Malicious Transaction URL Detection Using Logistic Regression

Aratrik Bose, Anandaprova Majumder, and Sumana Kundu

1 Introduction

Machine learning has helped in cyber security domain by letting systems analyze patterns and learn from past security attacks and prevent such attacks by responding to changing behavior. The world has seen significant frauds in the field of E-Commerce as more and more people have started shifting toward cash-free transactions. Since COVID-19 quarantine, people have shifted to online purchase more to stay safe or because the products they need have been unavailable in local shops. However, as we look into the worst part cybercrimes have increased at an alarming rate. With increasing online businesses, Website impersonation can be regarded as one of the easiest forms of cyber-attacks [1]. A common type of Website impersonation attack is typo squatting, where an attacker impersonates a popular Website using a closer variant of the domain name which is to be impersonated (For example, www.@mazon.com instead of www.amazon.com). Hackers usually make a copy of our intended destination by creating a similar interface as that of the original Website so that we do not understand that we are on a different Website. Sometimes, such sites are created to sell products that are in direct competition to the products we are actually looking for but most often such practices are used by hackers to get hold of personal highly sensitive data such as credit card details, CVV numbers or maybe passwords. Such sites are also quite risky as malicious software could be downloaded to our devices simply by visiting these links. So even accepting a download or clicking any link is not necessary for installing a dangerous code on any system.

A. Bose (✉) · A. Majumder · S. Kundu
Computer Science and Engineering, Dr. B.C. Roy Engineering College, Durgapur, West Bengal, India
e-mail: aratrikstudy@gmail.com

A. Majumder
e-mail: anandaprova.majumder@bcrec.ac.in

J. K. Mandal et al. (eds.), *Proceedings of International Conference on Network Security and Blockchain Technology*, Lecture Notes in Networks and Systems 738,
https://doi.org/10.1007/978-981-99-4433-0_8

This is called drive-by download, and many typo squatters use this method to spread malicious software in order to steal our personal information. Environment of online businesses and growth of E-Commerce platforms have also increased occurrences of virtual transactions. As more monetary transactions are being conducted, more monetary frauds are also being conducted side by side. According to RBI data, a total of 4071 fraud cases were reported by Indian lenders between April and September 2021 [2]. To prevent such illegal activities, this proposed work deals with certain links that can be considered for transaction purposes and are checked using a machine learning approach to determine whether it is safe for the same purpose or not.

2 Prior Researches

A malicious URL resembles normal at a simple glance. The biggest threats to present digital world are these malicious links which can cause heavy damage to our digital systems. Such attacks can also occur in our system by opening any file that might contain viruses or any link received through an e-mail. On the other hand, phishing Websites [3] are often cloned Websites of certain famous Websites. Mostly, online purchase Websites are cloned in order to conduct transaction-related attacks. E.g.: Amazon, Flipkart, Ekart, eBay, etc. Users think that they are on safe sites seeing the interface similar to original sites but they overlook the slits through which attackers get their work done. When it comes to user infections, malicious sites can be detected by identifying certain noticeable patterns. Many researchers already developed models to overcome these problems. In [4], Lakshmanarao et al. proposed a machine learning-based method for identifying malicious Websites. A Kaggle dataset with more than 5,000,000 URLs is used for the experiments. Then, they built a phishing Website detection model using four machine learning classifiers—logistic regression, KNN, decision tree, and random forest—and used three techniques for text feature extraction: the count vectorizer, the hashing vectorizer, and the IDF vectorizer. With a hash vectorizer and random forest, the machine learning model was accurate to 97.5%. Using Flask, they also developed a Web application for determining whether a URL entered is malicious or not. The main focus in [5] was on using super learner ensemble to implement a machine learning classifier model for classifying malicious URLs. To support offline and real-time detection, the static feature set is extracted from the URL information with less latency and computational complexity. The proposed multi-class classifier model divides URLs into multiple categories of attacks (phishing, malware, spam, and defacement), while the proposed binary classifier model is used to distinguish between benign and malicious URLs. A dataset of approximately 750,000 URLs is used to test these classifiers. The empirical results demonstrate that the proposed model is effective at detecting malicious URLs. The multi-class classifier has a precision of 96.234% and accuracy of 94.69%, while the binary classifier has a precision of 95.145%. In [6], Manyumwa et al. compared the following ensemble learners' performance: extreme gradient boosting (XGBoost), adaptive gradient boosting (AdaBoost), light gradient boosting (LightGBM), and categorical gradient boosting (CatBoost). The authors looked at how well a few URL features like the Kullback–Leibler Divergence (KL divergence), bag of words segmentation, and additional word-based features were performed. The outcomes demonstrated that the experiments with and without their chosen features performed better. These algorithms were trained on 126,983 URLs from benchmark datasets, and each of the four learners produced results with an overall accuracy greater than 0.95. In [7], Peng et al. demonstrated that adversarial samples are sensitive to the vulnerabilities of the existing DL-based malicious URL detection models. URL adversarial samples are constructed based on perturbations at the character and component levels. Then, these adversarial samples were used to attack conventional DL-based detection models, resulting in decreases in detection accuracies. In the meantime, the perturbations were constrained by the fact that each adversarial sample URL can be easily distinguished from the original URL. In addition, adversarial samples constructed by altering 14 different character types and all other components (with the exception of the scheme component) resulted in the greatest increase in missed blocking of malicious URLs, i.e., a greater decrease in accuracy than other constructed methods. The adversarial samples continued to function and exhibit oblique decreased in accuracy despite the adversarial training being applied to them. The use of a multilayer convolutional neural network (CNN) for malicious URL detection was suggested in [8]. The proposed model first looked at one CNN layer. After that, a two-layer CNN will be utilized to enhance accuracy. The result showed that when the model uses two layers of CNN, the accuracy of detecting malicious Websites increases from 89 to 91%. In [9], Ren et al. proposed an attentional-based BiLSTM model called AB-BiLSTM for detecting malicious URLs. Pre-trained Word2Vec was used to preprocess the URLs and turn them into word vectors. Next, BiLSTM and an attention mechanism were trained to extract and classify the features of URL sequences. The model was tested on a dataset that was collected. The experimental results show that the proposed model can achieve an F1-score of 95.92%, an accuracy rate of 98.06%, a precision rate of 96.05%, and a recall rate of 95.79%.

