DETECTING MALICIOUS URL VIA KEYWORD BASED CONVOLUTIONAL NEURAL NETWORK

V.Jayapriya[1], M. Kaviya[2], K.Keerthika[3], T.Hemalatha[4]
[1 2 3] Student, [4] Assistant Professor, Department of CSE
Krishnasamy College of Engineering and Technology, Cuddalore,

Abstract—*The main motive of URL (Uniform Resource Locator) is to use as a Web Address. However, some URLs can be also used to host unsolicited content that may potentially lead to cyber attacks. These URLs are called malicious URLs. The lack of ability of the end user to identify malicious URLs and removal of the malicious URLs can put the legitimate user in vulnerable condition. The main reason for the detection of malicious URL is that they supply an attack surface to the adversary. There have been many filtering mechanisms to detect the malicious URLs, but they are not effective .Thus we are using a Convolution Gated-Recurrent-Unit (CGRU) neural network for the detection of malicious URLs based on its characters as text classification features.*

1. INTRODUCTION

Due to the rapid development of the Internet, cyber security has become an crucial research topic, and therefore the energy waste caused by the occurrence of varied cyber security incidents is immeasurable. Malicious URL, a.k.a. malicious website, is a common and high threat to cyber security.

In recent years, a large number of Internet companies have stolen user information data, leading to the intrusion of users' online bank accounts. If the above information leakage incidents occur within the data platform of the relevant departments of the state finance and government affairs, the results are going unmanageable. The damage to national cyber security are going to be unprecedented. There are a wide sort of techniques to implement such attacks, like explicit hacking attempts, drive-by download, social engineering, phishing, watering hole, man-in-the middle, SQL injections, loss/theft of devices, denial of service, distributed denial of service, and lot of others. The Uniform Resource Locator, which is the global address of documents and other resources on the internet. A URL has two main components: (i) protocol identifier (which indicates what protocol to use) (ii) resource name (which specifies the IP address or the domain name where the resource is located). The protocol identifier and also the resource name are separated by a colon and two forward slashes.

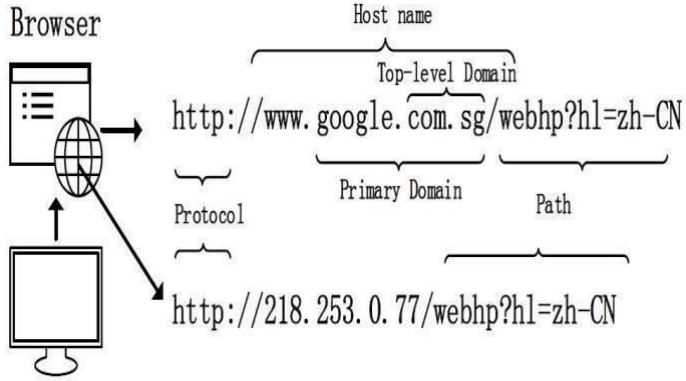

FIGURE 1:Uniform Resource Locator

Detection of malicious URLs is an crucial topic within the field of network security. In deep learning, it can be considered as a classification problem. We propose a CGRU neural network model specifically for network security.we follow the strategy of extracting malicious keywords and use the identical data set. Specifically, we built a typical URL-

based dictionary and malicious keyword library and split the first URL into short strings for input. Within the deep neural network model, the convolutional neural network is employed to extract the features of the URL and effectively represent the features. Then, the GRU is used as a pooling layer to process the obtained feature sequences to get the hidden layer nodes. Finally, we use the softmax regression method for multiple classifications of URLs.We designs a Convolutional Gated Recurrent unit (CGRU)neural network for the malicious URLs detection which revolves around on characters as text classification features. Gated recurrent unit (GRU) is employed in place of the original pooling layer to perform feature acquisition on the time dimension, leading in high-accuracy multi category results.

Most of the detection methods for malicious URL s used are supported on blacklists and rule detection. Blacklists are nothing but a database that holds URLs that are confirmed as malicious URLs in the past. This database is compiled over time. Such a method is extremely fast due to a simple query overhead, and hence is extremely easy to implement.Additionally, such a way would have a awfully low false-positive rate. However, it is almost impossible to stay up an exhaustive list of malicious URLs, since new URLs are generated everyday. Attackers use effective techniques to elude blacklists and make the users to believe the URL by modifying the URL to “appear" legitimate. This can be performed via a technique called obfuscation. The forms of obfuscation includes Obfuscating the Host with an IP, Obfuscating the host with the another domain, Obfuscating the host with large host names, and misspelling. All of those try and hide the malicious intentions of the web site by masking the malicious URL. Recently, with the development of URL shortening services, it has become a brand new and widespread obfuscation technique .Once the URLs appear legitimate, users visit them,and an attack are often launched. This can be often done by malicious code is embedded into the JavaScript.Attackers.The Blacklisting method have many limitations and especially blacklists are useless for creating predictions on new URLs ,since new ones are generating everyday.

