A Deep Learning Based Online Malicious URL and DNS Detection Scheme

Jianguo Jiang[1], Jiuming Chen[1,2], Kim-Kwang Raymond Choo[3], Chao Liu[1], Kunying Liu[1], Min Yu[1,2](✉), and Yongjian Wang[4](✉)

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, USA
[4] Key Laboratory of Information Network Security of Ministry of Public Security, The Third Research Institute of Ministry of Public Security, Shanghai, China
yumin@iie.ac.cn, wangyongjian@stars.org.cn

Abstract. URL and DNS are two common attack vectors in malicious network activities; thus, detection for malicious URL and DNS is crucial in network security. In this paper, we propose an online detection scheme based on character-level deep neural networks. Specifically, this scheme maps the URL and DNS strings into vector form using some natural language processing methods. The CNN (Convolutional Neural Network) network framework is then designed to automatically extract the malicious features and train the classifying model. Experimental results on real-world URL and DNS datasets show that proposed method outperforms several state-of-art baseline methods, in terms of efficiency and scalability.

Keywords: Network security · Malicious URL detection · Online detection CNN

1 Introduction

As more of our devices go online, cyber threats seeking to exploit vulnerabilities in people, process and technologies will be increasingly prevalent [1, 2]. For example, the recent WannaCry ransomware virus reportedly infected more than 300,000 devices in at least 150 countries, denying access to data stored on the compromised devices. While there is a wide range of attack vectors, a common tactic used is to lure users to visit malicious websites by clicking on a malicious URL. For example, the number of unique phishing websites detected by the Anti-Phishing Working Group in October 2016, November 2016, and December 2016 is 89232, 118928, and 69533, respectively [3]. As explained in the report, "a single phishing site may be advertised as thousands of customized URLS, all leading to basically the same attack destination".

Therefore, one way of reducing phishing and other cyber attacks is to have the capability to efficiently detect and block malicious URLs, as well as the capability to circumvent efforts used by cyber attackers such as URL obfuscation techniques.

Conventional malicious URL detection methods generally rely on the features extracted based on expert input or using machine learning techniques [4]. Such methods mainly construct massive feature sets, in order to provide a comprehensive coverage. However, in practice, these methods may have high false alarm rate and have a number of limitations, such as the following:

(a) A significant increase in the number of websites and size of network traffic complicate efforts to efficiently and effectively detect malicious URLs (e.g. due to the presence of a large number of new features required for malicious URL detection).
(b) Imbalanced dataset. In comparison with the total volume of online traffic, the number of malicious URLs is relatively small (perhaps analogous to the saying 'finding a needle in the haystack'). Such imbalance (between normal URLs and malicious URLs) can lead to an unstable classification model.
(c) Constant evolution of attack techniques. Attackers often use a wide range of techniques to circumvent or avoid existing detection technologies.

Thus, in this paper, we present an online malicious URL detection scheme by combining deep neural network with natural language processing and threat intelligence. This allows us to automate the extraction of hidden features within the URL strings. Specifically, we design a convolutional neural network (CNN) based deep learning network to train the classification model. In order to map the URL strings into vector, we use the character-level word embedding method to parse the URL inputs to vectors. We then demonstrate the utility of our approach using real-world datasets. The detection scheme combines both deep learning and threat intelligence for malicious URL detection. What's more, the scheme proposed is a general detection scheme for short text detection problem in the security field such as malicious DNS detection.

In the next section, we review related literature. In Sects. 3 and 4, we present our scheme for malicious URL detection and evaluate the scheme, respectively. Finally, in Sect. 5, we conclude the paper and discuss future work.

2 Related Literature

Existing literature on malicious URL detection can be broadly categorized into blacklist based methods, features sets based methods, and machine learning based methods, as well as URL based methods and content based methods.

Webpage content is a rich information source that can be leveraged for detection [5]. Content-based methods are useful for offline detection and analysis but are generally not effective in online detection (e.g. significant latency, as scanning and analyzing page content is computationally intensive).

