Malicious URL Detection Based On Lexical Features Using Machine Learning

Abdullah Ahmad
Department of Computer Science
Air University
Islamabad, Pakistan
abuba831@gmail.com

Muhammad Hasher
Department of Computer Science
Air University
Islamabad, Pakistan
muhammadhasher99@gmail.com

***Abstract*—Malicious URLs host unsought content and are responsible for most of the cybercrimes. Conventionally, this is done through the usage of blacklists, which cannot be comprehensive and cannot detect freshly produced spam URLs. With the constant growth of cyber-attacks on the web, there is a dire need for programs to detect infected web pages. In the everyday world, in a malicious URL detection job, the ratio between the number of malicious URLs and authentic URLs is highly imbalanced, making it very unsuitable for simply improving the prediction correctness. The most widespread and common previous studies have performed URL classification using extracting lexical attributes of the URL string, and by mining bag-of-words attributes, afterward applying machine learning models. In this paper, the new features have been extracted from the URLs, and different machine learning classification algorithms were applied and compared to determine whether the URL is malicious or not. Among these machine learning models, Random Forest offered the highest F-1 score of 95.86%.**

***Index Terms*—Malicious URLs, Lexical Features, Machine Learning, Classification**

I. Introduction

The evolution in Word Wide Web (WWW) is become faster in the past decade. The 4G and 5G technology makes more convenient for ordinary person to access high speed internet via smart phones. The availability of economical and high internet speed strengthens people to enjoy online social networking, e-commence, jobs and many more. Moreover, the current pandemic situation (COVID-19) enlighten the impact and significance of internet in our daily life.

However, there are uncountable positive aspect of internet, the dark web is also evolving exponentially. The website which contain spamming and malicious content become more challenging to handle. We live in a time where cyber warfare becoming more popular. Malicious websites are accountable for most of the cyber-attacks and scams today [16]. Moreover, the web has influenced and supported many criminal exercises such as spam advertising, financial fraud, and spreading malware. Most of these exercises are being spread and influenced to the internet users using Uniform Resource Locator (URL). Each time user clicks an unfamiliar link there is a serious threat or risk. Among other medium for spreading malicious content the email, links on other web pages, and web search results are prominent.

The aforementioned challenges are the motivation of this study to investigate on a question "What if the user enable to estimate in advance that a specific URL is malicious or not to visit?". To address this question researchers have developed various systems to protect users from their uninformed choices. Initially the dictionary and rule based systems were proposed. These systems were basic and only able to detect known URLs. Later on, more robust systems were proposed that able to label the websites using other information such as combinations of user feedback, manual reporting, honeypots, web crawlers and heuristic analysis on available information. However,it requires very less overhead the risk still exist that a user may click on a malicious URL before it appears on a blacklist [1][15]. Unfortunately, many new malicious websites are not blacklisted because they were evaluated inaccurately. Some user-side systems analyze the behavior and content of websites to address the malware, but this approach failed because it requires far more run-time overhead, and users can be exposed to the browser-based intrusions that we aimed to avoid.

In recent years machine learning based models overcome the above mentioned limitations to great extent. The machine learning models are frequently available in current studies to mitigate this challenge. However, the challenge still exist as complex pattern of URL are unable to detect easily. Therefore, it is need to further investigate this problem in different paradigm and provide more accurate solution.

The scope of this study is only to focus on the classification of the URL itself as malicious or not without any extra information. To achieve this objective the different lexical-based features of the URL that characterize them has been investigated. The classification model has been trained on different lexical features identified from current state-of-the-art techniques. The six different well-known binary classification models (Naive Bayes, Decision tree, Random forest, Logistic Regression, K-Nearest Neighbor and Support Vector Machine) have been trained, and test to detect malicious URLs. Among these machine learning models Random Forest offered the highest F-1 score of 95.86%.

The proposed approach has been evaluated on two different

datasets. The datasets contain 450,176 (77% benign and 23% malicious) and 411,247 (82% benign and 18% malicious) labeled URL's respectively.

II. Related Work

Past work on this topic has included content examination of the page itself [2]. These ordinarily include making features from the HTML structure of the page, joins, and anchor content, such as the number of words on a page, mean of length if a word, and the quantity of words in a title. More strategies include looking at the sum and rate of concealed content (not obvious to a client) on a page.

Another work endeavor to classify web spam into buckets, such as link spam, redirection, cloaking, and keyword stuffing [3]. Whereas part spam into more particular buckets will likely lead to enhancements in classifier capacity. This paper will center on building a common classifier for all sorts of spam.