3 Proposed Methodology

This proposed model is being carried out due to increased cybercrimes in recent days. As the world is shifting toward the digital world since the occurrence of pandemic more and more people have found online platforms reliable for money transaction as they find it easier to transfer money wherever and whenever they want. But handling money on a network is also dangerous as hackers or frauds are present to steal money, passwords, card details, personal information, identity, and much more by unethical means. To prevent such fraudulent attacks, we came with an approach to determine whether a particular URL is safe to make a transaction. A model is trained using machine learning algorithm to recognize the pattern of malicious URL and safe URL and in turn the trained model predicts if any site is safe for transaction or not

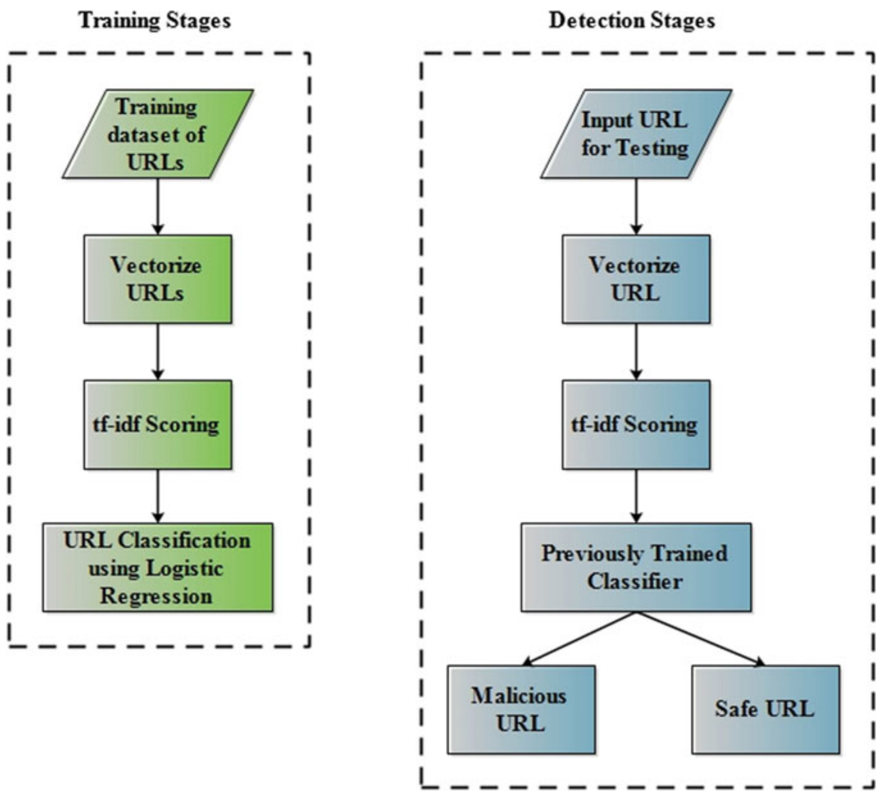


Fig. 1 Block diagram of the proposed model

(Refer to Fig. 1). The data is initially classified into safe and unsafe URLs depending on its encryption and occurrence of secure socket layer certification.

