Developing a Framework for Detecting Phishing URLs Using Machine Learning

Nguyen Tung Lam [1†],
LamNTHE130587@fpt.edu.vn
Information Assurance dept. FPT University, Hanoi, Vietnam

Summary
The attack technique targeting end-users through phishing URLs is very dangerous nowadays. With this technique, attackers could steal user data or take control of the system, etc. Therefore, early detecting phishing URLs is essential. In this paper, we propose a method to detect phishing URLs based on supervised learning algorithms and abnormal behaviors from URLs. Finally, based on the research results, we build a framework for detecting phishing URLs through end-users. The novelty and advantage of our proposed method are that abnormal behaviors are extracted based on URLs which are monitored and collected directly from attack campaigns instead of using inefficient old datasets.

Keywords:
phishing URLs; detecting phishing URLs; abnormal behaviors of phishing URLs; Machine learning;

1. Introduction

Regarding the problem of detecting phishing URLs, there are two main methods: rule-based detection method and behavior analysis-based detection method [1, 2, 3]. In which, the behavior-based detection method using machine learning and deep learning algorithms is developing a lot nowadays. Behavior-based detection methods often seek ways to analyze anomalous behavior based on URL information. Some information is often applied to extract such as [1, 3, 4, 5, 6, 7, 8, 9, 10] domain, HTTP protocol, body, DNS query, lexical, host-based, etc. In this paper, we propose some new features representing abnormal behaviors of phishing URLs. Specifically, we choose to use the following feature groups: domain, body, HTML, etc. Each feature group has some new features that are first proposed in this study. These new features are first extracted from phishing URLs, so they will bring high efficiency. After extracting these abnormal behaviors, some machine learning or deep learning algorithms are applied to classify phishing URLs and clean URLs. In reality, deep learning techniques are being applied in many different fields. However, the task of detecting phishing URLs requires fast response times and deep learning algorithms are ofter slower than machine learning algorithms. Therefore, in this paper, we propose to use a supervised machine learning algorithm for this classification task.

2. Related Works

In the study [2], the authors presented in detail the different components in URLs and some approaches for detecting phishing URLs using machine learning. Specifically, in their research, the authors [2] proposed a method to analyze URLs into different components including Lexical Features, Host-based Features, Content-based Features, Other Features. Then based on these components, the authors proposed to use some machine learning methods (namely, Batch Learning, Online Learning, Representation Learning, etc.) to classify these URLs. Similarly, in the research [3], the authors proposed a method to detect malicious URLs using the dynamic convolutional neural network model. In the experimental part, the authors conducted experiments and proved that this proposal is superior to other methods using convolutional neural networks. In addition, the study [4] also proposed a detection method based on Associative Classification. Specifically, in this study, the author used some abnormal behavior of URLs including URL Features, Webpage Content Features. Experimental results proved that this method is superior to the rule-based method. Besides, studies [5, 6, 7, 8, 9] presented some approaches using deep learning for phishing URL detection. The research [10] listed several tools and methods of detecting phishing URLs using machine learning.

3. Detection Method

3.1 Features of Phishing URL

Table 1 below lists features that represent abnormal behaviors of phishing URLs. These behaviors are directly extracted from some components of the URLs.

Manuscript received October 5, 2023
Manuscript revised October 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.10.19

