ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse330952020.pdf
https://doi.org/10.30534/ijatcse/2020/330952020

Evaluating the performance of some machine learning algorithms for malicious URL detection problem

Lai Van Duong[1], Tisenko Victor Nikolaevich[2],Le Duong Anh Tuan[3], Nguyen Tung Lam[4], Nguyen Quang Dam[5], Pham Thi Thuong[6]
[1,3,4,5,6]Information Assurance dept. FPT University, Hanoi, Vietnam, duonglvse05009@fpt.edu.vn , TuanLDAHE130230@fpt.edu.vn, LamNTHE130587@fpt.edu.vn, damnqse05820@fpt.edu.vn, ThuongPTSE05856@fpt.edu.vn
[2]Department Quality Systems, Peter the Great St. Petersburg Polytechnic University, Russia, St.Petersburg, Polytechnicheskaya, 29, v_tisenko@mail.ru

ABSTRACT

The World Wide Web has become an important part of daily life to impart information and provide knowledge. It helps to exchange information promptly, quickly, and easily. With the growth of the World Wide Web, malicious URLs quickly become a common and serious threat to cyber-security. Malicious URLs store unwanted content (including spam, phishing, etc.) and attract unwanted users to fall victim of phishing (users lose money, their information is stolen and malware is installed without users' knowledge) and cause a billion dollars of damage each year. The task of timely detecting and resolving such threats is crucial. This paper proposes a malicious URL detection model based on machine learning that makes it capable of detecting new or unknown malicious URLs.

Key words: machine learning, malicious URLs, TF- IDF.

1. INTRODUCTION

URL (Uniform Resource Locator) is used to refer to resources on the Internet. URLs bring hyperlinks to web pages. Different resources are referred to by the address that called the URL and also known as a web address. URLs are most used to reference web pages (HTTP). Besides, it is also used for FTP, email, JDBC, and many other applications. Each HTTP URL follows the syntax standard of a URI.

Malicious URL is understood as the links that adversely affect the user. The URLs redirect to the resources or pages on which these pages can execute code on the user's computer, direct users to other malicious or phishing web sites or to download malware without users' knowledge. Malicious URLs are increasingly diverse to serve different purposes. The most common types of malicious URLs include Malicious website URLs such as online gambling and porn sites; The website automatically downloads the malware: When users access, the pop-ups will automatically download the malware containing viruses or automatically install plug-ins for the browser to collect users' information; Insert and execute malicious JavaScript (XSS): the web pages infected with malicious JavaScript will infect by downloading the .js file which the browser will execute; Phishing website URL: This form scam based on social engineering to make the user willing to give information to the online criminal; etc. Because the malicious URL is becoming more diverse and difficult to detect, the task of early detecting and warning of malicious URL is essential. Currently, there are two popular approaches of detecting malicious URLs: blacklisting and machine learning. The blacklist approach is a popular and classic technique for detecting malicious URLs, often maintaining a list of known malicious URLs. Since new URLs can easily be created on a daily basis, blacklisting cannot detect new threats. The blacklist approach is simple, easy to build and use, and can contribute to a shared database. However, it is difficult to detect new malicious URLs with this approach. In particular, after a long period of time, the database can become cumbersome with old data that are no longer valuable. Filtering based on machine learning is a new approach. This approach uses machine learning algorithms to analyze large amounts of pre-categorized URLs data so it is possible to discover new URLs that have never appeared. This method is highly flexible, capable of detecting new malicious URLs that are generated by algorithms. In particular, it just needs the initial training data without storing this data for long periods. This helps to reduce storage costs. However, this approach also has the disadvantages that are requiring large amounts of pre-classified data, high collection costs, and long-term testing and evaluation to increase accuracy.

In this paper, we propose a method of detecting malicious URLs based on machine learning. Our paper is presented as follows: In section 3, we present the malicious URL detection model and the method to extract the features of malicious URLs. In section 4, we conduct experiments, compare and evaluate the results of detecting malicious URLs.

2. RELATED WORKS

There are many approaches to solving the malicious URL detection problem. We classify them into: (i) Blacklisting or Heuristics, and (ii) Machine learning approaches.

