ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/350931304

Malicious URL Detection: A Comparative Study

Conference Paper · March 2021

DOI: 10.1109/ICAIS50930.2021.9396014

CITATIONS
3

READS
2,046

3 authors, including:

Shantanu Maheshwari
National Institute of Technology Tiruchirappalli
1 PUBLICATION **3** CITATIONS

SEE PROFILE

All content following this page was uploaded by Shantanu Maheshwari on 28 June 2021.

The user has requested enhancement of the downloaded file.

Proceedings of the International Conference on Artificial Intelligence and Smart Systems (ICAIS-2021)
IEEE Xplore Part Number: CFP21OAB-ART; ISBN: 978-1-7281-9537-7

Malicious URL Detection: A Comparative Study

1st Shantanu
Data Analytics
Department of Computer Application
National Institute of Technology
Tiruchirappalli, India
shantanumaheshwari199@gmail.com

2nd Janet B
Department of Computer Application
National Institute of Technology
Tiruchirappalli, India
janet@nitt.edu

3rd Joshua Arul Kumar R
Deparment of ECE
MAM College of Engineering
Tiruchirappalli, India
joshuamamce@gmail.com

2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) | 978-1-7281-9537-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICAIS50930.2021.9396014

***Abstract*—Malicious uniform resource locator (URL), i.e., Malicious websites are one of the most common cybersecurity threats. They host gratuitous content (spam, malware, inappropriate ads, spoofing, etc.) and tempt unwary users to become victims of scams (financial loss, private information disclosure, malware installation, extortion, fake shopping site, unexpected prize etc.) and cause loss of billions of rupees each year. The visit to these sites can be driven by email, advertisements, web search or links from other websites. In each case, the user must click on the malicious URL. The rising cases of phishing, spamming and malware has generated an urgent need for a reliable solution which can classify and identify the malicious URLs. Traditional classification techniques like blacklisting, regular expression, and signature matching approach are challenged because of huge data volume, patterns and technology changing over time, along with complicated relationship among features. In this paper, we address the detection of malicious URLs as a binary classification problem and evaluate the performance of several well-known machine learning classifiers. We adopted a public dataset from Kaggle comprising of 450000 URLs to train the model. The best classifier was used to detect malicious URLs from openphish website. It was found to give better results.**

***Index Terms*—Malicious URL, Machine learning, Phishing, Spamming, Malware, Spoofing.**

I. Introduction

The Covid 19 has a great impact on the growth of online businesses such as e-banking, e-commerce, and social networking. Unfortunately, the technological advancements accompany state of the art techniques to exploit users. Such attacks generally include malicious websites that steal all kinds of private information that a hacker can exploit. In Malicious URL detection, traditional classification techniques like blacklisting [1], regular expression [2], and signature matching [3] approaches are challenged because of huge data volume, patterns changing over time, and complicated relationship among features. Inevitably, several malicious sites do not seem to be blacklisted. As any file on a computer is to be found by giving its filename, similarly, URL can be used to trace any website. It is the address of a resource on the WWW. Each URL has two main components. The first is Protocol. For URL https://www.google.com, the protocol identifier is HTTPS. Hypertext Transfer Protocol Secure (HTTPS) which is used to fetch hypertext documents. Other protocols include File Transfer Protocol (FTP), Domain Name System (DNS) etc. The second is Resource identifier. For URL https://www.google.com, the resource name is www.google.com. The resource identifier is the address of a webpage on the internet. The proposed work in this paper

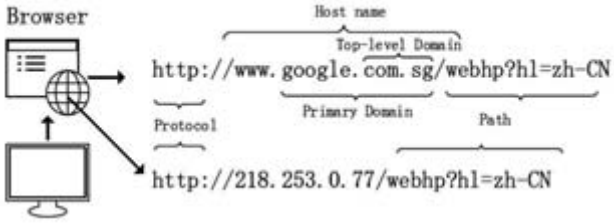

Fig. 1. URL Example

considers the identification of bad URLs and examines the evaluation metrics of various Machine Learning classifiers [4]. The source of data is a public dataset from Kaggle [5] comprising of 450000 URLs. The best classifier is used to detect malicious URLs from the openphish [6] website. The remaining paper is divided into the following sections. Section II describes the URL classification. Section III introduces the machine learning classification techniques used for solving it. The Dataset visualization is given in Section IV. Section V explains the experimental results achieved. Section VI gives the conclusion.