In this paper, we focus on online detection of malicious URL. Therefore, we will now discuss related literature on URL based methods. URL based detection methods use only the URL structures (e.g. length, domain, name length and number of dots in the URL) for detection. Such methods have been widely used due to its efficiency. These methods usually extract lexical features, either via artificial extraction or automated extraction. They can be divided into two categories, namely: machine learning based detection methods and manually constructed feature sets.

For example, McGrath and Gupta [6] analyzed the differences between normal URLs and phishing URLs to extract features that can be used to construct a classifier for phishing URL detection. Yadav et al. [7] examined more features, such as differences in bi-gram distribution of domain names between normal URLs and malicious ones. These and other related methods require the construction of large feature sets, and the detection outcome relies on the quality of these features. These features are extracted manually by experts and updating these feature sets can be challenging and time consuming. These methods also have a high false positive rate.

To mitigate these two limitations (high false positive rate and difficult to update), researchers have started examining the potential of using machine learning algorithms. In such approaches, the malicious detection problem is viewed as a classification or clustering problem, and machine learning algorithms (e.g. K-means, KNN, decision tree and SVM [8]) are used to train the classify model and extract relevant features. The machine learning based methods firstly construct an annotated URL dataset including both malicious and normal URLs. Then, some machine learning methods are used to train the classification model. Each algorithm has some specific advantages and weakness in malicious URL detection, as summarized in Table 1:

Table 1. Comparative summary of machine learning based detection methods [9, 10].

Model	Speed	Accuracy	Interpretability	Dataset size	Limitation
Bayes	High	Low	Good	Large	Need to assume the data is independent
SVM	Low	High	Pool	Small	Sensitive to data and parameters
Logistic regression	High	High	Good	Large	Maybe non-convergence

In this paper, we use character-level CNN network to automatically extract features hidden within the URL strings, as deep learning based methods have a strong generalization ability. We will present our approach in the next section.

3 Proposed Approach

In this section, we describe our approach to classify URLs and domain names based on CNN (Convolutional Neural Network). DNS content can be viewed as URL content, so we only describe the approach and implementation to classify URL. However, the proposed system can also be used to classify DNS.

CNN network has been widely used in image recognition [11, 12], perhaps due to its ability to directly perform some convolution operations on the original pixel binary data to find hidden features hidden between pixels. This allows one to extract features automatically without the need for manual extraction.

We posit that CNN can also be used in word sequence feature mining with neural language processing, and in our context, both URL and DNS can be viewed as a word sequence. In other words, malicious URL and DNS detection is similar to sentence classification. However, we cannot directly use deep learning methods to detect malicious URLs or DNS without solving the following limitations:

(1) Training time for deep learning model typically ranges from several hours to several days. Thus, it may not be realistic to constantly update the (deep learning) model.
(2) Construction of URL and DNS are more specific compared to other sentence classification scenes. Therefore, the framework for neural network needs to be specifically designed.

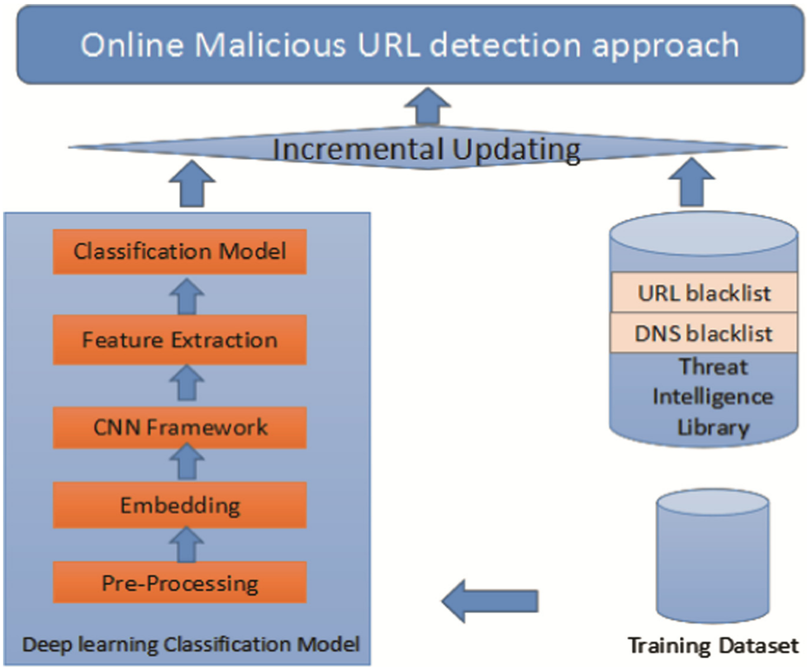

Fig. 1. Proposed online malicious URL detection approach.