With the development of artificial intelligence, machine learning has been widely used in increasingly more aspects. However, machine learning-based URL detection methods are depends on the artificial selection of features with data explosion and diversification.The first requirement for training a machine learning model is that that the available of training data.Machine learning-based detection models cannot be able to respond adaptively to existing URL detection tasks. Deep learning is a subfield of machine learning. By using this technique ,It helps in extracting and learn features from the foremost primitive input. It eliminates the foremost time consuming factor in machine learning and can be more flexible in adapting the complex attack behaviour.

2.MALICIOUS URL DETECTION

We mainly describe blacklist-based detection methods, traditional machine learning methods and deep learning methods based on feature extraction.

2.1 Blacklisting or Heuristic Approaches:

Blacklisting approaches are a common and classical technique for detecting malicious URLs, which often maintain a list of URLs that are known to be malicious. Whenever a new URL is visited, a database lookup is performed. URL detection was achieved by matching the IP address, hostname and directory structure. Bo Sun built an automatic blacklist generator (AutoBLG) by adding prefilters to ensure that the blacklist is always valid. . If the URL is present in the blacklist, it is considered to be malicious and then a warning will be generated; else it is

assumed to be benign. Blacklisting suffers from the inability to maintain an exhaustive list of all possible malicious URLs, as new URLs can be easily generated daily, thus making it impossible for them to detect new threats . Despite several problems faced by blacklisting , due to their simplicity and efficiency, they continue to be one of the most commonly used techniques by many anti-virus systems today.

Heuristic methods are a kind of extension of Blacklist methods.Common attacks are identified, and a signature is assigned to this attack type. A more specific version of heuristic approaches is through analysis of execution dynamics of the webpage These methods necessarily require visiting the webpage and thus the URLs actually can make an attack.Such techniques are very resource intensive, and require all execution of the code (including the rich client sided code). Another drawback is that websites may not launch an attack immediately after being visited, and thus may go undetected.

2.2 *Machine Learning Approaches:*

Although the blacklist method can effectively find known attacks, this approach is very fragile, and the blacklist update is slow Thus various machine learning methods have been introduced. These approaches try to analyze the information of a URL and its corresponding websites or web pages, by extracting good feature representations of URLs, and training a prediction model on training data of both malicious and benign URLs. There are two-types of features that can be used - *static* features, and *dynamic* features. In static analysis, we perform the analysis of a webpage based on information available without executing the URL. The features extracted include lexical features from the URL string, information about the host, and sometimes even HTML and JavaScript content. Since no execution is required, these methods are safer than the Dynamic approaches. Using this distribution information, a prediction model can be built, which can make predictions on new URLs. Due to the relatively safer environment for extracting important information, and the ability to generalize to all types of threats , static analysis techniques have been extensively explored by applying machine learning techniques

2.3Deep learning and Feature extraction:

Machine learning methods have achieved effective results in the detection of malicious URLs, but manually extracting features is time-consuming and requires constant adjustment of features to accommodate changes in URLs in combination with human knowledge; this limits the accuracy of the classification model to some extent. In recent years, deep learning has been applied to intrusion detection and surpassed traditional detection methods. J Saxe's work is most closely related to our research on character-level ,URL embedding. This paper treats URLs, file paths, and registries as a short string and uses character-level embedding and a convolutional neural network to simultaneously extract features and classifications.

3.FEATURE REPRESENTATION

This model combines the characteristics of URLs in the field of Web attacks at the character level. Our neural network model is divided into three parts:

Keyword based URL character encoding: The character type embedding module is used to map the original URL character into a low-dimensional vector, thereby encoding the original sequence as a two-dimensional floating-point matrix. In the character embedding, the malicious keyword in the URL is distinguished from the ordinary character. Such differentiation can highlight the

key part in the URL, which is advantageous in allowing the feature detection module to extract the representative feature more quickly.