Historically the well-known methods for identifying malicious web pages was the implementation of searchable blacklists like the Google Safe browsing interface. Although this approach of blacklisting can quickly become outdated as scammers change the domain rapidly. This technique takes many hours and days to update therefore it has failed in the detection of phishing links. Blacklist features have also been shown to be less effective than URL lexical analysis [4].

In 2009, a bunch of researchers proposed a classification model with URL lexical features. This model was based on binary bag-of-words approach from which they extracted 369,585 features. Although this work was based on previous research [5] with the major difference is the absence of host-based features.

To detect phishing attacks by training classifiers has been the topic of many researchers. Another approach is to merge host features with URL lexical analysis to construct a classifier [4]- [6], [7]. Mainly the host-based attributes include IP address, domain name, geo-location, and WHOIS properties.

In 2011, Prophiler was invented, which is a fixed classifier that acts as a filter for a dynamic classifier. It filters the benign pages, meanwhile preserving the primary concern of low false negatives [8]. This technique decreases the computational burden on a honey client.

Another approach is, to begin with, deciding what are vital features in terms of ranking in a search engine and after that discover which features are likely to be utilized by spammers [9]. The drawback to this approach is that it is impracticable to identify each positioning component and hence critical features may be missed.

In 2015, research proposed a light weighted system that detects malicious URLs based on static lexical features with an accuracy of 99.1% having average speed of 0.627 milliseconds [10]. The drawback in this research was the system rejects shortened URLs.

In 2019, another approach was presented using the Random Forest model for URL classification. This study was based entirely on lexical features extracted from URL string [11]. Similarly, In 2020 a study was published of comparative analysis of multiple models using deep learning techniques. Among 5 models, two are based on RNN, two are based on CNN, and one is based on hybrid CNN and LSTM [12]. However deep learning based model improve the detection accuracy to some extent for malicious URL detection, it required high computation cost. Therefore, it is dire need to further explore and fine tuned the light weight machine learning models and improve their accuracy.

While depending on the page content and links increase the amount of data accessible for spam classification, there are solid inspirations for being able to classify spam earlier than crawling a page. This paper explores utilizing the URL string as the essential feature in malicious classification.

III. Methodology

While there are a variety of features that one can use to classify if a URL is safe or not. The aim of this research is to use only the URLs and limited metadata information to classify if web pages are malicious or not [17]. As mentioned earlier, this choice is made for performance reasons, as scraping HTML from web pages is resource-intensive and not useful since the page must have already been crawled. This section provides the complete methodology and our approach towards the solution of this problem.

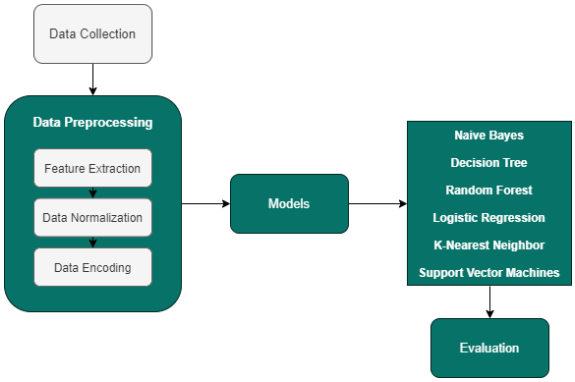

Fig. 1. Proposed Methodology Diagram

A. Data Preprocessing

1) Feature Extraction: We categorize the features that we gather for URLs as being lexical-based. Lexical features permit us to apprehend the property that differentiate malicious URLs from benign URLs. Lexical features are the textual properties of the URL itself [4]. The word lexical itself describes that we aimed to tokenize the URL and generate new features from it, also known as feature engineering. For instance, we separate the two parts of a URL: the path and host name. As an example, with the URL www.wikipedia.org/wiki/URL, the host name part is www.wikipedia.org and the path segment is wiki/URL. The length of hostname and length of path are lexical features.

Similarly, a total of 19 features of three types were extracted, which includes

- Length Features

- Binary Features
- Count Features

Length Features: A total of five length features are extracted from the URLs. Length features define the length of a certain pattern in a URL. The extracted length features are: 'url_length', 'hostname_length', 'path_length', 'fd_length', 'tld_length'.

Binary Features: Binary features describes whether the existing pattern exists in the URL or not. A total of two binary features were implemented. The first extracted feature is whether the URL contains an IP address or not. Whereas, the second extracted feature is whether the given URL is a shortened version or not.