Table 1: Some abnormal behaviors

Group		***Feature Name***	***Description***
Domain		DashCharacter	The existence of dash character (-) in domain.
		SensitiveWords	Sensitive words are commonly used in the phishing domain such as "chuyển tiền", "vay tín dụng", "trúng thưởng", etc.
		PercentNumericChars	Percentage of numeric characters in domain.
		TrustTLD	Some credible TLD, example .com, .co, .org, .us, .net, .blog, .io, .biz.
URL		ShorteningService	URL shortening is used to create shorter aliases for long URLs. Some URL shortening services are bit.ly, ow.ly, short.io.
		RedirectInURL	URL contains "//" character to redirect to another website.
		HTTPS	Check if HTTPS exists in the URL.
HTML	Head	Favicon	Check if favicon is installed from a hostname different from the URL hostname of the website.
		MissingTitle	Check if the title tag is empty in HTML source codes.
	Body	ServerFormHandler	Check the action of form tag.
		Iframe	Check if iframe is used in HTML source codes.
		PopupWindow	Check if HTML source code contains a popup command to start a popup window.
		AnchorHref	Correlated percentage of external *href* in anchor tags.
		ImageSrc	Correlated percentage of external *src* in image tags.
		ScriptAndLink	Correlated percentage of total external *src* in script tags and *href* in link tags.
		FrequentDomainNameMismatch	Correlated percentage of external URL.
	Footer	ContactMail	Check if "mailto" is in the HTML source code.
		VerifyByMoitVn	Check if the website has been verified by Vietnam MOIT.
		VerifyByTnm	Check if the website has been verified by Tinnhiemmang – NCSC Vietnam.

3.2 Detection Method

Based on the abnormal behaviors shown in Table 1, machine learning algorithms are used to classify URLs. To detect phishing URLs, we use Random Forest (RF) and SVM machine learning algorithms. The documents [11, 12] presented the mathematical basis and operating principle of these algorithms in detail.

4. Experiments and evaluation

4.1 Experimental dataset

The dataset used for this paper contains two label types: phishing and legitimate websites. In which, the phishing websites were collected from some sources:

- NCSC Phishing Database: This is a credible source to get active phishing URLs in Vietnam.
- Chongluadao: The blacklist of ChongLuadDao is a useful phishing URL source in Vietnam, which is verified [13].
- Openphish.com: OpenPhish is a service website dedicated to sharing phishing URLs. Suspicious URLs could be sent to OpenPhish for verification [14].
- Phishing.army: This website provides a phishing domain list which is generated every 6 hours from reports: PhishTank, OpenPhish, Cert.pl, PhishFindR, Urlscan.io, and Phishunt.io. Each domain is analyzed to remove false positives through the Whitelist of Anudeep and the Alexa Rank [15].

The legitimate websites were taken from:

- Online.gov.vn: This is a website of MOIT Vietnam. It contains addresses of e-commerce websites selling goods, websites providing e-commerce services in Vietnam [16].
- Tinnhiemmang.vn: This website provides safety domains that are verified by NCSC Vietnam Manual. It collects website addresses of airlines, banks in Vietnam: ticket booking websites of airlines, websites used in e-banking services [17].

Besides, we used several datasets on the network including [18, 19, 20 21].

The dataset of malicious/clean URLs collected from the crawling process described above is divided into 2 parts: 10,000 URLs (5,000 malicious URLs and 5,000 clean URLs), and 470,000 URLs (400,000 malicious URLs and 70,000 clean URLs). 80% of the data is used for training and 20% of the data is used for testing.

4.2 Evaluation criteria

- Accuracy: is defined as the percentage of the correct prediction in the test data.

$$acc = \frac{TP+TN}{TP+TN+FP+FN} * 100\%$$

- Confusion Matrix: is a table that is often used to describe the performance of a classification model.
- Precision: is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$recision = \frac{TP}{TP+FP} * 100\%$$

- Recall: is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

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

- F1-score: is the harmonic mean of precision and recall. High F1 value means the classifier is good.

$$F1 = \frac{2 * precision * Recall}{precision + Recall}$$

4.3 Experimental results

The dataset built in the previous step is divided into two subsets. 80% of the dataset is used for training, and the rest is used for testing. The experiment is repeated many times to choose the best model with each algorithm. In the first phase, we build models in the dataset with 21 features. The two algorithms gave the following results, respectively:

Table 2: Experimental results when using RF algorithms

n_estimators	*Test Accuracy*	*Train Accuracy*	*Precision*	*Recall*	*F1-score*
10	0.8700	0.9420	0.8387	0.9176	0.864
20	0.8462	0.9570	0.8020	0.9167	0.8556
30	0.8698	0.9539	0.8604	0.8810	0.8706
40	0.8639	0.9480	0.8222	0.9136	0.8654
50	0.8580	0.9525	0.8333	0.875	0.8537
60	0.8580	0.9480	0.8519	0.8519	0.8519
70	0.8462	0.9510	0.8587	0.8587	0.8587
80	0.8640	0.9524	0.8488	0.8795	0.8639
90	0.8698	0.9480	0.8333	0.9444	0.8854
100	0.8580	0.9540	0.8276	0.8889	0.8571
200	0.8876	0.9494	0.8817	0.9111	0.8961
300	0.8520	0.9495	0.8404	0.8876	0.8634
400	0.8698	0.9495	0.8621	0.8823	0.8720
500	0.8639	0.9480	0.8132	0.925	0.8655
600	0.8521	0.9495	0.8205	0.8533	0.8366
700	0.8757	0.951	0.8333	0.9259	0.9259
800	0.8876	0.9495	0.8763	0.9239	0.9239
900	0.8757	0.9494	0.87951	0.8690	0.8690
1000	**0.8934**	**0.9435**	**0.8333**	**0.9615**	**0.8928**

Table 2 is some experimental results when using RF algorithm. Obviously, when changing the number of trees in the algorithm, the accuracy of the algorithm changed. Comparing all models in the above table, we noticed that the model with n_estimators=1000 gave the best result with the highest Test Accuracy of 0.8934 and the highest Recall of 0.9615.

Table 3: Experimental results when using SVM algorithms

n_estimators	*Test Accuracy*	*Train Accuracy*	*Precision*	*Recall*	*F1-score*
linear	0.8402	0.8157	0.7652	1	0.8670
poly	0.8166	0.8663	0.7327	0.9487	0.8268
rbf	0.8225	0.8678	0.8065	0.8621	0.8333
sigmoid	0.7752	0.74	0.7802	0.7978	0.7889

Comparing all models in the above table, we found that the model with kernel='linear' gave the best result with the highest Test Accuracy of 0.8402 and the highest Recall of 1.

After comparing models build by the two algorithms above, we decided to optimize the RF model with n_estimators=1000 in this paper.

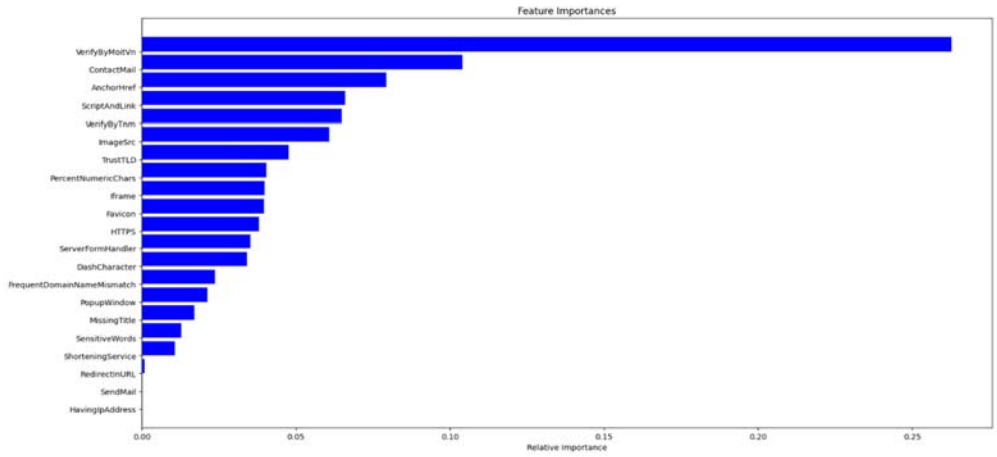

Fig. 1 Feature importance assessment results

Look at the visualization of the feature importance, the two features SendMail and HavingIpAddress have zero weight. Therefore, we removed these features and rebuilt the model. The model finally chosen has the result as good as before removing features. Finally, the chosen model is built by using the RF algorithm with parameters:

- Number of features: 19
- n_estimators: 1000 (1000 trees)
- Test accuracy: 0.8934