3.1 Identification Learning and Testing

1. We have a dataset of the different URLs where transactions are made, which we use for processing and extract the features.
2. So initially, the dataset of URLs was classified into two parts whether they were safe or unsafe.
3. Furthermore, we used a vectorizer function on the URLs to vectorize the words in URLs in order to find out words that have positive or negative impact. For e.g: words like 'virus', 'dat', 'exe' being in a URL needs more attention for considering whether safe or not.
4. Here, we used TF-IDF scoring. TF-IDF was basically used to find out the importance of specific words whether negative or positive impact in terms of URL

safety. The URLs were broken down into small tokens, and the tokenized words were further vectorized and fed into the model.

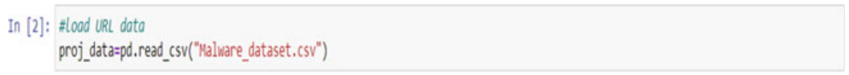
5. Next, the dataset was split for training and testing data in 80–20 ratio, and train_test_split was used.
6. Then, the logistic regression classifier was used to train the model. Logistic regression is one of the most commonly known machine learning algorithms that come under the category of supervised learning technique. A given set of independent variables are used to detect the categorical dependent variable. The result must be a categorical or discrete value in logistic regression. Logistic regression is represented using an equation much similar to linear regression. The independent values (X) are linearly combined with certain valued weights or coefficients for finding dependent variable (Y). A noticeable difference of logistic regression from linear regression is that the output value of the model is a binary value (0 or 1) instead of a numeric value. Below is the equation of logistic regression:

$$Y = \frac{e^{(b_0 + b_1 \times x)}}{(1 + e^{(b_0 + b_1 \times x)})}$$

Finally, the model was tested by using the test dataset's URL and got expected results.

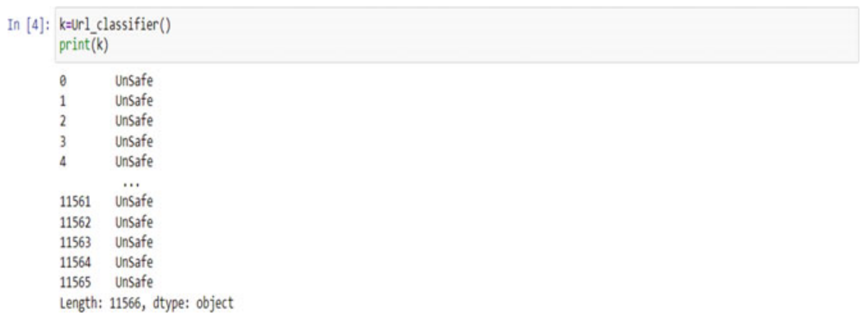
4 Experimental Results and Comparative Analysis

More than 11,500 URLs related to malware Websites were obtained from DNS-BH which is a project that maintain list of malware sites [10]. The data is put into a data frame using Python package pandas.



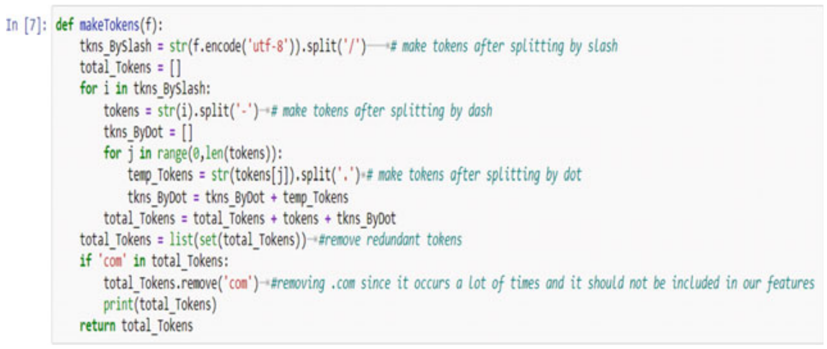
```
In [2]: #load URL data
        proj_data=pd.read_csv("Malware_dataset.csv")
```

The dataset has been further categorized on the basis of their encryption layer. This is being done by the separate function that has been created named Url_classifier.