In our proposed approach (see Fig. 1) consists of three main components, as follows:

(1) Dataset. The real-world dataset of URLs and DNS can be downloaded (e.g. from collaborating entities, such as APWG) or crawled from some URL or DNS sharing websites – see Sect. 3.1.
(2) Deep learning classification model, which consists of five processes such as preprocessing the input data and training a classification model using deep learning method – see Sect. 3.2.

(3) Incremental update, which allows one to periodically and incrementally update the classification model, based on existing threat intelligence data – see Sect. 3.3.

3.1 Training Dataset

When building the training dataset, we need to define the URL string and its feature, as well as the evaluation method for our approach.

3.1.1 Data Characteristics

The URL string contains three different semantic segments, namely: domain name, directory path and file name. The URL and DNS strings consist of numbers, letters and symbols such as "?", "=", and "&". We define the pattern that could be used to classify the malicious URLs or normal ones as follows:

A URL string is a tuple p = (h, d, f), where h is a URL segment pattern corresponding to the domain name, $d = \{s_1, s_2, \ldots s_n\}$ is a URL sequential patterns corresponding to the directory path, and f is a URL segment pattern represent the file name. For malicious URL strings p = (h, d, f) and normal malicious $p' = (h', d', f')$, if there is a text fragment pattern t' or other patterns t'' such as URL length is covered by p but not covered by p $'$, then we view t' and t'' as features which can be used to classify the URL. We seek to automatically find out these features and use them to build an online detection system.

The following are malicious URL examples:

http://www.aaa.com/1.php?Include=http://www.bbb.com/hehe.php
http://www.sqlinsertion.com/adminlogin.php/**/and/**/1=1.

3.1.2 Model Evaluation

To evaluate the efficiency and accuracy of the detection model, the recalling rate and precision rate are widely used as metrics as they are simple to interpret [10]. We use the number of mislabeled URLs and the precision rate to compare the accuracy between our model and the baseline model. In addition, to evaluate the efficiency of the model, we compare the execution time of one million URLs detection between our model and the baseline model. The indicators we used for model evaluation are defined as follows:

FN (False Negatives) denotes the number of URLs that are normal but classified as malicious, and FP (False Positives) denotes the number of URLs that are malicious but classified as normal. TN (True Negatives) and TP (True Positives) respectively denote the number of URLs which are malicious, normal and are correctly classified.

Mislabeled number: $FN + FP$
Accuracy rate: $TN/FN + TN$.

3.2 Character-Level Deep Learning Framework

Using some neural language processing method to map the input URL and DNS string to vector, we design a character-level CNN network to train the classification model (see Fig. 2). The deep learning model is described in Sects. 3.2.1 to 3.2.3.

3.2.1 Pro-processing

Since HTTP and HTTPS protocols are often used, the "http://" and https:// could be safely omitted from the detection. URLs generally consist of numbers, letters and some symbols. In our approach, we filter special symbols such as "_" and "#" which have been deemed to have little effect on the classification results. After pro-processing, the dataset can be more concise to reduce the time and resource requirements in the following steps.

3.2.2 Embedding

We need to map the input sequence URL to vectors as the start of the deep learning framework. We use one-hot which is a famous embedding method in NLP. Our model starts with length L sequence of characters and embeds them into an L * M matrix. Our model views the input URL or DNS string as characters sequence. Then we transform the sequence of characters to a sequence of such m sized vectors with fixed length L. Any character exceeding length L is ignored, and any characters that are not in the alphabet including blank characters are quantized as all-zero vectors.