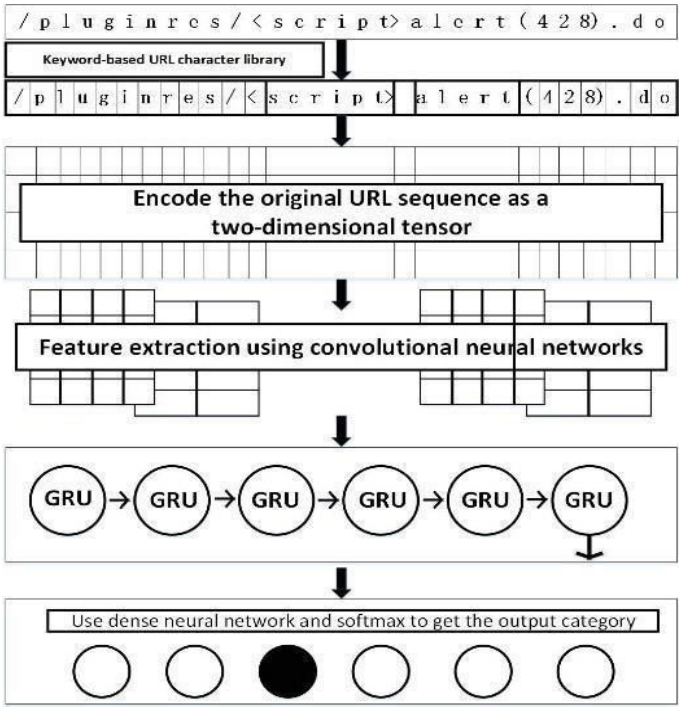

FIGURE 2: Keyword based CGRU neural netwok classification model

Feature Extraction Module: The feature detection module uses the convolutional neural network to extract features on the abstract level of the URL and uses the GRU as a pooling layer, retaining the important features on the premise of preserving the context relationship. It uses a combination of different-length convolution windows to more fully extract features at each level. To make full use of the extracted features, the features extracted from the convolution windows need to be merged. The pooling section after the convolution operation uses the GRU as a pooling layer..

Classification module: The fully connected neural network is used to classify the detected features. In our detection model, a stochastic gradient descent is used to jointly optimize the model. In terms of model evaluation indicators, in addition to the accuracy rate, precision, recall, and F1-score were selected to more fully evaluate our CGRU model.. We have abandoned the traditional pooling methods and used GRUs to learn temporal characteristics. To achieve the effect of pooling, we only output the result of the last unit in all GRU units. This simplifies the characteristics while also taking into account the timing of the URLs.

URL sttack type	Malicious keyword
SQL Injection	and, or,xp_, substr, utl, benchmark, shutdown, hex, sqlmap, md5, hex, select, union, drop, delect, concat, orderby, exec
XSS Attack	script, iframe, eval, prompt, alert, javascript, cookie, onclick, Onerror, prompt
Sensitive File Attack	access_log, text/plain, phpinfo, proc/self/cmdline, /fckeditor/
Directory Travel	../, /, ..\, \
Normal	base64, wget, curl, redict, upload, ping, shal, java.lang

TABLE 1:Malicious URL keywords

4.MALICIOUS URL DETECTION IN DEEP LEARNING

This section describes how to extract URL malicious keywords based on the characteristics of Web attacks and

how to apply the established malicious keyword database to the feature extraction module of the original URL.

4.1Character-level CNN for Malicious URL Detection :

Here we present the key ideas for building Character level CNNs for Malicious URL Detection. We aim to learn an embedding that captures the properties about the sequence in which the characters appear in a URL. To do this, we first identify all the unique alphanumeric and special characters in the dataset. Characters which appear less frequently are replaced with the unknown token denoted by . We obtained M = 96 unique characters including the and tokens. We set the length of the sequence L1 = 200 characters. URLs longer than 200 characters would get truncated from the 200th character, and any URLs shorter than 200 would get padded with the token till their lengths reached 200. Each character is embedded into a k−dimensional vector. In our work, we choose k = 32 for characters. This embedding is randomly initialized and is learnt during training. For ease of implementation, these representations are stored in an embedding matrix EM ∈ R M×k , where each row is the vector representation of a character. Using this embedding, each URL u is transformed into a matrix, u → x ∈ R L1×k , where k = 32 and L1 = 200. Using the URL matrix (for all the URLs xt∀t = 1, ..T) as the training data, we can now add convolutional layers. We use 4 types of Convolutional filters W ∈ R k×h , with h = 3, 4, 5, 6 respectively. Thus, temporal patterns in a sequence of characters of lengths 3, 4, 5, 6 are learnt. For each filter size, we use 256 filters. This is followed by a Max-Pooling layer which is followed by a fully connected layer regularized by dropout. The result is concatenated with other branches of the URLNet, finally leading to the output layer. Apart from the ability to learn structural patterns in the URL String, Character-level CNNs also allow for easily obtaining an embedding for new URLs in the test data, thus not suffering from inability to extract patterns from unseen words. As the total number of characters is fixed, the model size of the Character-level CNN remains fixed .However, Character-level CNN is not able to exploit information from long sequences of components in the URL.. Further, in scenarios where the malicious URLs try to mimic benign URLs by having a minor modification to one or few words of the URL , Character-level CNN may struggle to identify this information.. Thus, Character-level CNNs alone are not sufficient to comprehensively obtain structural information from the URL String, and it is necessary to consider word-level information as well.

