-known algorithm. It uses a gradient descent method to find the minimum. The error is propagated backward to update the weights so that with each iteration, the Mean Squared Error (MSE) decreases. The iterations are terminated when the MSE is less than some constant e_{min} or the number iterations exceeds the maximum set value. The backpropagation learning algorithm can be described in the following steps.

Step 1: Initialize weights with small random values.

Step 2: Present an input vector and make a forward pass to compute weighted sums S_i and activations $o_i = f(S_i)$ for each unit, where $f(.)$ represents the activation function.

Step 3: Backpropagation: Starting with the output units, make a backward pass through output units and hidden layer units using Equations 3 and 4.

$$f'(S_i) = o_i(1 - o_i) \quad (3)$$

$$\delta_i = \begin{cases} (t_i - o_i) f'(S_i) & \text{for units in output layer} \\ \left(\sum_m w_{m,i}\delta_m\right) f'(S_i) & \text{otherwise} \end{cases} \quad (4)$$

In Equations (3) and (4), $w_{m,i}$ represents weights, and t_i represents target output.

Step 4. Update the weights using Equation 5 where α is a learning rate.

$$w_{i,j} \leftarrow w_{i,j} + \alpha \delta_i o_i \quad (5)$$

Repeat Steps 2 through Step 4 until the MSE is less that e_{min} for all samples in the training set.

IV. Results and Conclusions

A. Results

The data set we used from The University of California, Irvine Machine Learning Repository has nine attributes and contains 1,353 samples. The histogram for the first attribute is shown in Fig. 1. It can be seen that there are three peaks in the histogram that represents three classes. The architecture of the neural network is shown in Fig. 2. There are nine units in the input layer, one for each feature. The hidden layer consists of ten units, and the output layer has three units. The three units in the output layer represent the three classes. Thus, target vectors {1, 0, 0}, {0, 1, 0}, and {0, 0, 1} represent the three classes, which are phishing, suspicious, and legitimate, respectively.

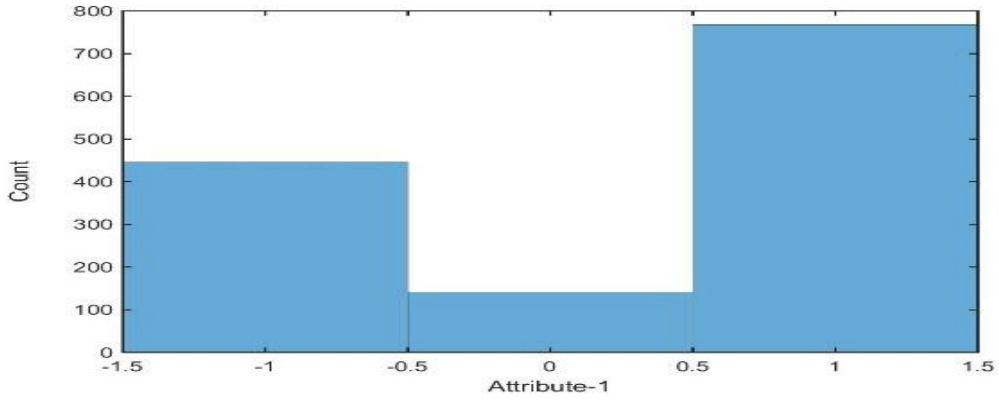

Fig. 1. Histogram for Attribute 1.