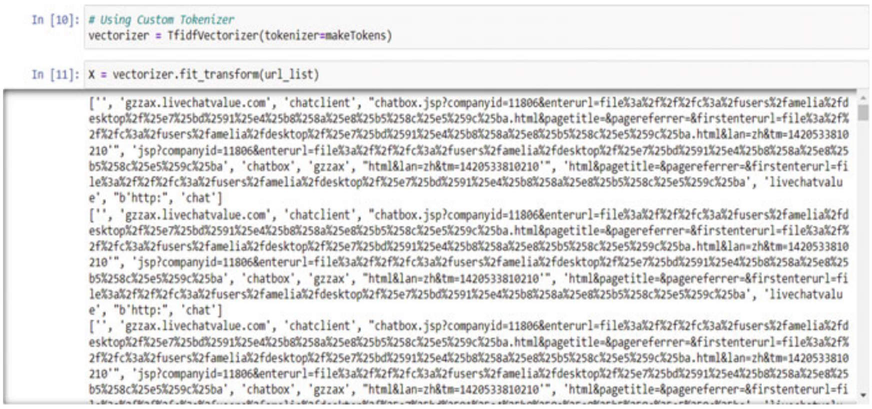
```
In [4]: k=Url_classifier()
        print(k)

0        UnSafe
1        UnSafe
2        UnSafe
3        UnSafe
4        UnSafe
          ...
11561    UnSafe
11562    UnSafe
11563    UnSafe
11564    UnSafe
11565    UnSafe
Length: 11566, dtype: object
```

The URLs are further broken into small tokens to make it suitable to be vectorized using another separate function named make_Tokens to find out each feature that could be depicted through the URL.



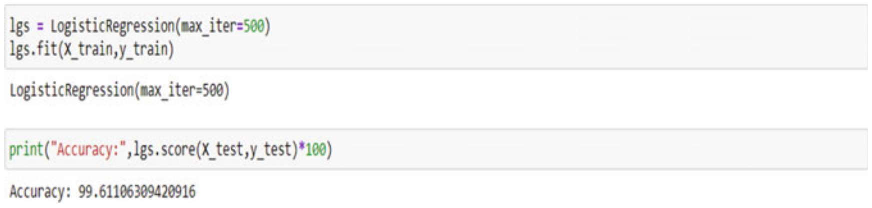
```
In [7]: def makeTokens(f):
            tkns_BySlash = str(f.encode('utf-8')).split('/')    # make tokens after splitting by slash
            total_Tokens = []
            for i in tkns_BySlash:
                tokens = str(i).split('-')  # make tokens after splitting by dash
                tkns_ByDot = []
                for j in range(0,len(tokens)):
                    temp_Tokens = str(tokens[j]).split('.')  # make tokens after splitting by dot
                    tkns_ByDot = tkns_ByDot + temp_Tokens
                total_Tokens = total_Tokens + tokens + tkns_ByDot
            total_Tokens = list(set(total_Tokens))  #remove redundant tokens
            if 'com' in total_Tokens:
                total_Tokens.remove('com')  #removing .com since it occurs a lot of times and it should not be included in our features
                print(total_Tokens)
            return total_Tokens
```

The usage of TF-IDF vectorizer shows the importance given to each word in the URL such as spam words found in the URL shall be marked as unsafe, and the words are further converted to vectors used to fit into the model.