4.2 Word-level CNN for Malicious URL Detection:

Word-level CNNs are similar to Character-level CNNs, except the convolutional operators are applied over words. We present a basic word-level CNN, followed by two advanced methods. Word-level CNNs. We first identify all the unique words that appear in the training corpus of the URLs. All unique words are obtained as a sequence of alphanumeric characters (including '-' and '_'), separated by special characters (e.g. '.', '/', etc.). We also use the token as an additional word to make the lengths of the URLs uniform in terms of number of words (L2 = 200). This set of unique words forms a dictionary for the URL training corpus. In the next step, we obtain the k−dimensional vector representation for each word. In our work, weset k = 32, i.e., each word is embedded into a 32-dimensional vector. For M unique words, we have to learn an embedding matrix EM ∈ R M×k. Using this representation, all the URLs are converted to their respective matrix representation (L2 × k), on which the CNN is applied. We use the same CNN architecture as in Character CNNs, i.e., we use 4 types of Convolutional filters W ∈ R k×h , with h = 3, 4, 5, 6 and for

each filter size, we use 256 filters. Here, the aim is to learn temporal properties from a sequence of words of length 3, 4, 5, 6 appearing together. This is followed by a Max-Pooling layer and the fully connected layer regularized by dropout. This output is then concatenated with the other branches of the URLNet. As the number of words can be extremely large the corresponding Embedding Matrix would become very large, and the model would suffer from memory constraints. As a result, all words that appeared only once in the entire training corpus (also called rare words) were replaced with a single token. This would substantially reduce the memory constraints faced by word-based models. During test time, a lot of URLs would contain words that have not been seen during training. Therefore, during conversion of a test URL to matrix form, the unseen words are also replaced with an unknown token .

4.3 Feature extraction:

We analyzed the characteristics of URLs of various types of attack and applied their key features to the feature extraction as a part of deep learning. We used the original URL because the input and map each character as a word vector. We used a hash to map the constructed URL keyword character library to a listing of integers, where each character corresponded to an integer. To create better use of context in characters, this article combined word embedding with the rest of the model. The original input character was converted to a low-dimensional vector, and these transformed vectors were spliced into a floating-point matrix and used as the input of the feature extraction part of the model. For the URL character library, we used the embedding layer provided by Keras to map the characters represented by the numbers into a 64-dimensional vector and then embedded the vectors into a 200×64 floating-point matrix. Compared to using the standard one-hot method, the representation dimension of the feature was greatly reduced. We randomly initialized the character's embedded vector, and this embedded vector was trainable and back-propagated to the remaining of the model. In the process of training, each embedded word vector was updated within the process of training the neural network, which made the embedded vector more semantic and more hyper plane separable.When we obtained the two-dimensional tensor accustomed to represent the input URL, we next needed to use a convolutional neural network to feature map the resulting vector.

The convolution kernel uses a convolution operation on the embedding sequence of the original URL.

The operation of each convolution window is as follows:

$$hi = f(\omega i\, x + bi)$$

here ωi is that the ith convolution kernel of the convolutional layer, t is that the length of the convolution window, bi is that the bias term corresponding to the ith convolution kernel, x may be a matrix of the kth to (k+t−1)th character vectors within the embedded character vector , and f(·) represents the activation function of the convolutional layer.

we have selected ReLU and obtained a new feature hi after convolution. In the feature extraction section, we used multiple convolution kernels to generate feature maps. Each convolution kernel had the same length. For the embedded character vector xk:k+t−1, the new features generated by n convolution