5. Building frameworks

5.1 Architecture of malicious URL detection application

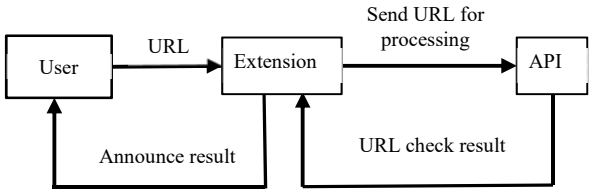

Fig. 2 Architecture of phishing URL detection application

From Figure 2, it can be seen that there are 2 main components in the phishing URL detection application including Extension and API. In which, Extension has the function of getting URL information from the user and then sending that URL to the API. The API has the function of checking this URL and responding to the Extension with the URL check results so that the Extension notifies the user. If the URL is clean, the app will allow the user to continue to access it. Conversely, if the URL is malicious, the app will proceed to block and display warnings to the user. In order to quickly and accurately implement the ability to detect URLs, in this paper, the authors build and install the Extension on the user's web browser. Besides, the malicious URL detection API is installed and built on the server.

5.2 Developing Extension in phishing URL detection application

5.2.1 Building Extensions on the Web browser

a) Building Extensions on the Chrome browser

The main file component of the Extension includes:

- manifest.json: is the configuration file for the Extension.
- popup.html: is the Interface of the Extension.
- blocked.html: is the Result display interface.
- background.js: performs access control, handles the main logic of sending and receiving information from the server, and then outputs the results to the user.

b) Building Extensions on the Firefox browser

The Extension on Chrome is fully compatible with Firefox based on some slight changes in the manifest.json declaration. The difference in the manifest.json declaration is due to the fact that browsers have different ways of supporting Extension development, which can be seen in some manifest keys and their sub-keys supported by browsers.

c) Building Extensions on the Microsoft Edge browser

This browser also uses the chromium core and only differs in the interface, so it can use the version of Chrome. You can also refer to other support keys on [22].

5.3 Developing API for detecting phishing URLs

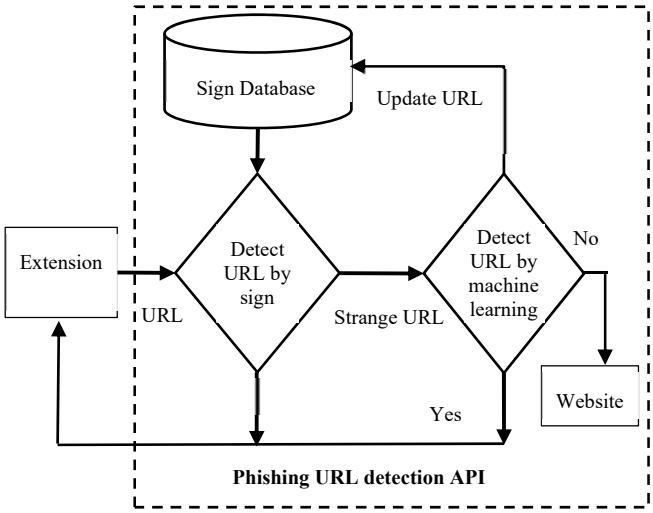

Fig.3:Architecture of phishing URL detection API

Figure 3 illustrates the architecture of the phishing URL detection API. Accordingly, the phishing URL detection API includes two main modules: a phishing URL detection module by signs and a phishing URL detection module using machine learning. Thus, after receiving URL information from the Extension, the API checks whether this URL is in the API's sign database or not. Components of the sign database will be described later in the paper. If the URL is in the sign database, return the result to the Extension to notify the user. Otherwise, if this URL is not in the sign database, API will check it again by using machine learning. At here, this URL is extracted its behaviors and features. The list of URL features and behaviors is presented in Table 1 of the paper. After obtaining the features and behavior of the URL, the API uses the RF algorithm to check whether this URL is a phishing URL or a clean URL.