The alphabet used in all of our models consists of 50 characters, including letters, digits and 14 other special characters. The non-space characters are:

abcdefghijklmnopqrstuvwxyz0123456789
-;!?:@#$^*% = <>

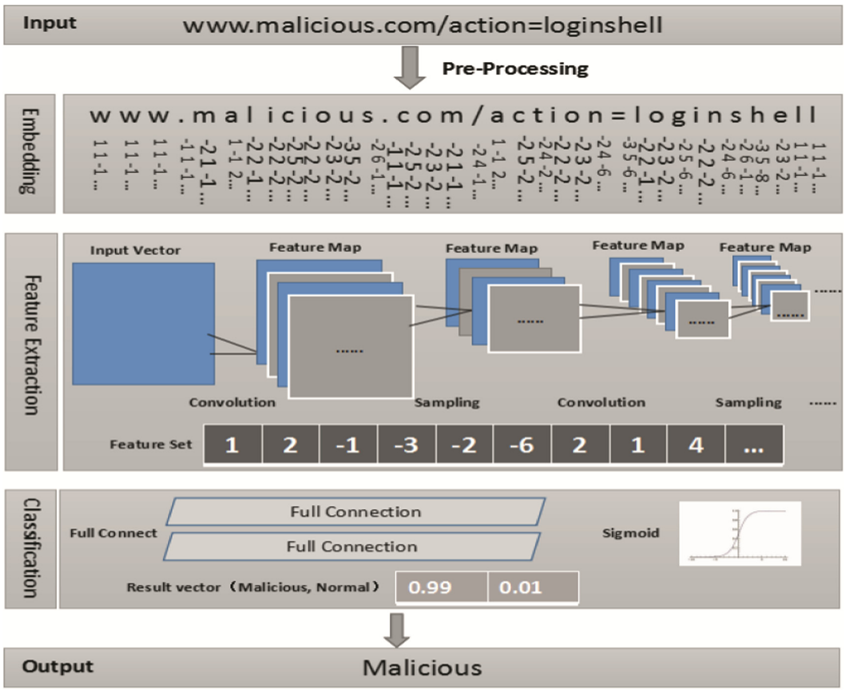

Fig. 2. Deep learning model architecture.

It appears that the value of 256 can capture most of the URL string and considering the balance between accuracy and efficiency of model training, we set the parameter L = 256 and M = 50 for our experiment described in Sect. 4.

3.2.3 CNN Framework and Classification Model

The system could automated extract features of the lexical features of these URLs using CNN layers after we embed the input strings into 256 * 50 matrix. The CNN framework consists of convolutions layers and pooling layers. The multiple kernel convolutions can learn the local features and the pooling layer such as Sum pooling layer can aggregate the results of the multiple convolutions. We initialize the weights using a Gaussian distribution, and we use layer-wise Batch Norm and Dropout (0.5) between layers to speed up training time and prevent over fitting. Table 2 shows the configurations for convolutions layers and pooling layers. The results are then concatenated together into a 1024 length vector, which represents the feature vector.

Table 2. Convolutional layers used in the experiment.

Layer	Feature map size	Kernel size	Pooling
1	256	5	3
2	256	5	3
3	256	3	N/A
4	256	3	N/A
5	256	3	3

Once we extract the feature vector, we use the full connection layer to classify the URL. We use two full connection layers, followed by the Sigmoid layer with l = 1024 units. The full connection layer learns a non-linear kernel given the convolution features, and the sigmoid layer output provides the probability that the input URL is malicious given the output of the final connection layer.

3.3 Incremental Updating

Because the training process for deep learning model is generally time consuming, the model is difficult to update for online detection system. However, to keep pace with advances in techniques used by attackers, it is necessary to update the classification model regularly (similar to patching for software and applications).

We implement the update process using current threat intelligence such as URL and DNS blacklists and the incremental learning model. Both URL and DNS blacklists are updated periodically. The incremental learning method stores information of the previous training model, which can be rolled back if necessary.

3.4 Discussion

We remark that in developing the deep learning model, other options were considered but found to be unsuitable. For example, LSTM has been widely used in text processing and machine translation [13]. However, LSTM requires significantly more time than CNN during the training of the classification model and computing [14]; thus, our choice of CNN.