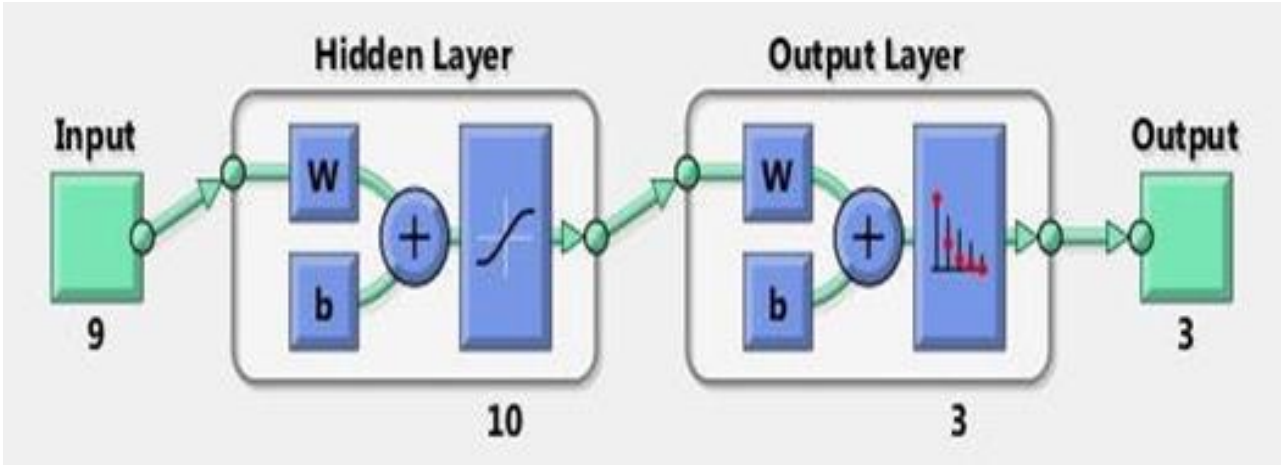

Fig. 2. A Three-Layer Neural Network.

Sixty percent of the samples were selected randomly for training the neural network. of the remaining forty percent, twenty percent were used for validation, and the other twenty percent were used for testing. The graph for the mean squared error during training states is shown in Fig. 3. for the decision tree, Naïve Bayes' classifier, and SVM, forty percent of the records were randomly selected records for training, and the remaining sixty percent were used for testing. The pruned decision tree is shown in Fig. 4. The confusion matrix obtained with the pruned decision tree class is shown in Table 1. We used the same data set as a benchmark and compared the results of all of the classifiers. The results compared included the overall accuracy, True Positive Rate (TPR), and False Positive Rate (FPR) for phishing URL samples. The results of the tests are shown in Table 2.

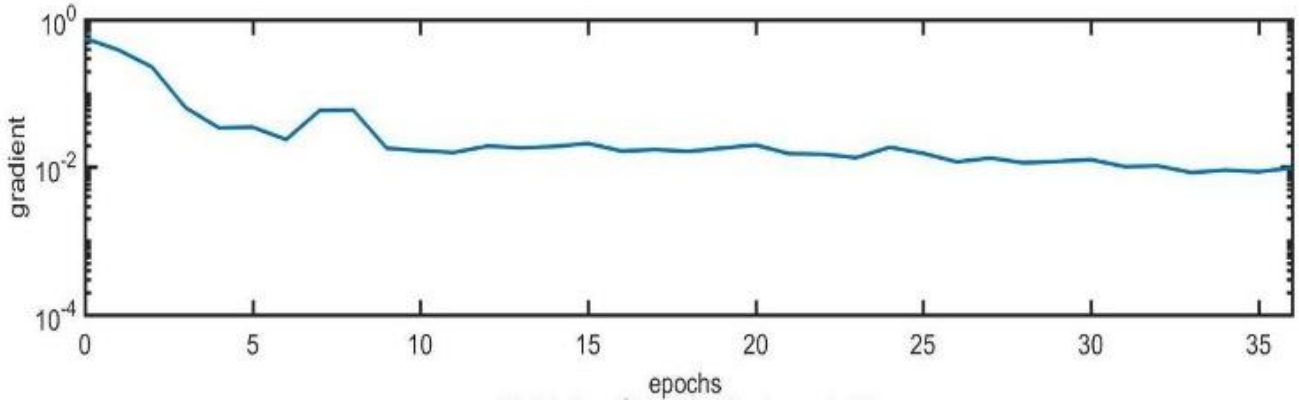

Fig. 3. Mean Squared Error.