6. Conclusion

In this paper, we have built an application for detecting phishing URLs based on machine learning algorithms and the ruleset. The initial purposes of the study have been solved by some features and abnormal behavior of URLs, and machine learning algorithms. The experimental results

have shown the effectiveness and suitability of the proposed model with the task of detecting phishing URLs. In the future, we will continue to improve the new features of the graph to enhance the efficiency of the phishing URL detection method.

References

[1] D. Sahoo, C. Liu, S.C.H. Hoi: *Malicious URL Detection using Machine Learning: A Survey*. CoRR, abs/1701.07179, (2017).

[2] M. Khonji, Y. Iraqi, A. Jones: *Phishing detection: a literature survey*. IEEE Communications Surveys & Tutorials, vol. 15(4), pp. 2091–2121 (2013).

[3] Zhiqiang Wang, Xiaorui Ren, Shuhao Li, Bingyan Wang, Jianyi Zhang, Tao Yang: *A Malicious URL Detection Model Based on Convolutional Neural Network*. Security and Communication Networks, vol. 2021 (2021). https://doi.org/10.1155/2021/5518528.

[4] Kumi, S.; Lim, C.; Lee, S.-G: *Malicious URL Detection Based on Associative Classification*. Entropy 2021, 23, 182 (2021). https://doi.org/10.3390/e23020182.

[5] Y. Xin et al.: *Machine Learning and Deep Learning Methods for Cybersecurity.* In: IEEE Access, vol. 6, pp. 35365-35381, (2018). https://doi.org/10.1109/ACCESS.2018.2836950.

[6] W. Yang, W. Zuo, B. Cui: *Detecting Malicious URLs via a Keyword-Based Convolutional Gated-Recurrent-Unit Neural Network.* In: IEEE Access, vol. 7, pp. 29891-29900 (2019). https://doi.org/10.1109/ACCESS.2019.2895751.

[7] Yan Ding, Nurbol Luktarhan, Keqin Li, Wushour Slamu: *A keyword-based combination approach for detecting phishing webpages.* Computers & Security, vol.84, pp. 256-275 (2019).

[8] Sheikh Shah Mohammad Motiur Rahmana, Takia Islam, Md. Ismail Jabiullah: *PhishStack: Evaluation of Stacked Generalization in Phishing URLs Detection*. Procedia Computer Science, vol. 167, pp. 2410-2418, (2020).

[9] Dipankar Kumar Mondal, Bikash Chandra Singh, Haibo Hu, Shivazi Biswas, Zulfikar Alom, Mohammad Abdul Azim: *SeizeMaliciousURL: A novel learning approach to detect malicious URLs*. Journal of Information Security and Applications, vol. 62 (2021).

[10] Cho Do Xuan, Hoa Dinh Nguyen, Tisenko Victor Nikolaevich: *Malicious URL Detection based on Machine Learning*. International Journal of Advanced Computer Science and Applications (IJACSA), vol. 11(1) (2020). http://dx.doi.org/10.14569/IJACSA.2020.0110119.

[11] Leo Breiman: *Random Forests*. Machine Learning, vol. 45(1), pp. 5-32 (2001).

[12] John Shawe-Taylor, Shiliang Sun: *Kernel Methods and Support Vector Machines*. Academic Press Library in Signal Processing, vol. 1, pp. 857-881 (2014).

[13] https://chongluadao.vn/

[14] https://Openphish.com

[15] https:// Phishing.army

[16] https:// Online.gov.vn

[17] https:// Tinnhiemmang.vn:

[18] Developer Information. https://www.phishtank.com/developer_info.php. [Last accessed 9/2021].

[19] URLhaus Database Dump. https://urlhaus.abuse.ch/downloads/csv[Last accessed 9/2021].

[20] Dataset URL. http://downloads.majestic.com/majestic_million.csv. [Last accessed 9/2021].

[21] Malicious_n_Non-MaliciousURL. https://www.kaggle.com/antonyj453/ urldataset#data.csv. [Last accessed 9/2021]

[22] JSON manifest file example. https://docs.microsoft.com/en-us/microsoft-edge/extensions/api-support/supported-manifest-keys/json-manifest-example. [Last accessed 9/2021]