Malicious URL Detection using Deep Learning

This paper was downloaded from TechRxiv (https://www.techrxiv.org).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

01-01-2020 / 11-01-2020

CITATION

R, vinayakumar; S, Sriram; KP, Soman; Alazab, Mamoun (2020): Malicious URL Detection using Deep Learning. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.11492622.v1

DOI

10.36227/techrxiv.11492622.v1

Malicious URL Detection using Deep Learning

Vinayakumar R, Sriram S, Soman KP, and Mamoun Alazab, *Senior Fellow, IEEE*

***Abstract*—Malicious Uniform Resource Locator (URL), a.k.a. malicious website is a primary mechanism to host unsolicited content, such as spam, malicious advertisements, phishing, drive-by exploits, to name few. There is imperative to detect the malicious URLs in a timely manner. Previous studies have used blacklisting, regular expression and signature matching approaches. These approaches are completely ineffective at detecting variants of existing malicious URL or entirely newly found URL. This issue can be mitigated by proposing the machine learning based solution. This type of solution requires an extensive research in feature engineering and feature representation of security artifact type e.g. URLs. Moreover, feature engineering and feature representation resources must be continuously reformed to handle the variants of existing URL or entirely new URL. In recent times, with the help of deep learning, artificial intelligent (AI) systems achieved human-level performance in several domains and even outperformed human vision in several computer vision applications. They have the capability to extract optimal feature representation by itself by taking the raw inputs. To leverage and transform the performance improvement of them towards the cyber security domain, we propose DeepURLDetect (DUD) in which raw URLs are encoded using characters level embedding. Character level embedding is a state of the art method in NLP for representing character in numeric format. Hidden layers in deep learning architectures extract features from character level embedding and then feed forward network with a non-linear activation function estimates the probability that the URL is malicious. In this work, we evaluate various state of the art deep learning based character level embedding methods for malicious URL detection. The optimal deep learning based character level embedding model is selected by conducting various experiments. All the experiments of various deep learning based character level embedding models are run till 500 epochs with learning rate 0.001. The performance obtained by proposed method, DUD is closer and computationally inexpensive in compared to various state of the art deep learning based character level embedding methods in all test cases. Moreover, deep learning architectures based on character level embedding models outperformed n-gram representation. This is primarily due to the reason that the embedding captures the sequence and relation among all the characters of URL.**

***Index Terms*—Cyber security, Cybercrime, Malicious URL, Machine learning, Deep learning, Character embedding.**

I. Introduction

MALICIOUS Uniform Resource Locator (URL) host unsolicited information and attackers use malicious URLs as one of a primary tool to carry out cyber crimes. E-mail and social media resources such as Facebook, Twitter, WhatsApp, Orkut etc. are the most commonly used applications to spread the malicious URLs [1], [2], [3]. They host unsolicited information on the web page. Whenever an unsuspecting user visits that website unknowingly through the URL, a host will be compromised, making them victims of various types of frauds including malware installation, data and identity theft. Every year, malicious URLs have been causing around billions of dollars worth of losses [4]. These factors force the development of efficient techniques to detect malicious URLs in a timely manner and give an alert to the network administrator.

Most of the commercial products exist in markets are based on blacklisting [5]. This relies on a data base which contains a list of malicious URLs. These are continually updated by anti-virus group through scanning and crowdsourcing solutions. This can detect the malicious URL which already exists in the data base more accurately. This completely fails at detecting the variants of existing malicious URL or the entirely new malicious URL itself. In recent days, malicious authors follow mutation techniques to generate the several variants of existing malware. To cope with this, machine learning techniques are followed.

In recent days, the most commonly used approach is applying domain knowledge to extract lexical features of URL, followed by applying machine learning models. Most commonly used feature engineering is Bag-of-words (BoW) and most commonly used machine learning model is support vector machine (SVM) [6]. Though machine learning based solution can be used instead of blacklisting methodology, it suffers from many issues:

1) The conventional URL representation methods fails to capture the sequential patterns and relation among the characters.
2) Conventional machine learning models relies on manual feature engineering. This requires an extensive domain knowledge in cyber security domain and considered as a daunting task. The methods based on features are not safe in an adverserial environment.
3) Fails to hold unrevealed features and it doesn't generalize on the test data. Additionally, the number of unique words immensely large, the machine learning model faces memory constraints while training.