II. Problem Description

URL has been used and misused a lot to exploit the vulnerability of the user. This paper focusses on classification of any URL as benign or malicious. Furthermore, it compares the results of the multiple machine learning classification techniques such as Logistic Regression (LR), Stochastic Gradient Descent (SGD), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbours (KNN), and Decision Tree (DT). The best performing classifier is used to detect malicious websites from the OpenPhish website. The proposed framework has five stages:

- Data Collection: A labelled dataset of malicious and benign websites is collected from the Kaggle repository [5].
- Data Cleaning and Extraction: Pre-processing includes extraction of additional features, normalisation, encod-

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI. Downloaded on June 02,2021 at 07:11:55 UTC from IEEE Xplore. Restrictions apply.

ing of categorical values, standardisation of values and handling of missing data.

- Model Training: Sklearn python library is used for training the model using different machine learning techniques such as Logistic Regression (LR), Stochastic Gradient Descent (SGD), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), K-Nearest Neighbours (KNN), and Decision Tree (DT) on 80% of the data.
- Model Testing and Optimisation: Trained model is tested on the remaining 20 % of the data. Hyperparameters are tuned to increase accuracy, precision, and recall.
- Model Comparison: The machine learning classification techniques are compared based on evaluation metrics.

III. Classification Techniques

Classification [7] is a machine learning process of categorizing the given data into a set of classes. Data can be in both structured and unstructured format. The process includes pre-processing, training the model and categorising data into given classes. The classes are also referred to as targets, categories, or labels. There are two types of classification namely Binomial Classification and Multi-Class Classification. Some of the key areas where classification is used are categorising email spam or ham, classifying tweets as negative or positive sentiments, classifying different images such as fruits, animals, insects, and many more complex tasks.

- Logistic Regression: Linear regression is a linear ML algorithm which is used for classification. In logistic regression, the probabilities of possible classes are calculated using the sigmoid function. The sigmoid function is used because the range of the function is from 0 to 1. Logistic regression is used to understand the relationship between independent and dependent variables. It is easy to implement and most computationally efficient algorithm among those compared in this paper. Logistic Regression can be used in the case of binomial classification. It assumes that the independent variables are uncorrelated. Figure 2 shows the LG model used.

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=100)
lr.fit(X_train, y_train)
lr_y_pred = lr.predict(X_test)
```

Fig. 2. Logistic Regression

- Stochastic Gradient Descent: SGD in an iterative method for stochastically approximating gradient descent optimisation. Its advantages include ease of implementation. It is computationally less expensive as well. Being a linear model, it does not handle the non-linear relationship between dependent and independent variables. It is sensitive to standardisation, normalisation and requires tuning of hyper-parameters. Figure 3 shows the SGD model used.
- Naïve Bayes: Naïve Bayes is a statistical classification model which is largely based on Bayes Theorem. It

```
from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier(n_jobs=-1)
sgd.fit(X_train, y_train)
sgd_y_pred = sgd.predict(X_test)
```

Fig. 3. Stochastic Gradient Descent

presumes that the independent variables have a very low correlation among them. Generally, Naïve Bayes classifiers are linear models, but when Kernel density functions are passed to them, the models can even classify non-linear data with good accuracy. The main advantage of Naïve Bayes is that its learning speed is greater than some of the more complex algorithms. It even requires lesser amount of data compared to other models. The disadvantage is lower accuracy compared to the other machine learning classifiers. Figure 4 shows the Gaussian model used.

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)
nb_y_pred = nb.predict(X_test)
```

Fig. 4. Gaussian Naïve Bayes

- K-Nearest Neighbours: K-Nearest Neighbours is a statistical model of classification. The data points in this model are classified based on the proximity of their neighbours. It is a type of non-parametric model. The number of neighbours is the main hyperparameter passed to the function. The advantages include optimal model that give good accuracy if trained with large data. It can handle noisy data as well. The cost of finding the optimised classification model is high because we must test the model for different values of k which is the number of neighbours. Figure 5 shows the classifier model with K = 5.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
knn.fit(X_train, y_train)
knn_y_pred = knn.predict(X_test)
```

Fig. 5. K-Nearest Neighbours

- Decision Tree: Decision tree classifier constructs a tree for categorising data into classes by generating a set of rules. Splitting of a node in Decision tree is based upon information gain and entropy. Unlike Artificial Neural Networks which are like a black box, decision trees can be visualised and are easily understandable. Numerical and categorical type of data can be used in the Decision tree. Decision trees tend to overfit the data when it is trained too much. A completely different tree could be generated because of slight variations in the data. Figure 6 shows the Decision tree classifier used.
- Random Forest: Random forest classifier belongs to a class of ensemble classifiers which fits numerous decision

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI. Downloaded on June 02,2021 at 07:11:55 UTC from IEEE Xplore. Restrictions apply.

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state=14)
dt.fit(X_train, y_train)
dt_y_pred = dt.predict(X_test)
```

Fig. 6. Decision Tree

trees on various subsets of the data. The final model is based on the average of various trained decision trees. Generally, it performs better than decision trees and even solves the problem of overfitting. It cannot be used in real-time applications as it is computationally expensive to train random forest classifier. It is a complex algorithm to train as well. Figure 7 shows the random classifer model.