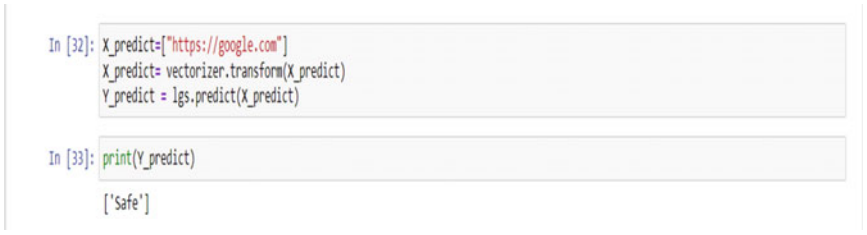
```
In [10]: # Using Custom Tokenizer
         vectorizer = TfidfVectorizer(tokenizer=makeTokens)

In [11]: X = vectorizer.fit_transform(url_list)

['', 'gzzax.livechatvalue.com', 'chatclient', "chatbox.jsp?companyid=11806&enterurl=file%3a%2f%2f%2fc%3a%2fusers%2famelia%2fd
esktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba.html&pagetitle=&pagereferrer=&firstenterurl=file%3a%2f%
2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba.html&lan=zh&tm=1420533810
210'", 'jsp?companyid=11806&enterurl=file%3a%2f%2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25
b5%258c%25e5%259c%25ba', 'chatbox', 'gzzax', "html&lan=zh&tm=1420533810210'", 'html&pagetitle=&pagereferrer=&firstenterurl=fi
le%3a%2f%2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba', 'livechatvalu
e', "b'http:", 'chat']
['', 'gzzax.livechatvalue.com', 'chatclient', "chatbox.jsp?companyid=11806&enterurl=file%3a%2f%2f%2fc%3a%2fusers%2famelia%2fd
esktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba.html&pagetitle=&pagereferrer=&firstenterurl=file%3a%2f%
2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba.html&lan=zh&tm=1420533810
210'", 'jsp?companyid=11806&enterurl=file%3a%2f%2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25
b5%258c%25e5%259c%25ba', 'chatbox', 'gzzax', "html&lan=zh&tm=1420533810210'", 'html&pagetitle=&pagereferrer=&firstenterurl=fi
le%3a%2f%2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba', 'livechatvalu
e', "b'http:", 'chat']
['', 'gzzax.livechatvalue.com', 'chatclient', "chatbox.jsp?companyid=11806&enterurl=file%3a%2f%2f%2fc%3a%2fusers%2famelia%2fd
esktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba.html&pagetitle=&pagereferrer=&firstenterurl=file%3a%2f%
2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25b5%258c%25e5%259c%25ba.html&lan=zh&tm=1420533810
210'", 'jsp?companyid=11806&enterurl=file%3a%2f%2f%2fc%3a%2fusers%2famelia%2fdesktop%2f%25e7%25bd%2591%25e4%25b8%258a%25e8%25
b5%258c%25e5%259c%25ba', 'chatbox', 'gzzax', "html&lan=zh&tm=1420533810210'", 'html&pagetitle=&pagereferrer=&firstenterurl=fi
```

The model further predicts an accuracy of 99%.



```
lgs = LogisticRegression(max_iter=500)
lgs.fit(X_train,y_train)

LogisticRegression(max_iter=500)

print("Accuracy:",lgs.score(X_test,y_test)*100)

Accuracy: 99.61106309420916
```

The model is tested with different URLs, and it gives desired output though it is giving too much accuracy value which might be a sign of overfitting. Also best results are achieved when popular links are being used which can be regarded as a drawback. As for example, we tested it on one of the most popular URLs 'https://google.com/' and got expected results, which can be obviously termed as success.



```
In [32]: X_predict=["https://google.com"]
         X_predict= vectorizer.transform(X_predict)
         Y_predict = lgs.predict(X_predict)

In [33]: print(Y_predict)

         ['Safe']
```

4.1 Comparative Study

This paper works on binary classification of the dataset which means a probabilistic outcome. The dataset has limited data labels and with no noise in it. In such a situation, choosing logistic regression as the classifier is more beneficial as it is easy to be implemented, does not give discrete output and gives output in probabilistic manner which is our goal. On the other hand, though other classifiers give better accuracies but they are complex in structure, and for linearly separable balanced data, linear regression is the best classifier. Our proposed work is compared with other existing works, and its performance can be summed up as shown in the following Table 1. Also from Table 2, it can be shown that the time complexity of our proposed model is less than other conventional models. On the other hand, Table 3 and Fig. 2 show that our proposed model gives higher accuracy than other traditional models.