To alleviate the aforementioned issues, this work propose christened DeepURLDetect (DUD) which uses modern machine learning typically called as 'deep learning' with non-linear character embedding. Deep learning uses multiple hidden layers in which each layer does nonlinear projection to learn representations of multiple levels of abstraction. The main contributions of the proposed work are:

1) Detailed investigation and analysis of various benchmark

This work was supported by the Department of Corporate and Information-Services, Northern Territory Government of Australia, the Paramount Computer Systems, and Lakhshya Cyber Security Labs.

Vinayakumar R, Sriram S, and Soman KP was with Center for Computational Engineering and Networking(CEN), Amrita School of Engineering, Coimbatore Amrita Vishwa Vidyapeetham, India e-mail: (vinayakumarr77@gmail.com).

Mamoun Alazab was with Charles Darwin University, Australia e-mail: (mamoun.alazab@cdu.edu.au).

deep learning architectures are performed for malicious URL detection.

2) Various types of data sets are used in experimental analysis to find out how the models are generalizable. The difference between the time-split and random-split is shown in the experimental analysis.
3) Experiments are shown for character level embedding and n-gram representation with various deep learning architectures

The rest of the sections of this chapter are organized as follows. Section 2 discusses the related works of malicious URL detection. Section 3 provides information about URLs. Section 4 discusses the background details of benchmark text classification models and the proposed model. Section 5 provides the major shortcomings in malicious URL detection. Section 6 includes description of data set. The working flow of malicious URL detection is discussed in Section 7. Section 8 contains information of proposed architecture. Details of performance measures is discussed in Section 9. Results are discussed in Section 10. At last, conclusion and future works are placed in Section 11.

II. Related works

The detailed literature survey of machine learning based malicious URL detection, see [6]. This section discusses the most important works towards malicious URL detection.

At beginning stages, blacklisting, regular expression and signature matching approaches are most commonly used for malicious URL detection. These methods completely fails to detect new or variant of existing URL. Moreover, the signature data base has to be updated frequently to handle new patterns of malicious URL. Later, machine learning algorithms were used to effectively detect new types of malicious URL. Conventional machine learning algorithms depends on feature engineering to extract a list of features from URL. This feature engineering requires an extensive domain knowledge of URL in cyber security and a list of good features has to be carefully chosen through feature selection. There are various types of features were used in the published works for malicious URL detection. This includes blacklist features [7], [8], lexical features [9], [8], [10], host based features [11], [12], [13], content features, context and popularity based features [14], [15], [16]. Blacklist features are estimated through by checking its presence of a URL in a blacklist. This could serve as a strong feature in identifying malicious URL. Lexical features are estimated through the string properties of the URL e.g. number of special characters, length of URL etc. Host based features are obtained from the host name properties of the URL. This includes information related to WHOIS information, IP Address, Geographic location etc. Content feature are derived from the HTML and JavaScript when an unsuspecting user visits a webpage through the malicious URL. Content features includes information related to their ranking, popularity scores and source of sharing. Many existing studies have used separate feature category and as well as a combination of these features which was continually determined through domain experts.

Feature engineering is a daunting task with considering the security threats. For example, obtaining context based features consumes more time and it is high risky too. Moreover, feature selection requires extensive domain knowledge. The information which is obtained directly from the raw URL was well-known approach [8], [17]. From the published results, obtaining the lexical feature is easier in comparison to other features and it gave good performances [18]. Statistical properties of the URL string such as length of the URL, number of special characters [19] have been most commonly used and other most popular features were BoW, term frequency methods such as term document matrix (TDM) and term frequency and inverse document frequency (TF-IDF) and n-gram features [8], [18], [19]. All these features are not effective in extracting sequential order and semantics of URL. This completely disregards the information from unseen characters. Moreover, malicious URL detection solution based on the feature engineering with conventional machine learning can be easily broken by an adversary.