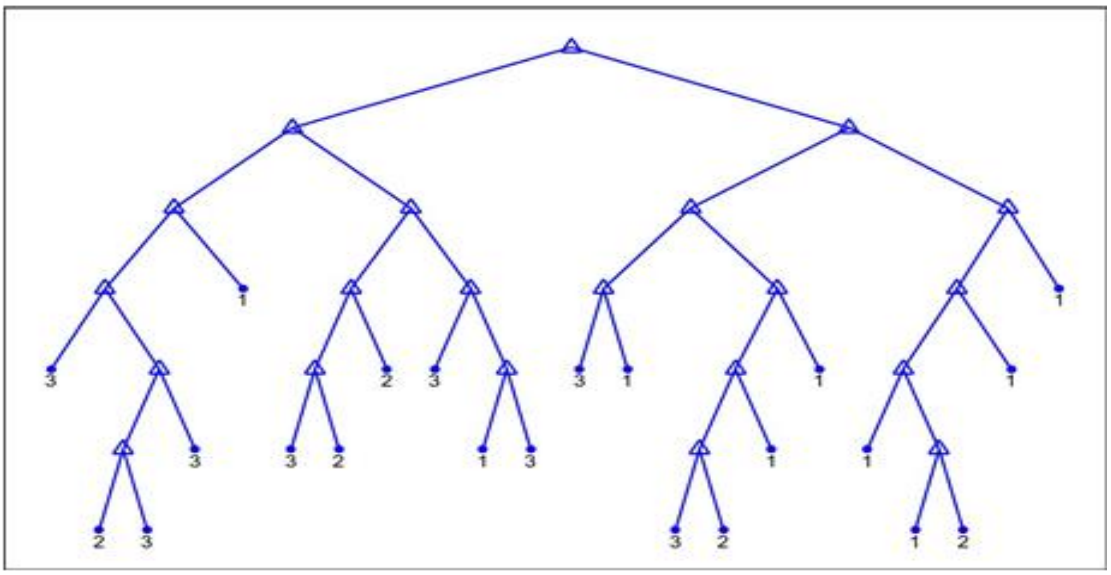

Fig. 4. A Pruned Decision Tree.

B. Conclusions

In this work, we implemented four classifiers using MATLAB scripts, which are the decision tree, Naïve Bayes' classifier, Support Vector Machine (SVM), and the Neural Network. The classifiers were used to detect phishing URLs. In detecting phishing URLs, there are two steps. The first step is to extract features from the URLs, and the second step is to classify URLs using the model that has been developed with the help of the training set data. In this work, we used the data set that provided the extracted features. The data set, from The University of California, Irvine Machine Learning Repository, contained nine features.

One of the main concerns in the decision tree classifiers is over fitting. Generally, the decision tree classifies the training set data very well but yields poor results with a testing dataset. It is often required to prune the decision tree to work well with testing data. The pruned decision tree provided the highest classification accuracy 90.39 percent. with more features in the data set it may be possible to obtain higher accuracy. In addition, the accuracy may be improved by using an ensemble of trees.

It can be seen from Table 2 that the neural network classifier yielded the lowest accuracy for this data set. One of the reasons that it did not perform well is that feature vector values were discrete which results in non-smooth decision boundaries that separate the three classes. However, it is possible to use more number of units in the hidden layer or use deep learning techniques such as adding a number of hidden layers to improve the performance of the network.

TABLE. I. CONFUSION MATRIX USING A DECISION TREE

	Phishing	Suspicious	Legitimate
Phishing	262	3	23
Suspicious	7	34	1
Legitimate	13	5	193

TABLE. II. RESULTS FOR THE TESTED CLASSIFIERS

Classifier	**TPR**	**FPR**	**Accuracy**
Pruned Decision Tree	90.97 %	7.81 %	91.5 %
SVM	90.97 %	18.18 %	86.69 %
Naïve Bayes' Classifier	88.19 %	16.21 %	86.14 %
Neural Network	85.61 %	15.91 %	84.87 %

V. FUTURE WORK

The research work presented here has some limitations and it can be extended further. The first limitation is that we considered a small data set that contains 1353 URLs, and there are 9 features for each URL. The second limitation is that all features are discrete. Often, classifiers such as decision trees, Naïve Bayes classifier, and rule-based systems are more suitable when features are discrete. Furthermore, we used features that were already extracted from URLs. The present work can be extended as below.