Table 1 Comparison with other existing works

Proposed work	Previous works
Our model can derive the significance of features quite fast	Other works could not derive the significance of features or even if could was quite slow
In our model, the algorithm has convex loss function, so it will not hang in local minima	In other works, the algorithms used complex non-linear mathematical functions at different times and most often the loss function is non-convex, thus it is quite possible to get stuck in local minima
Our model has an extra feature of checking the secured layer of the links and hence can be better used for checking transaction URLs	The previous works did not have any feature of checking the encryption
Our model can derive confidence level (about its prediction)	Other models could only output the labels
Our model is based on statistical approaches (that is dealing probabilistic data here)	Other models were based on geometrical properties of the data more and less focus on probabilistic data

Table 2 Comparison based on time complexity

Comparison perspective	Time taken by proposed model	Time taken by other existing models		
	Using logistic regression	Using decision tree	Using SVM	Using K-nearest neighbors
Train time complexity	**O(n*m)**	O(n*log(n)*m)	O(n^2)	O(k*n*m)
Test time complexity	**O(m)**	O(m)	O(n'*m)	O(n*m)

Table 3 Comparison with respect to accuracy

Models	Accuracy (%)
Ref. [4]	97.5
Ref. [5]	94.69
Ref. [6]	95
Ref. [8]	91
Ref. [9]	98.06
Proposed model	99

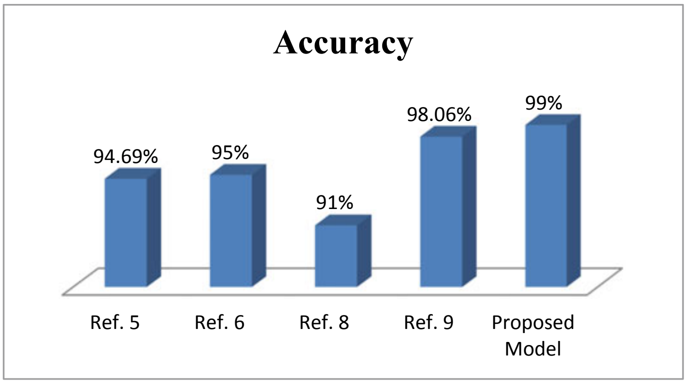


Fig. 2 Graphical representation of comparative analysis with respect to accuracy

5 Conclusion and Future Scope

We can conclude that with the help of certain methods and processes of hacking which is known to be social attack, a hacker can steal money, obtain card details, and much more. This can either be private or can also be public data, where they can make alterations using user's profile for their selfish needs. By getting into the account, the user might have unknowingly given access with just a single click on the respective malicious Website. Therefore, to make a safer transaction and protect data, we have implemented a machine learning algorithm to detect and help users in tracking malicious Websites before performing any transaction on that particular site.

In future, our target will be to implement these algorithms such that they can categorize the URLs more precisely. We shall try to work on the URLs so that they predict the output in terms of rating and percentage of the safeness of the URL. We shall also implement it's working in the browser in which user can copy a transaction URL and paste to check whether it is safe. The machine learning algorithm shall automatically help the user by showing a message whether to continue if the site is malicious or else proceed if it is safe.

References

1. Bendovschi A (2015) Cyber-attacks—trends, patterns and security countermeasures. Procedia Econ Finan 28:24–31
2. Digital commerce transactions. Report https://www.moneycontrol.com/news/business/banks/digital-commerce-transactions-increased-30-in-2021-but-so-are-digital-frauds-says-rbi-report-8256031.html
3. Karabatak M, Mustafa T (2018) Performance comparison of classifiers on reduced phishing website dataset. In: 2018 6th international symposium on digital forensic and security (ISDFS), pp 1–5
4. Lakshmanarao A, Babu MR, Bala Krishna MM (2021) Malicious URL detection using NLP, machine learning and FLASK. In: 2021 international conference on innovative computing, intelligent communication and smart electrical systems (ICSES), pp 1–4
5. Hevapathige A, Rathnayake K (2022) Super learner for malicious URL detection. In: 2022 2nd international conference on advanced research in computing (ICARC), pp 114–119
6. Manyumwa T, Chapita PF, Wu H, Ji S (2020) Towards fighting cybercrime: malicious URL attack type detection using multiclass classification. In: 2020 IEEE international conference on big data (Big Data), pp 1813–1822
7. Peng Z, He Y, Sun Z, Ni J, Niu B, Deng X (2022) Crafting text adversarial examples to attack the deep-learning-based malicious URL detection. In: ICC 2022—IEEE international conference on communications, pp 3118–3123
8. Singh A, Roy PK (2021) Malicious URL detection using multilayer CNN. In: 2021 international conference on innovation and intelligence for informatics, computing, and technologies (3ICT), pp 340–345
9. Ren F, Jiang Z, Liu J (2019) A bi-directional LSTM model with attention for malicious URL detection. In: 2019 IEEE 4th advanced information technology, electronic and automation control conference (IAEAC), pp 300–305
10. Malware dataset. https://www.unb.ca/cic/datasets/url-2016.html