In recent days, the application of deep learning with character level embedding has been used for malicious URL detection [37], [38], [39], [40], [41], [42], [43], [44], [45]. In [20] compared a detailed analysis of deep learning with character level embedding and conventional machine learning with feature engineering methods for malicious and phishing URL detection. Deep learning architectures performed well in comparison to the conventional algorithms. The application of recurrent neural network (RNN) and long short-term memory (LSTM) applied to phishing URL detection [21]. For comparative analysis, lexical features and statistical URL analysis was used with random forest classifier. Both models performed well, the performance of LSTM was good in comparison to the conventional machine learning method. In [22] used convolution neural network (CNN) with character level character level Keras embedding for detecting malicious URLs, file paths and registry keys. This study showed how a unique deep learning architecture could be used on different cyber security problems. Like this, there are so many benchmark deep learning architectures exist. In this work, we evaluate the performance of various deep learning architectures for malicious URL detection.

III. An Overview of Uniform Resource Locator (URL)

A uniform resource locator (URL) is a part of uniform resource identifier (URI) which is used to identify and retrieve a resource from the internet service. A URL is composed of two parts, shown in Fig. 1. The first part defines the type of protocol, for example http or https, the second part defines the domain name or IP address and the third part defines path and its parameters to a specific resource on the web. The protocols are separated by double slash from a complete URL, domain name are separated by dot and path and its parameters are separated by single slash. A sample URL is given in Fig. 1. An adversary use URL as a main source to host malicious activities. Most commonly the malicious URLs are spreaded via email and other social media apps. Once an unsuspecting

TABLE I: Character level deep learning architectures

Name	Architecture	Task
Endgame [28]	LSTM	Detecting and categorizing domain names that are generated by DGAs
Invincea [22]	CNN	To detect malicious URLs, file paths and registry keys
CMU [34]	Bidirectional recurrent structures	Social media text classification, Twitter
MIT [42]	Hybrid of CNN and LSTM	Social media text classification, Twitter
NYU [35]	Stacked CNN layers	Text classification

user visits a malicious URL, a host will be compromised. Thus detecting the nature and type of URL is considered as a significant task.

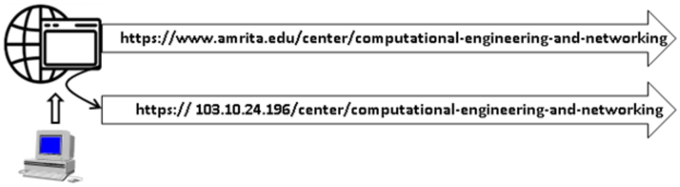

https://www.amrita.edu/center/computational-engineering-and-networking
https:// 103.10.24.196/center/computational-engineering-and-networking

Fig. 1: Uniform resource locator (URL) components

IV. Background details of Deep learning Models

A. Hybrid architecture - convolutional neural network and long short-term memory (CNN-LSTM) with character level Keras embedding

Convolutional neural network (CNN) based on character (CNN-C) level is a minimal variant of the deep CNN based on character level [36]. CNN-C primarily uses one dimensional convolution and pooling operations also called as temporal convolution and temporal pooling respectively. CNN-C extracts optimal features from the character level representation of URL. For character level representation, the character level Keras embedding representation is used. This takes 3 parameters such as dictionary size, maximum length of the character vector and embedding vector length. Initially, the character level Keras embedding weights are initialized and it can be controlled so it served as hyper parameter. The weights are optimized during backpropogation. The CNN features are passed into LSTM which facilitates to learn character level sequence representation. The flow can be formulated as given below

$$E = Character\ Level\ Keras\ embedding(URL) \tag{1}$$

$$C = CNN(E) \tag{2}$$

$$L = LSTM(C) \tag{3}$$

Finally the L is passed into fully connected layer for classifying the URL into either benign or malicious.