Count Features: A total of twelve count features are extracted. Count features are the occurrences or frequency of a character or word in a URL. The extracted characters for count features are: 'dash -', 'address @', 'question mark ?', 'percentage %', 'dot .', 'equal =', 'https', 'http', 'www', 'digits', 'letters', 'number of directories'.

TABLE I
FEATURES DESCRIPTION

Type	*Feature Name*	*Feature Description*
Length	url_length	Length of URL
Length	hostname_length	Length of host name of URL
Length	path_length	Length of path
Length	fd_length	Length of first directory
Length	tld_length	Length of top level domain
Binary	contains_ip	URL contains an IP or not
Binary	shorterned_url	Short URL exists or not
Count	count_dash	Frequency of '-' in a URL
Count	count_address_sign	Frequency of '@' in a URL
Count	count_question	Frequency of '?' in a URL
Count	count_percent	Frequency of '%' in a URL
Count	count_dot	Frequency of '.' in a URL
Count	count_equal	Frequency of '=' in a URL
Count	count_https	Frequency of string 'https' in a URL
Count	count_http	Frequency of string 'http' in a URL
Count	count_www	Frequency of string 'www' in a URL
Count	count_digits	Number of digits in a URL
Count	count_letters	Number of letters in a URL
Count	count_no_of_dir	Number of directories in a URL

2) Data Normalization: The machine learning models tend to work better when features are on a relatively closer or on a similar scale. It also helps in tackling the outliers. Therefore, in this research min-max normalization is performed on the data. Min-max returns the values ranging between 0 to 1. The equation for min-max normalization is given below.

$$Z = \frac{X - min(X)}{max(X) - min(X)} \tag{1}$$

3) Data Encoding: It is needed to transform the categorical i.e. text features to their numeric form. Label encoding encodes data with a value between 0 and n-1 where *n* is the number of unique class labels. If a label is repeated, it allocates the same value to that label that was allocated earlier. The target variable has been converted into its numeric representation.

B. Classification Algorithms

In this research different machine learning classification algorithms were used such as Naive Bayes, Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbor, and Support Vector Machines to detect the malicious URLs. All these algorithms were fitted on the training data to analyze which algorithm will work the best on our testing data.

1) Naive Bayes: It is a probabilistic predictive modeling technique based on Bayes theorem with an assumption of independence between each input feature.

$$P(Y|Z) = \frac{P(Z|Y).P(Y)}{P(Z)} \tag{2}$$

Y and Z denotes the events. $P(Y|Z)$ implies that the probability of A given B is true whereas, $P(Z|Y)$ implies that the probability of B given A is true. $P(Y)$, and $P(Z)$ are independent probabilities of each event.

In simple terms, a Naive Bayes classifier assumes that the existence of a specific feature in a class is unrelated or independent to the existence of any other attribute.

2) Decision Tree: A decision tree is the graphical presentation of all the achievable solutions to a decision. First, we compute entropy for the entire dataset. After that, we will decide which node to decide as a root node. So, one by one calculate entropy for each feature then calculate information gain for each feature. In the end, we will select the feature with maximum value and call it our root node. So, in other words, a decision tree makes binary splits of the data using the features and cut off points that lead to the greatest possible reduction in the loss in a greedy manner.

3) Random Forest: Random forest is a method that operates by constructing multiple decision trees using a method called bagging and outputs the class that is the mode of individual trees. This method first split the data randomly. Then decision trees are fit on each sub-sample and their output is averaged over all decision trees. Random forest compiles the result from all decision trees and leads towards an outcome. Random forest reduces the risk of variance. The random forests are efficient because the large number of relatively uncorrelated trees working as a group will transcend any of the single constituent models.

4) Logistic Regression: Logistic regression assigns the observations to a unique or seperate set of classes. It makes a prediction based on probability, the prediction will fall in the range of [0,1]. It is based on sigmoid function.

$$h\theta(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \tag{3}$$

5) K-Nearest Neighbor: K-Nearest neighbor is a non-linear classifier, it is relatively simpler yet very effective technique to classify data. It simply calculates the distance between the nearest number of k data points, and assign that point to the closest class. The distance is calculated using Euclidean distance formula which states

$$d(p, q) = \sqrt{\Sigma(qi - pi)^2} \tag{4}$$

6) Support Vector Machine: Support Vector Machine is a classification, linear model that generates a hyperplane or line which splits the data into different classes. A hyperplane is an n-dimensional Euclidean space is a flat subset with dimension. The intuition behind the algorithm is that it finds the points nearest to the line from the classes which are known as support vectors, and the distance between these lines and support vectors are known as margin. The idea is to maximize the margin, which would be the optimal hyperplane.