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=500)
rfc.fit(X_train, y_train)
rfc_y_pred = rfc.predict(X_test)
```

Fig. 7. Random Forest

- Support Vector Machine: SVMs are supervised learning algorithm for classification that predicts a hyperplane that categorises the data with the maximum margin. New data points are mapped by the side of the hyperplane they lie on. In case of very large data, it is memory efficient compared to other models as it trains on a subset of data. The algorithm does not perform well with a huge dataset that contains noisy data. Figure 8 shows the model.

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
svc_y_pred = svc.predict(X_test)
```

Fig. 8. Support Vector Machine Classifier

IV. Data Visualization

- Data Collection: An open-source labelled dataset consisting of 450,000 websites is collected from Kaggle repository for training and evaluating machine learning models. The data consists of two features: URL and label as shown in figure 9.

	url	label
0	https://www.google.com	benign
1	https://www.youtube.com	benign
2	https://www.facebook.com	benign
3	https://www.baidu.com	benign
4	https://www.wikipedia.org	benign

Fig. 9. Dataset collected

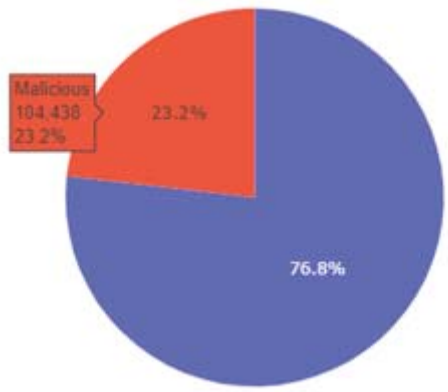

Fig. 10. Dataset analysis

23.2% (104,438) of the complete data are malicious url and the rest are benign urls as shown in figure 10.
The data set is visualized by grouping them. The top 20 domains grouped by domain name on a logarithmic scale is visualized in figure 11. Figure 12 and 13 give the top 20 domains grouped by subdomains and suffix respectively on a logarithmic scale.

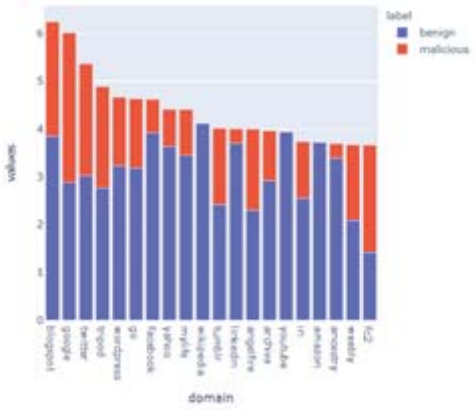

Fig. 11. Top 20 Domains Grouped by domain names

- Feature Extraction: Feature extraction [8] is the process of representing or augmenting features that make machine learning models perform better. It helps in dimensionality reduction facilitating faster processing. Most common approaches are Linear Discriminant Analysis and Principal Component Analysis.
The table I shows the list of features extracted for detecting malicious websites.
The figure 14 gives the correlation between the dependent variables listed in table 1.
The categorical features such as sub-domain, domain, suffix, and target are encoded to numbers, as a machine learning model cannot interpret text directly. The encoding technique used is a count encoder. In the target column, Malicious websites are set to 1 while benign websites are set to 0.

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI. Downloaded on June 02,2021 at 07:11:55 UTC from IEEE Xplore. Restrictions apply.