$$O = \sigma(L) \tag{4}$$

Where σ is sigmoid activation function

B. Character based models

Character based model takes an input text as a string of characters and automatically extracts features. These features can be used for performing different tasks for e.g. text classification. There are different character based models exist in the field of natural language processing (NLP), in this work the efficacy of them are evaluated for cyber security application namely malicious URL detection, otherwise the best character based model will not be known for malicious URL detection. All model use embedding as first layer to transform the URLs into numeric vectors. The details of the various character based models are given in Table I. The details of the various character based models are given below

1) **Character level models based on RNN**
Endgame Architecture: [28] It uses LSTM with character level Keras embedding for modeling domain generation algorithms (DGAs) to detect and categorize the domain names that are generated by DGAs. As character level Keras embedding facilitates to learn the sequence of characters of domain names which helps to preserve the order of character in domain names. Moreover, it completely avoids the feature engineering method that is an important step for classical machine learning methods. The method has outperformed well in comparison to the other methods such as hidden markov model, feature engineering and bigrams with classical machine learning classifiers. The proposed LSTM network composed of an embedding layer for URL representation, LSTM layer for optimal feature extraction and logistic regression for classification. The embedding layer maps each character to of shape 128 and passes into LSTM for feature learning and logistic regression for assigning a probability score for each domain name.
CMU Architecture: [34] CMU Architecture named as Tweet2vec for tweet representation and classification for social media data. It uses bidirectional gated recurrent unit (BGRU) to learn feature representation of Twitter data. The tweets are tokenized into a stream of characters and each character is represented by using one-hot character encoding. These one-hot representations are mapped into a character space and passes into the BGRU. It contains forward and backward GRU which facilitate to learn the sequence of characters in the domain name. Both the forward and backward GRU layers are combined using a fully connected layer and $softmax$ non-linear activation function was used for tweet classification particularly to predict hashtags of tweets. For comparative study, the tweet2vec is evaluated

on the word level tweet representation.

2) **Character level models based on CNN**
   NYU Architecture: Convolutional neural network (CNN) is most commonly used in the field of image processing. In recent days, 1D CNN has been mapped into text classification [35]. They used word based CNN and LSTM. The CNN of NYU is stacked CNN. With CNN, pretrained embedding, embedding and look up tables are used as text representation methods. With LSTM, pretrained word embedding is used as text representation method. For evaluation, they used different types of large scale data sets. They claimed that the character level CNN model performed well in comparison to the classical and deep learning models. The efficacy of deep learning models is evaluated on the classical text representation methods such as BoW, n-gram with TF-IDF.
   Invincea Architecture: To model short character strings such as URLs, file paths, or registry keys of cyber security data, [22] proposed CNN network. This CNN network is composed of character level Keras embedding layer, parallel CNN layer and followed by 3 fully connected layers. All 3 fully connected layers contain 1,024 units and $ReLU$ as an activation function. The architecture uses batch normalization and dropout regularization technique to speed up the model training and prevent from over fitting. To classify the short character strings as either legitimate or malicious, the CNN network contains a fully connected layer with unit 1 and $sigmoid$ non-linear activation function.

3) **Character level models based on hybrid CNN and RNN**
   MIT Architecture: [36] This is an extension of NYU model for tweet classification. It is composed of stacked CNN layers followed by an LSTM layer. The stacked CNN layer results in overfitting. To alleviate this, minimum number of parameters is used, i.e. one CNN layer followed by LSTM layer.

C. Problem Formulation

The objective of this work is to classify a given URL as either legitimate or malicious and classification problem is binary. Let's consider a set of URLs $U = \{(u_1, y_1), (u_2, y_2), \cdots (u_n, y_n)\}$ where u represents URL and y represents '0' for legitimate and '1' for malicious.

There are two steps involved in classification procedure, one is appropriate feature representation and second one is prediction function. Feature representation forms n dimensional vector representation x_n which can be passed into prediction function as input $y_n = sign(f(x_n))$. The main aim is to minimize the total number of misclassification in classification procedure. This can be achieved by minimizing the loss function. This type of loss function can also include regularization term. In this work f is represented as deep learning architectures.