This document [1] presented the basic Blacklisting technique. Blacklisting usually maintains a list of known malicious URLs. Whenever a new URL is accessed, the database lookup is performed. If the URL is in the Blacklist, it is considered malicious and then an alert will be generated; otherwise, it is considered benign. The heuristic approach is an extension of Blacklist methods. The idea of this approach is to create a "Blacklist of signatures" of the malicious activity. This approach often analyzes the execution dynamics of the webpage [2,3,4]. Here, the idea is to look for signs of malicious activity like irregular process creation, recurring redirects, etc. These methods necessarily access the website and therefore the URLs can do an attack. Therefore, such techniques are usually performed in a virtual machine environment. Its disadvantages are resource consumption and code execution requirements. For the machine learning approach, currently there are 2 main directions: Static analysis and Dynamic analysis. In static analysis, perform website analysis based on available information without executing the URL. There are many studies on extracting malicious URL features. Birhanu Eshete et al. [5] proposed using three groups of features including URL features, Page-Source features (HTML and JavaScript), and Social-Reputation features. Justin Ma et al. [6] used statistic methods to extract the lexical and host-based features of malicious URLs. In addition, the documents [7 , 8] proposed collecting lexical and host-based features in real-time. Since there is no code execution required, these methods are more secure than the Dynamic approaches. Dynamic analysis techniques monitor the behavior of potential victim systems in order to look for any anomalies. These include monitoring system call sequences for detecting abnormal behavior [9] and mining internet access log data for seeking suspicious activity [10]. The dynamic analysis technique is difficult to implement and generalize. In this paper, we propose to use static analysis with machine learning to detect malicious URLs.

3. The method of detecting malicious URL based on machine learning

3.1. The malicious URL detection model

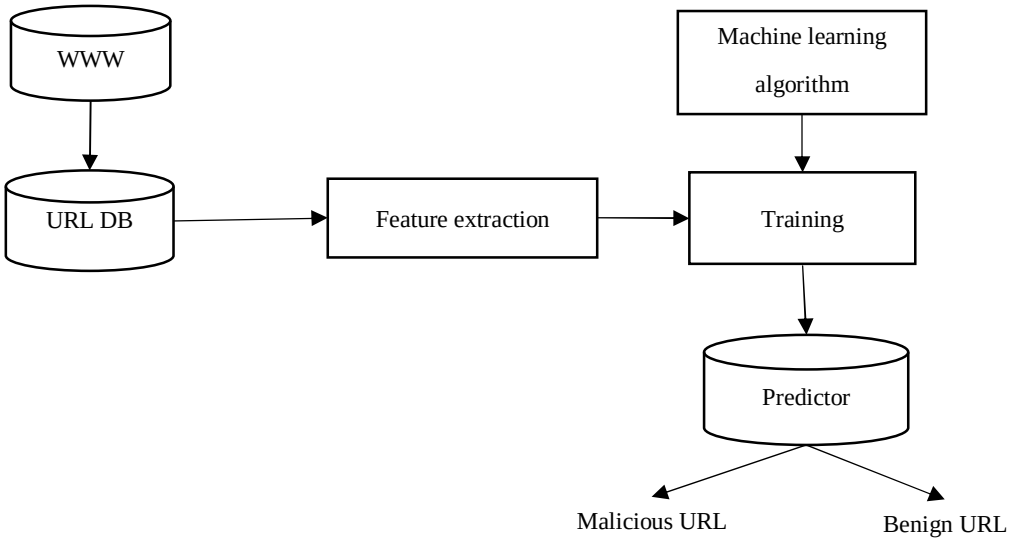

Fig. 1. The malicious URL detection model

The model for detecting malicious URLs is based on 4 basic steps:

- Crawling data: This module collects the URL data from different sources.
- Feature extraction: Features are extracted from URLs and then represented as vectors. With each URL, it is labeled malicious or non-malicious.
- Training: After obtaining the data that has been labeled and represented in the vector format, the data

will be run through a machine learning algorithm to generate the model.

- Detecting malicious URLs: Model obtained from the training process is used to predict each time new URLs appear.