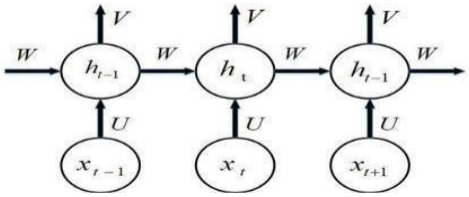

FIGURE 3: RNN internal status

kernels are to be expressed as follows:

$$W = [h_1, h_2, \cdots, h_n]$$

Here, we obtained the higher-order feature representation of the embedded characters, and also the resulting features were fed into the GRU rather than the pooling layer. After the convolution operation, we did not obtain the features directly for the classification operation. Because of the large number of data features, a large number of training parameters were required, and it was extremely easy to overfit. Therefore, the pooling layer was used for further sampling of features to get more representative features. The commonly used pooling operations are max pooling, mean-pooling, and k-max pooling. proposed a SumPool method to reduce the tensor dimension. Once the maximum pooled or averaged pooling was used to obtain the higher order features obtained by the convolution, the feature sequence information in the feature was destroyed. In neural networks that classify images, the maximum convolution guarantees translation invariance, and the average pooling fully considers all high-order features in the space. At this point, if you want to obtain further timing characteristics, you will generally continue to merge the LSTM layer or the RNN layer after the pooling layer, but this network structure is more complex than our proposed CGRU model and can create many parameters and a training time burden.

Therefore, for the URL, we use GRU rather than the standard pooling operation to raised obtain the timing-based representative features on higher-order features. Thus by using GRU we are far superior to other detection indicators.

5.RELATED WORK

Deep learning or Representation Learning has received increasing interested in recent years owing to their success in several applications.The core idea is to automatically learn the feature representation from raw or unstructured data,in an end-to-end manner without using any hand designed feature.

By Following this,we aim to use Deep Learning for Malicious URL Detection, in order to directly learn representation of the raw URL string,withoutusing any hand designed expert features.Since we aim to train Deep Networks over lexical features,a closely related area is Deep Learning for natural Language processing(NLP). Deep learning method have found success many NLP tasksincludes : text classification, machine transaction, question answering,ect.. Recurrent neural networks have been widely used due to their ability in capturing sequential information. However,the problem of exploding and vanishing gradients is magnified for them ,making them difficult to train.Convolutional Neural Networks have become excellent alternatives to LSTMsin particular showing promising performance for text classification using word-level CNNs and Character-level CNNs.

We recently noticed a work parallel to ours that attempted to use Character -level CNNs for this task.However, they ignored several types of structural information that could be captured by words in the URLs.In contrast to their work,we consider both word-level and character-level information through extensive analysis we show the importance of word-level information in capturing longer temporal patterns. Thus ,we develop novel character-level word embedding for effectively utilizing word-level information-in particular handling the presence of too many unique words,and obtaining embeddings for unseen words at test timeand we train the whole model in a jointly optimized framework.URL Net comprehensively captures the

structural information available in the URL String through both character and word-level information .Thus our proposed URL Net, where only character-level information is considered.Further, we have even shown that URLNet(full)is consistently better than just a Character-level URLNet .

6.CONCLUSION

This paper introduces a neural network model CGRU for malicious URL detection in the field of cyber security. Through comparison experiments with the feature of manually extracting malicious URLs and comparison experiments with other classification models, we proved that our model has a good effect of malicious URL detection.

The innovations of our proposed model are mainly manifested in two parts. In the data pre-processing section, our model does not use traditional artificial features but adopts a method of directly extracting features from the original input; this approach greatly ensures the pertinence and effectiveness of features. In the design part of the model, compared to the traditional pooling operation after convolution, we innovatively use GRU for pooling, which ensures the timeliness of higher-order features while streamlining the parameters required for training.

Through the analysis of the experimental results, we can see that our model has performed well enough in terms of detection. In terms of energy, our network security detection model can effectively prevent improper use and the waste of information resources and can optimize network resources and computing resources to a certain extent. In the future, we will conduct optimization research to reduce memory consumption while ensuring excellent test results. At the same time, we can study how the model is updated online to ensure that the model has a greater advantage in actual network detection.

7.REFERENCES

[1] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, “Phishnet: Predictive blacklisting to detect phishing attacks,” in 2010 Proceedings IEEE INFOCOM. San Diego, CA, USA: Citeseer, 14-19 Mar 2010, pp. 1–5.

[2] D. Sahoo, C. Liu, and S. C. H. Hoi, “Malicious URL detection using machine learning: A survey,” arXiv:1701.07179 [cs.LG], 2017.

[3] J. Saxe and K. Berlin, “eXpose: a character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys,” arXiv:1702.08568 [cs:CR], 2017.

[4] B. Cui, S. He, X. Yao, and P. Shi, “Malicious url detection with feature extraction based on machine learning,” International Journal of High Performance Computing and Networking, vol. 12, no. 2, 2018, DOI:10.1504/IJHPCN.2018.10015545.

[5] B. Sun, M. Akiyama, T. Yagi, M. Hatada, and T. Mori, “Automating url blacklist generation with similarity search approach,” IEICE TRANSACTIONS on Information and Systems, vol. 99, no. 4, pp. 873–882, 2016.