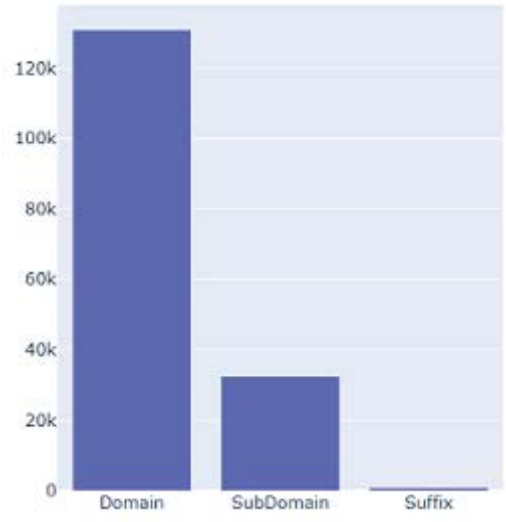

Fig. 12. Top 20 SubDomains Grouped by subdomains

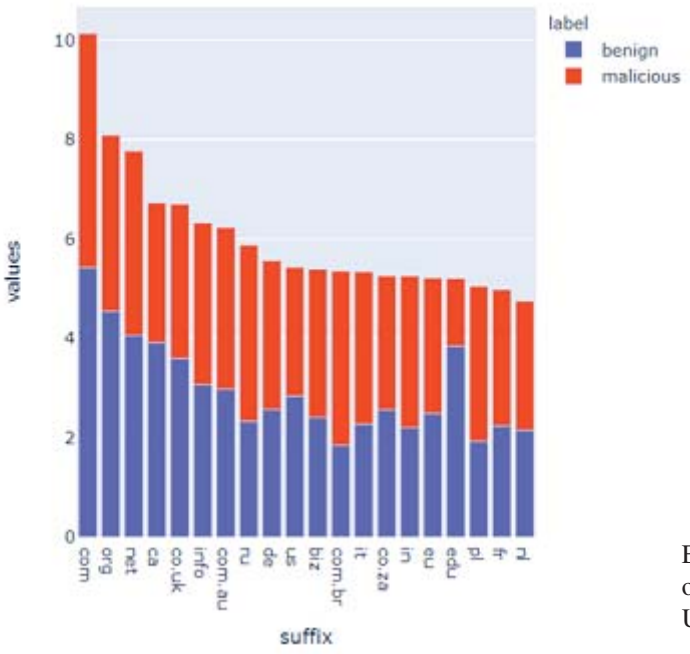

Fig. 13. Top 20 Sub Domains Grouped By Suffix

TABLE I
LIST OF FEATURES EXTRACTED FOR DETECTING MALICIOUS WEBSITES

Features	Features
Suffix	Scheme length
URL length	Path length
Parameter length	Query length
Fragment length	Count of '-'
Count of '&'	Count of '?'
Count of '%'	Count of '.'
Count of digits	Count of alphabets

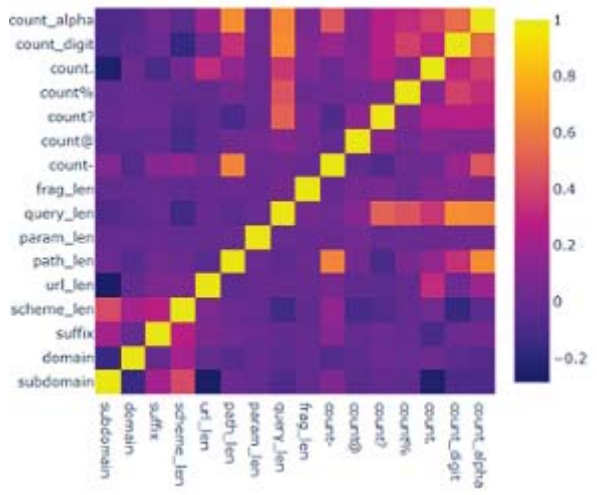

Fig. 14. Correlation between dependent variables

- Feature Scaling is a technique to scale the data features in a fixed range. It is implemented during data pre-processing to handle high variance data. Without data scaling, machine learning model tends to give more importance to higher values and less to lower values. It is one of the most important and time-consuming steps of data pre-processing. The two of the most common techniques are:
  Standardisation: After applying standardisation, the transformed data X has zero mean and unit variance.

$$X^1 = \frac{X - \mu}{\sigma} \tag{1}$$

  Normalization: In this technique, the values are rescaled in the range between 0 and 1.

$$X^1 = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{2}$$

V. EXPERIMENTAL RESULTS

The large amount of data is divided using the 80-20 rule. Each of the model is trained on 80% of the data and tested on the remaining unseen 20% of the data. The GitHub project URL is found in the link [9].

The metrics that are used to evaluate the classification models:

True Positive(TP): The model predicted True and it was True

False Positive(FP): The model predicted True, but it was False

True Negative(TN): The model predicted False and it was False

False Negative(FN): The model predicted False, but it was True

Accuracy: It is the ratio of true values among the total number of values examined.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI. Downloaded on June 02,2021 at 07:11:55 UTC from IEEE Xplore. Restrictions apply.