For the embedding step, we use character-level mapping method instead of word-level mapping method for improved accuracy. Word-level embedding based deep learning model learns the associated features between words. However, in our context, such a model cannot traverse all the words because of the randomly generated URL strings by techniques used by attackers. Generally, the number of different characters in the URL string is less than 300. Thus, character-level feature can cover all the possible features required for effective classification.

4 Evaluations

4.1 Experiment Environment and Baseline

The evaluations were based on the Tensor Flow framework, and the experiment environment and configuration information are as follows.

- Computer Configuration: Ubuntu 16.04, memory 16 GB, CPU i7
- Tensor Flow version: 1.1.0 GPU: GeForce GTX1060, 6 GB Python version: 3.5
- Training Time: 10 h.

We implemented two baseline models. The first baseline model is based on manually extracted features described in [8], which include URL length, number of ".", separators in a URL, and categorical lexical features (e.g. domain name and URL suffix tokens). These features form a very large, but sparse feature vector. To determine the accuracy between character-level embedding based deep learning and word-level embedding based deep learning, we implemented the word-level deep learning framework based on word embedding method.

4.2 Dataset

We built a large URL dataset that consists of more than 7 million URLs. These URLs were obtained from online public datasets or crawled from malicious URL sharing websites. For malicious URLs, we crawled these data from Phish Tank and Virus Total. The normal URLs were mainly downloaded from some public datasets such as Google and DMOZ. The training dataset and the test dataset were randomly assigned according to 9:1; we randomly select ninety percent labeled data for the training dataset, and the other ten percent data as testing dataset. The distribution of the dataset is shown in Table 3.

4.3 Findings

Figure 3 shows the accuracy of the detection models. The x-axis represents three detection models being compared, and the y-axis represents the average number of URLs

Table 3. Distribution of data

URL type	Training dataset (million)	Testing dataset (million)
Malicious URL	0.9	0.1
Normal URL	5.4	0.6
Sum	6.3	0.7

mislabeled for per thousand URLs in the testing dataset. It is clear that our character-level deep learning model outperforms the other baseline approaches. In addition, the deep learning method allows good generalization, which can potentially be used to mitigate techniques used by attackers to avoid detection.

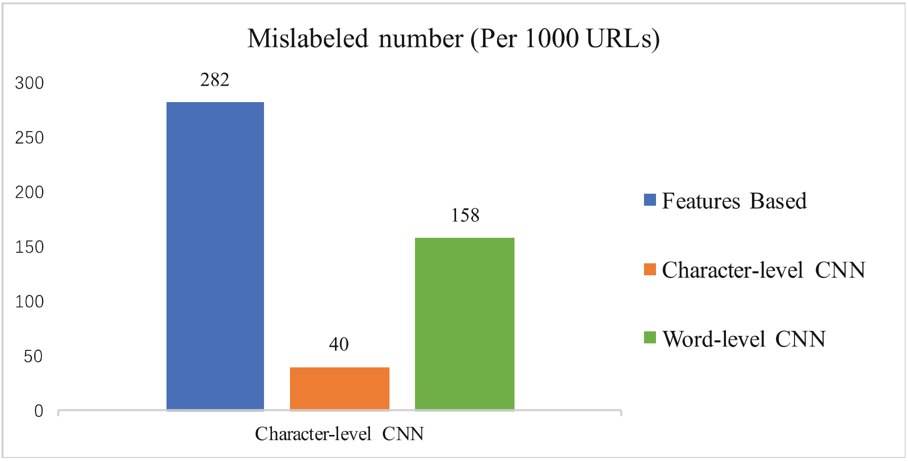

Fig. 3. Accuracy

Figure 4 shows the efficiency of the three detection models being compared. We used two testing datasets which contain 1 thousand URLs and 2 thousand URLs to determine the time required for the classification models. The x-axis represents the detection model with testing dataset, and the y-axis represents the time (in seconds) required for classification. It is clear that our model is as efficient as other baseline models. In addition, our approach allows periodic updates. For example, newly detect malicious URL patterns can be included in the updating model in real-time.