IV. Experiments

In this research, multiple classification algorithms are implemented and compared against each other. Various experiments are performed to determine the best classifier to detect malicious URLs. The above section summarizes various metrics like Accuracy, Precision, Recall, F-1 score which were used to compare between the algorithms.

A. Dataset Description

We drew our data from Kaggle. The performance of different machine learning algorithms is analyzed on two different datasets. First dataset consists of 450,176 different URLs [1]. In this dataset 77% of URLs were benign and 23% were malicious. Second dataset contains 411,247 URLs. In this dataset 82% of URLs were benign and 18% were malicious [2]. After merging, the total frequency of benign URLs were 690,559 (79.3%) whereas, 180,081 malicious URLs (20.7%) are displayed below. This shows that the dataset is highly imbalanced.

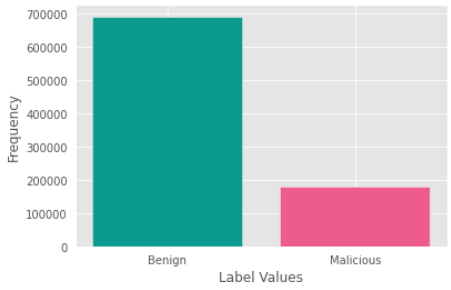

Fig. 2. Bar Plot of Target Variable

B. Performance Evaluation Metrics

In order to evaluate the performance of the trained classification model, various performance metrics were used such as Accuracy, Precision, Recall, and F-1 Score. These evaluation metrics are very useful for evaluating an imbalanced classification problem. The evaluation metrics used in this research are defined below.

- Accuracy: It shows the number of accurately predicted data points.
- Precision: It is a proportion of correct positive classification from cases that are predicted as positive.
- Recall: It is a proportion of actual positive classification from cases that are identified correctly.
- F1 Score: Formally F score can be called the F1 score or F measure. It is a measure of a test's accuracy. F1 score is generally a function of Precision and Recall.

[1]https://www.kaggle.com/siddharthkumar25/malicious-and-benign-urls
[2]https://www.kaggle.com/antonyj453/urldataset

C. Results

TABLE II
Classification Model Evaluation

Classification Algorithms	**Evaluation Metrics**			
	Accuracy	***Precision***	***Recall***	***F-1 Score***
Naive Bayes	81.82	80.50	81.82	77.12
Decision Tree	95.32	95.27	95.32	95.28
Random Forest	**95.91**	**95.86**	**95.91**	**95.86**
Logistic Regression	91.47	92.05	91.47	90.64
K-Nearest Neighbor	95.26	95.23	95.26	95.14
Support Vector Machine	91.76	92.40	91.76	90.97

From table II, it can be observed that random forest out performs the other classifiers. Accuracy achieved using random forest classifier is 95.91%, precision, and recall scores are 95.85%, and 95.91% respectively. Achieved F-1 score is 95.86%. Meanwhile decision tree is the second best classifier with an accuracy of 95.32%, 95.27% precision, 95.32% recall score, and 95.28% F-1 score. K-Nearest neighbor is a close third, achieving 95.26% accuracy, 95.23% precision score, 95.26% recall score, and 95.14% F-1 score. Then, support vectors outperforms logistic regression. Accuracy achieved on support vector is 91.76%, and 91.47% on logistic regression. The precision score on support vector is 92.40% compared to 92.05% of logistic regression. 91.76% and 91.47% recall score respectively. Similarly, the F-1 score using support vectors is 90.97%, compared to 90.64% of logistic regression. Lastly, naive bayes performed the worst with the accuracy score of 81.82%, precision score of 80.50%, recall score of 81.82%, and 77.12% F-1 score.

TABLE III
Per Class F-1 Score

Labels	***NB***	***DT***	***RF***	***LR***	***KNN***	***SVM***
Benign	90	**97**	**97**	95	**97**	95
Malicious	29	88	**90**	74	88	75

The F-1 score shows the better picture for imbalanced classification models. The best F-1 score for benign class achieved is 97% using random forest, decision tree, and k-nearest neighbors. Whereas, logistic regression and support vectors achieved the score of 95%, and naive bayes has a F-1 score of 90%. For malicious class, it can be observed that random forest outperformed other classifiers by achieving the F-1 score of 90%. Decision tree and k-nearest neighbors has a F-1 score of 88%. Meanwhile, support vector classifier and logistic regression achieved 75% and 74% F-1 score respectively. Whereas, the F-1 score attained by naive bayes is 29%.