We can evaluate classifiers using a large data set that contains a few thousands of URLs and extract more number of features that may be significant in decision making. Larger data sets are available in public domain.

We can generate associative rules using the frequent item data sets with the minimum support and confidence values and build a rule-based system using associative rules to classify URLs. The rule-based classifier then can be compared with other classification methods. another approach for generating classification rules from data samples is to divide the feature space using fuzzy membership functions and extract and optimize classification rules [17]. The extracted rules can be used to build a fuzzy inference system that can classify URLs.

In order to avoid the problem of overfitting a classifier, we need to include a pre-process stage. In processing, we can use clustering to find out outliers or noisy data samples. Such samples should not be used in the training set data.

REFERENCES

[1] N. Lord, "What is a Phishing Attack? Defining and Identifying Different Types of Phishing Attacks". https://digitalguardian.com/blog/what-phishing-attack-defining-and-identifying-different-types-phishing-attacks, 2018.

[2] N. Sadeh, A. Tomasic, and I Fette, "Learning to detect phishing emails", Proceedings ofthe16thinternational conference on world wide web, pp.649–656, 2007.

[3] J. Ma, S. S. Savag, G. M. Voelker, "Learning to detect malicious URLs", ACM Transactions on Intelligent Systems and technology, vol. 2, no. 9, pp 30:1-30:24, 2011.

[4] S. Purkait, "Phishing counter measures and their effectiveness–literature review", Information Management & Computer Security, vol. 20, no. 5, pp. 382–420, 2012.

[5] N. Abdelhamid, A. Ayesh, F. Thabtah, "Phishing Detection based Associative Classification", Data Mining. Expert Systems with Applications (ESWA), vol. 41, pp 5948-5959, 2014.

[6] D. R. Patil and J. Patil, J., "Survey on malicious web pages detection techniques", International Journal of u-and e-Service, Science and Technology, vol. 8, no. 5, pp. 195–206, 2015.

[7] W. Hadi, F. Aburrub, and S, Alhawari, "A new fast associative classification algorithm for detecting phishing websites", Applied Soft Computing vol. 48, pp 729-734, 2016.

[8] R. K. Nepali and Y. Wang, Y., "You look suspicious!! Leveraging visible attributes to classify malicious short urls on twitter", 2016 49th Hawaii International Conference on System Sciences (HICSS). IEEE, pp. 2648–2655, 2016.

[9] M. Kuyama, Y. Kakizaki, and R. Sasaki, "Method for detecting a malicious domain by using whois and dns features", The Third International Conference on Digital Security and Forensics (DigitalSec2016), p. 74, 2016.

[10] D. Sahoo, C. Liu, and C. H. Hoi, "Malicious URL detection using machine learning: A Survey", https://arxiv.org/abs/1701.07179, 2017.

[11] UCI Machine Learning Repository: Website Phishing Data Set (Online) https://archive.ics.uci.edu/ml/datasets/Website+Phishing.

[12] J. R. Quinlan, "Induction of Decision Trees", Machine Learning, vol. 1, no. 1, pp. 81–106, 1986.

[13] J. Han and M. Kamber, "Data Mining–Concepts and Techniques", Morgan Kaufmann, San Francisco, CA, 2011.

[14] V. N. Vapnik, "Support-vector networks", Machine Learning, vol. 20 no. 3, pp 273–297, 1995.

[15] R. P. Lippman, "An introduction to computing with neural nets". IEEE ASSP Magazine, vol. 3 n,o. 4, pp. 4-22, 1987.

[16] S. L. Gallant, "Neural network learning and expert systems", The MIT Press, Cambridge, MA, 1993.

[17] A.D. Kulkarni, "Generating classification rules from training samples", International Journal of Advanced Computer Science Applications, vol 9, no. 6, pp 1-6.