3.2. The feature extraction and representation

3.2.1. *Extracting feature with N-gram*

N-gram is the technique of splitting a sequence into sub-sequences by evenly dividing an existing sequence into sub-sequences with the same length N. Basically, N is usually from 1 to 3 with the corresponding names unigram (N = 1), bigram (N = 2), trigram (N = 3). The output of the N-gram process is the list of terms (features) of the URL. These features can be vectorized for later computation. The following is an illustrative example of N-gram. Assume the following 3 URLs:

https://www.google.com.vn/?search=facebook.com
https://www.facebook.com/groups/
http://dantri.vn

After vectorization according to unigram, we obtained the following results:

	com	dantri	facebook	google	groups	http	https	search	vn	www
https://www.google.com.vn/?search=facebook.com	2	0	1	1	0	0	1	1	1	1
https://www.facebook.com/groups/	1	0	1	0	1	0	1	0	0	1
http://dantri.vn	0	1	0	0	0	1	0	0	1	0

Figure 2. Representing feature with the number of occurrences of terms

3.2.2. *Extracting feature with TF-IDF*

TF (Term Frequency) is used to estimate the frequency of terms appearing in the document. The TF formula is as follows:

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d) : w \in d\}}$$

Where: TF value in the range [0, 1]; f(*t, d*) is the number of times that term *t* occurs in document *d*; max({*f(w, d): w* ∈ *d*}) is the frequency of the most occurring term in document d.

IDF (Inverse Document Frequency): is the inverse frequency of a term in the corpus. Calculate IDF to decrease the value of common terms. Each term has only 1 unique IDF value in the corpus.

$$idf(t,D) = \log\frac{|D|}{|\{d \in D : t \in d\}|}$$

Where: |*D*| is the total number of documents in the corpus; |{*d* ∈ *D: t* ∈ *d*}| is the number of documents where the term *t* appears.

TF-IDF is calculated by the formula:

$$tf\text{-}idf(t,d,D) = tf(t,d) \times idf(t,D)$$

Terms with high TF-IDF value appear much in this document, and appear less in other documents. This helps to filter out common terms and retain high-value terms (keywords of that document) [12]. Figure 3 below shows the results after applying TF-IDF to the three above URLs.

	com	dantri	facebook	google	groups	http	https	search	vn	www
https://www.google.com.vn/?search=facebook.com	0.590852	0.000000	0.295426	0.38845	0.000000	0.000000	0.295426	0.38845	0.295426	0.295426
https://www.facebook.com/groups/	0.417790	0.000000	0.417790	0.00000	0.549301	0.000000	0.417790	0.00000	0.000000	0.417790
http://dantri.vn	0.000000	0.622766	0.000000	0.00000	0.000000	0.622766	0.000000	0.00000	0.473630	0.000000

Figure 3. Results of feature extraction with TF-IDF

4. Experiments and evaluations

4.1. Measures

To evaluate the performance of detecting malicious URLs, we use 5 different measures: accuracy, precision, recall, f1-score, and confusion matrix. These metrics are calculated based on the following components:

- True positive (TP) is the number of malicious URLs correctly classified.
- True negative (TN) is the number of benign URLs correctly classified.
- False positive (FP) is the number of benign URLs missed classified into malicious.
- False negative (FN) is the number of malicious URLs missed classified into benign.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$Precision = \frac{TP}{TP + FP} \times 100\%$$

$$Recall = \frac{TP}{TP + FN} \times 100\%$$

$$F1 - \text{score} = \frac{2 \times precision \times \text{Re } call}{precision + \text{Re } call}$$

Confusion matrix contains the following four components for each class of classification:

Table 1. Confusion matrix

		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

4.2. Experimental data and scenario

For training, we collected 370,311 benign URLs and 226,895 malicious URLs taken from the sources shown in Table 2 below.

Table 2. Sources of URL crawling

No.	Source	Description	Web address
1	Phishtank	Phishtank is a Web service that shares phishing URLs. Suspected URLs are submitted and re-verified. This data is updated every hour.	[11]
2	URLhaus	URLhaus is a project from abuse.ch with the goal of sharing malicious URLs that are being used to distribute malware.	[12]
3	Alexa	The database that ranks the Web pages	[13]
4	Malicious_n_Non-Malicious URL	Database with over 400,000 labeled URLs { **good (82%), bad (18%)}.**	[14]

From the above URL datasets, we extract features and label 'bad' for the malicious URL and label 'good' for the benign URL. Aggregating the feature and label to form the training dataset. This dataset is randomly divided at a ratio of 90% for training and 10% for validation. After training the model with the above dataset, we will test the model with 30 new URLs (50:50 ratio) taken from the internet. These URLs are listed in Tables 3 and 4 below.