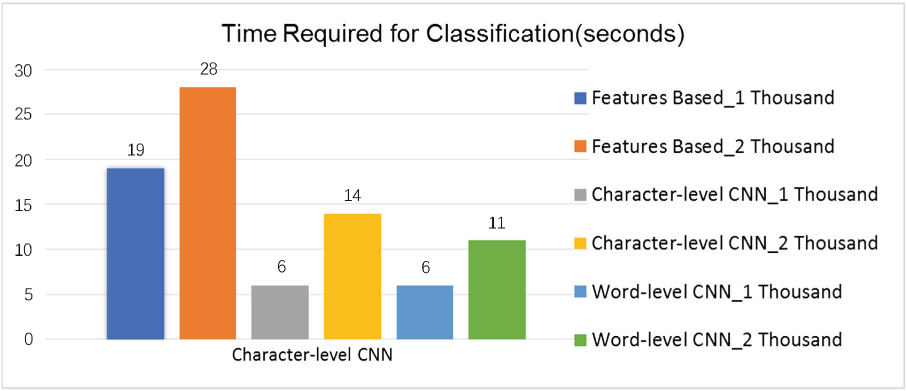

Fig. 4. Efficiency

5 Conclusion

The capability to detect malicious URLs and DNS will be increasingly important in our Internet-connected society, particularly in Internet of Things deployment.

In this paper, we proposed a character-level CNN based malicious URL and DNS detection based on the textual patterns of the URL and DNS. We evaluated our approach using real-world datasets, which demonstrated that our approach is both accurate and efficient. Besides, the scheme proposed is a general detection scheme for short text detection problem and has applications in other contexts.

Future research includes deploying the proposed approach in a real-world environment for further evaluation and fine-tuning, if necessary.

Acknowledgment. This work is supported by National Natural Science Foundation of China (No. 61173008, 61402124), Strategic Pilot Technology Chinese Academy of Sciences (No. XDA06010703) and Key Lab of Information Network Security, Ministry of Public Security (No. C17614).

References

1. Choo, K.-K.R.: A conceptual interdisciplinary plug-and-play cyber security framework. In: Kaur, H., Tao, X. (eds.) ICTs and the Millennium Development Goals, pp. 81–99. Springer, Boston (2014). https://doi.org/10.1007/978-1-4899-7439-6_6
2. Choo, K.-K.R., Grabosky, P.: CyberCrime. In: The Oxford Handbook of Organized Crime. Oxford University Press, Oxford, 24 Oct 2014
3. https://docs.apwg.org/reports/apwg_trends_report_q4_2016.pdf
4. Prokhorenko, V., Choo, K.-K.R., Ashman, H.: Web application protection techniques: a taxonomy. J. Netw. Comput. Appl. **60**, 95–112 (2016)
5. Provos, N., et al.: All your iFRAMEs point to Us. In: Conference on Security Symposium USENIX Association, pp. 1–15 (2008)

6. McGrath, D.K., Gupta, M.: Behind phishing: an examination of phisher modi operandi. In: Usenix Workshop on Large-Scale Exploits and Emergent Threats, 15 April 2008, San Francisco, CA, USA, Proceedings DBLP (2008)
7. Yadav, S., et al.: Detecting algorithmically generated malicious domain names. In: ACM SIGCOMM Conference on Internet Measurement 2010, Melbourne, Australia, November DBLP, pp. 48–61 (2010)
8. Ma, J., et al.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June – July DBLP, pp. 1245–1254 (2009)
9. Yen, T.F., et al.: Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In: Computer Security Applications Conference, pp. 199–208 (2013)
10. Huang, D., Xu, K., Pei, J.: Malicious URL detection by dynamically mining patterns without pre-defined elements. World Wide Web **17**(6), 1375–1394 (2014)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems Curran Associates Inc., pp. 1097–1105 (2012)
12. Ouyang, W., et al.: DeepID-Net: deformable deep convolutional neural networks for object detection. IEEE Trans. Pattern Anal. Mach. Intell. **pp**(99), 1 (2016)
13. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: International Conference on Neural Information Processing Systems, pp. 3104–3112. MIT Press (2014)
14. Zhang, X., Zhao, J., Lecun, Y.: Character-level convolutional networks for text classification. In: International Conference on Neural Information Processing Systems, pp. 649–657. MIT Press (2015)