The results shows that tree-based and non-linear classifiers performed better than linear classifiers. The performance of random forest is pre-eminent, then decision tree and KNN performed equally. Tree-based classifiers maps non-linear relationships quite well. The reason why tree-based and non-linear classifiers performed better because of the non-linearity, complex relationship between dependent and independent variables.

The reason that random forest outperformed decision trees and KNN is because it has methods for balancing errors in datasets where dependent variable classes are imbalanced, and it has the power to handle large and complex data to identify the most significant variables with higher dimensionality. The only drawback of using random forest is the computational cost, it takes alot of time to train. Also, the training time increases as the number of features increases because it determines how many number of trees should be built.

V. Conclusion

With the increasing number of websites on the internet, it is becoming increasingly important to identify the malicious and spam websites to improve the security, and quality for users.

In this research, a lightweight model to identify malicious URLs is presented. In order to improve the predictive performance, various steps were performed. Firstly, lexical features were extracted from raw URLs, and were trained on multiple algorithms to classify the URLs as benign or malicious.

Furthermore, six different classification algorithms were compared and it is observed that random forest classifier outperforms other algorithms. The accuracy achieved using random forest is 95.91%, precision and recall score is 95.86% and 95.91%. F-1 score is 95.86%.

Besides, the proposed approach showed great performance, there is still room for improvement. As of late, ideas like deep neural networks have been displayed to increase the performance of classification algorithms radically. Therefore, deep learning algorithms can be explored as part of future work.

References

[1] Sinha, Sushant, Michael Bailey, and Farnam Jahanian. "Shades of Grey: On the effectiveness of reputation-based "blacklists"." 2008 3rd International Conference on Malicious and Unwanted Software (MALWARE). IEEE, 2008.

[2] Gyongyi, Zoltan, and Hector Garcia-Molina. "Web spam taxonomy." First international workshop on adversarial information retrieval on the web (AIRWeb 2005). 2005.

[3] Ntoulas, Alexandros, et al. "Detecting spam web pages through content analysis." Proceedings of the 15th international conference on World Wide Web. 2006.

[4] Ma, Justin, et al. "Beyond blacklists: learning to detect malicious web sites from suspicious URLs." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. 2009.

[5] Ma, Justin, et al. "Identifying suspicious URLs: an application of large-scale online learning." Proceedings of the 26th annual international conference on machine learning. 2009.

[6] Xiang, Guang, et al. "Cantina+ a feature-rich machine learning framework for detecting phishing web sites." ACM Transactions on Information and System Security (TISSEC) 14.2 (2011): 1-28.

[7] Thomas, Kurt, et al. "Design and evaluation of a real-time url spam filtering service." 2011 IEEE symposium on security and privacy. IEEE, 2011.

[8] Canali, Davide, et al. "Prophiler: a fast filter for the large-scale detection of malicious web pages." Proceedings of the 20th international conference on World wide web. 2011.

[9] Egele, Manuel, Clemens Kolbitsch, and Christian Platzer. "Removing web spam links from search engine results." Journal in Computer Virology 7.1 (2011): 51-62.

[10] Darling, Michael, et al. "A lexical approach for classifying malicious URLs." 2015 international conference on high performance computing simulation (HPCS). IEEE, 2015.

[11] Joshi, Apoorva, et al. "Using Lexical Features for Malicious URL Detection–A Machine Learning Approach." arXiv preprint arXiv:1910.06277 (2019).

[12] KP, Soman, and Mamoun Alazab. "Malicious URL Detection using Deep Learning." (2020).

[13] Le, Hung, et al. "URLnet: Learning a URL representation with deep learning for malicious URL detection." arXiv preprint arXiv: 1802.03162 (2018)

[14] Classifying Spam using URLs, Di Ai Computer Science Stanford University Stanford, CA, Autumn 2018.

[15] Sahoo, Doyen, Chenghao Liu, and Steven CH Hoi. "Malicious URL detection using machine learning: A survey." arXiv preprint arXiv:1701.07179 (2017).

[16] Thatte, Revati. "Is cyber security the biggest technological challenge that we are facing today?." Sea (2020).