Precision: It is the ratio of true values among the total number of values predicted as true.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Recall: It is the ratio of predicted true values and the total number of actual true values.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

F1-score: It is the harmonic mean of precision and recall.

$$F_1 = 2\,\frac{Precision * Recall}{Precision + Recall} \tag{6}$$

Using the above metrics different models were trained and tested. The Random forest model gave the best results. The classification report after testing the trained Random forest classifier on openphish data is shown in Figure 15.

	precision	recall	f1-score	support
0	0.00000	0.00000	0.00000	0
1	1.00000	0.92659	0.96190	6307
accuracy			0.92659	6307
macro avg	0.50000	0.46329	0.48095	6307
weighted avg	1.00000	0.92659	0.96190	6307

Fig. 15. Openphish data classification report

Figure 16 shows the metric scores of the models [10] used in this comparitive study.

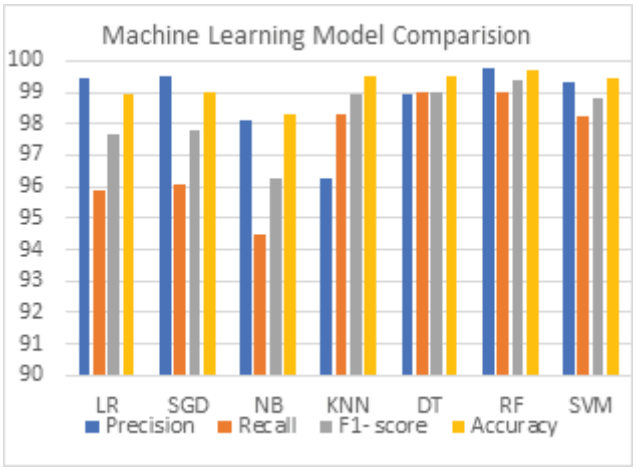

Fig. 16. Machine Learning Model Scores Comparison

VI. Conclusion

The dataset of urls have been visualized. We can infer that the models used attain high prediction accuracy however random forest attains the highest F1 score and accuracy. The accuracy of the trained random forest classifier on openphish data can be increased by training it on a more balanced data i.e., the data containing malicious and benign websites in almost equal proportion. The analysis helps to establish that malicious URL detection is possible by training a model using a database with selected features and using it to predict new phishing attacks.

References

[1] Dhanalakshmi Ranganayakulu, Chellappan C., *Detecting Malicious URLs in E-mail – An Implementation*, AASRI Procedia, Vol. 4, 2013, Pages 125-131, ISSN 2212-6716, https://doi.org/10.1016/j.aasri.2013.10.020.

[2] Yu, Fuqiang, *Malicious URL Detection Algorithm based on BM Pattern Matching*, International Journal of Security and Its Applications, 9, 33-44, 10.14257/ijsia.2015.9.9.04.

[3] K. Nirmal, B. Janet and R. Kumar, *Phishing - the threat that still exists*, 2015 International Conference on Computing and Communications Technologies (ICCCT), Chennai, 2015, pp. 139-143, doi: 10.1109/ICCCT2.2015.7292734.

[4] F. Vanhoenshoven, G. Nápoles, R. Falcon, K. Vanhoof and M. Köppen, *Detecting malicious URLs using machine learning techniques*, 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, 2016, pp. 1-8, doi: 10.1109/SSCI.2016.7850079.

[5] `https://www.kaggle.com/xwolf12/malicious-and-benign-websites` accessed on 27.01.2021

[6] `https://openphish.com/` accessed on 27.01.2021

[7] Doyen Sahoo, Chenghao lua, Steven C. H. Hoi, *Malicious URL Detection using Machine Learning: A Survey*, arXiv:1701.07179v3 [cs.LG], 21 Aug 2019

[8] Rakesh Verma, Avisha Das, *What's in a URL: Fast Feature Extraction and Malicious URL Detection*, ACM ISBN 978-1-4503-4909-3/17/03

[9] `https://github.com/ShantanuMaheshwari/Malicious Website Detection`

[10] Frank Vanhoenshoven, Gonzalo Napoles, Rafael Falcon, Koen Vanhoof and Mario Koppen, *Detecting Malicious URLs using Machine Learning Techniques*, 978-1-5090-4240-1/16 2016, IEEE

Authorized licensed use limited to: NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI. Downloaded on June 02,2021 at 07:11:55 UTC from IEEE Xplore. Restrictions apply.