Table 3. List of malicious URLs for testing

No.	URL
1	stcroixlofts.com/inc/manager/config/auth/log/a422d72c6770a3b73ba0a058bad45eccZjdmODY4ODA5MTZiOGMxMzRmN2I4ZWRjNTFmMDljYWU=/resolution/websc_login/?country.x=&locale.x=en_
2	gotrucktrans.com/security/Sign%20in%20to%20your%20Microsoft%20account%20pass.php
3	stcroixlofts.com/inc/manager/config/auth/log/09df04f20f978f53f685d649757d4f9cYWZiYmRhOWVmZTY2YmFkMmJkZjhlNjA2MmMxYzI5Y2Q=/resolution/websc_login/
4	www.nttdocomo-smt-protects.com/
5	dacele34.beget.tech/dob/
6	www.silexminerals.in/seprue/9xba/source/?email=user@cslanet.calstatela.edu
7	marikazillasu.myscriptcase.com/driper/sax/Tan/wolf/jazzz/winwin/file/4d747a3d6dc68ce1315559b8c8f67368/
8	www.tangoargentino.it/dnir.rps/sc.php
9	banckpichincha1.webcindario.com/RecuperarPwd.aspx.html
10	slc-cr.com/myphp/GMA-VALIDATE.php
11	nttdocomo-smt-protects.com/
12	pegocontabilidade.com.br/Fazendo-melhor-INTER/acessar.php
13	datoscede.uniandes.edu.co/datoscede/system/ckeditor/servicionovo/acessar.php
14	www.englishcenteridiomas.com/lib/modal/banco.inter/sincronia.de.dados/conta-digital/2018/acesso/images.jpg/acessar.php
15	fieldorf.pl/multimedia/o-melhor-para-voce/id_digital/sincronia.php

Table 4. List of benign URLs for testing

No.	URL
1	colab.research.google.com.vn/
2	ereka.vn/topic/tri-tue-nhan-tao-97249675912513823
3	github.com/
4	vtvgo.vn/xem-truc-tuyen-kenh-vtv3-3.html
5	stackoverflow.com/questions/8486294/how-to-add-an-extra-column-to-a-numpy-array
6	dantri.com.vn/
7	dantri.com.vn/su-kien/ngo-ngang-lac-vao-benh-vien-cong-hien-dai-nhu-khach-san-o-viet-nam-20181217152549298.htm
8	thanhnien.vn/gioi-tre/sang-kien-tao-nen-chuoi-gia-tri-cho-su-phat-trien-ben-vung-cua-doanh-nghiep-1033119.html
9	www.24h.com.vn/tin-tuc-trong-ngay/cau-em-ut-doan-van-hau-tung-mac-toi-tay-dinh-vuon-minh-thanh-nha-vo-dich-c46a1014077.html
10	www.bbc.com/news/uk-politics-46586673
11	news.zing.vn/tiec-sinh-nhat-cua-trum-xa-hoi-den-quy-tu-dan-sao-lon-hong-kong-post901282.html
12	news.zing.vn/bao-tay-ban-nha-chuc-mung-tuyen-viet-nam-vo-dich-aff-cup-post901267.html
13	news.zing.vn/mu-gap-psg-liverpool-dung-bayern-o-vong-18-champions-league-post901232.html
14	baomoi.com/thu-tuong-chi-dao-ve-2-nghi-quyet-dau-nam-moi-2019/r/29025286.epi
15	baomoi.com/thua-thien-hue-nhieu-diem-sang-trong-nganh-chan-nuoi-nam-2018/r/29025662.epi

4.3. Experimental results

4.3.1. *Training results*

Using training dataset at the ratio of 90% for training and 10% for validation, in turn train the model with Multinomial Naive Bayes, Bernoulli Naive Bayes, Decision Trees, and SVM algorithms. The training results of each algorithm are presented in the following tables:

Table 5. Confusion matrix with Multinomial Naive Bayes algorithm

		Actual	
		Malicious	Benign
Predicted	Malicious	19294	1613
	Benign	3129	35685

Table 6. Confusion matrix with Bernoulli Naive Bayes algorithm

		Actual	
		Malicious	Benign
Predicted	Malicious	19964	2159
	Benign	2459	36139

Table 7. Confusion matrix with Decision Tree algorithm

		Actual	
		Malicious	Benign
Predicted	Malicious	21315	1253
	Benign	1108	36045

Table 8. Confusion matrix with SVM algorithm

		Actual	
		Malicious	Benign
Predicted	Malicious	21707	681
	Benign	716	36617

Table 9. Experimental results with 4 measures: accuracy, precision, recall, and F1-score

	Accuracy	Precision	Recall	F1-score
Multinomial Naive Bayes	92.060	92.285	86.046	89.056
Bernoulli Naive Bayes	92.267	90.241	89.034	89.633
Decision Tree	96.047	94.448	95.059	94.753
SVM	**97.661**	**96.954**	**96.811**	**96.882**

Through the experimental results in the above tables, we notice that in 4 algorithms, the SVM algorithm gave the highest accuracy. Measures when using the SVM algorithm are all high, namely Accuracy as 97.661%, Precision as 96.954%, Recall as 96.811%, and F1-score as 96.882%. The algorithm that gave the second-highest result is the Decision Tree algorithm with the accuracy as 96.047%. Meanwhile, both the Multinomial Naive Bayes and the Bernoulli Naive Bayes algorithms gave significantly lower results with the accuracy of 92.060 % and 92.267 % respectively.

4.3.2. *Testing results*

Table 10. Testing results

Algorithm	Accuracy	Precision	Recall	F1-score
Multinomial Naive Bayes	90	100	80	88.889
Bernoulli Naive Bayes	90	92.857	86.667	89.655
Decision Tree	93.333	93.333	93.333	93.333
SVM	100	100	100	100

Comment: Through Table 10, we notice that the testing results with the new URLs are similar to the training results. SVM is still the most accurate algorithm with accuracy as 100%. Meanwhile, Multinomial Naive Bayes and Bernoulli Naive Bayes algorithms still gave significantly lower results. The experimental results proved that our proposed method is capable of detecting new URLs well.

5 CONCLUSION AND FUTURE DIRECTION

Currently, malicious URLs are growing rapidly and increasingly diverse. They have become a common and serious threat to cyber-security. The task of early detecting and warning malicious URLs is very important. In this paper, based on the feature extraction and the machine learning algorithms, we have successfully built models of detection malicious URLs with high accuracy. In particular, the SVM algorithm gave the highest results on all measures. In the future, we will continue research to improve the model on larger datasets or use other machine learning or deep learning algorithms to increase the accuracy.

REFERENCES

[1] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and ChengshanZhang..**An empirical analysis of phishing blacklists**. *In Proceedings of Sixth Conference on Email and Anti-Spam (CEAS)*.2009. pp. 1- 10.

[2] Byung-Ik Kim, Chae-Tae Im, and Hyun-Chul Jung. **Suspicious malicious web site detection with strength analysis of a javascript obfuscation**. *International Journal of Advanced Science and Technology*. Vol. 26, 2011, pp. 19-32.

[3] Christoph Kolbitsch, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Rozzle: **De-cloaking internet malware**. *IEEE Symposium on Security and Privacy*.20-23 May 2012. pp. 443-457

[4] Mahmoud T Qassrawi and Hongli Zhang. **Detecting malicious web servers with honeyclients**. *Journal of Networks*.Vol 6, No 1. 2011, pp.145-152.

[5] Birhanu Eshete, Adolfo Villafiorita, and KomministWeldemariam. **Binspect: Holistic analysis and detection of malicious web pages**. *International Conference on Security and Privacy in Communication Systems*.2013, pp 149-166.

[6] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. **Beyond blacklists: learning to detect malicious web sites from suspicious URLs**. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 1245-1254.

[7] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. **Identifying suspicious URLs: an application of large-scale online learning**. *In Proceedings of the 26th Annual International Conference on Machine Learning*. June 2009 Pages 681–688

[8] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. **Learning to detect malicious urls**. *ACM Transactions on Intelligent Systems and Technology* (TIST).May 2011 Article No.: 30.

[9] Gerardo Canfora, Eric Medvet, Francesco Mercaldo, and Corrado Aaron Visaggio. **Detection of Malicious Web Pages Using System Calls Sequences**. *In Availability, Reliability, and Security in Information Systems*. pp 226-238.

[10] Yu Tao. *Suspicious URL and Device Detection by Log Mining*. Ph.D. Dissertation. Applied Sciences: School of Computing Science.

[11] Join the fight against phishing. https://www.phishtank.com/ developer_info.php

[12] URLhaus Database Dump. https://urlhaus.abuse.ch/%20downloads/csv/

[13] Majestic Million CSV. http://downloads.majestic.com/%20majestic_million.csv.

[14] Datasets. https://www.kaggle.com/antonyj453/%20urldataset#data.csv