V. Shortcomings in Malicious URL Detection

There are no publically available benchmark data sets for research in malicious URL detection. Most of the published results on malicious URL detection have used their own private data sets in evaluating the efficacy of various conventional machine learning algorithms and deep learning architectures. These private data sets are collected from various sources such as Alexa, DMOZ, Phishtank, OpenPhish, MalwareDomains, MalwareDomainList and many others. Though, these approaches cannot be regarded as generic methods due to the data sets are not common in the methods. Most of the published results haven't given any importance to the time-split methodology to divide the data into train and test. Recently, [27] discussed the importance of time-split in dividing the data into train and test sets. The data splitting methodology based on time-split is very important to meet the zero day malware detection. Recently, the background reason behind why machine learning based solutions for security are not deployable discussed by [24]. The detailed test cases that should be considered in test experiments is discussed in detail by [25]. These different test cases helps to evaluate the robustness of machine learning based solutions. Moreover, they have discussed the difficulty behind applying data science techniques for cyber security.

VI. Description of data set

It is necessary that different forms of URL have to be used to assess the performance of various conventional machine learning classifiers and deep learning architectures. There are two types of data sets are used. They are Data set 1 and Data set 2. The Data set 1 is collected from publically available sources such as Alexa.com, DMOZ directory, MalwareDomainlist.com and MalwareDomains.com, CEN Amrita Vishwa Vidyapeetham research internal network backbone. The Data set 2 is from Sophos research [26]. Most commonly used methodology for dividing data into train and test is random-split [31]. Data set 1 follows random-split. The classifier which is modeled using random-split approach is not an efficient splitting methodology to meet zero day malware detection [27]. Data set 2 follows both the random-split and time-split [27]. In the domain of Cyber security, it is good to follow the time-split. This facilitates to enhance the zero day malware detection. The detailed statistics of both Data set 1 and Data set 2 are reported in Table II.

TABLE II: Detailed statistics URL data set

Data set	Category	Legitimate	Malicious
Data set 1	**train**	212,751	175,121
Data set 1	**test**	122,406	101,616
Data set 2 random-split	**train**	43,771	43,430
Data set 2 random-split	**test**	18,516	18,857
Data set 2 time-split	**train**	39,271	47,044
Data set 2 time-split	**test**	23,016	15,243

VII. Model Configuration of Malicious URL Detection Engine

The working flow of malicious URL detection process is shown in Fig. 2. It is composed of 3 different sections. They are (1) preprocessing (2) optimal features extraction (3) classification
In preprocessing, the URLs are transformed into feature vector using text representation methods and the optimal features from the numeric vectors are extracted using various benchmark models such as Invincea, NYU, MIT, CMU, Endgame and proposed model, DeepURLDetect (DUD) and finally classification is done using fully connected layer with non-linear activation function.

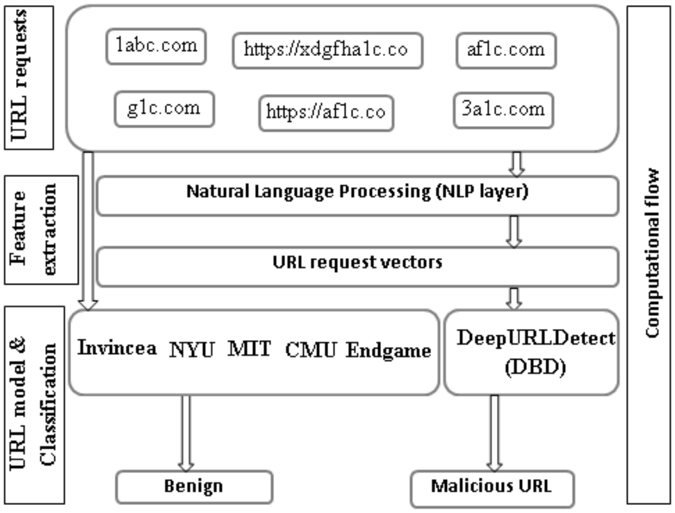


Fig. 2: Malicious URL Detection Engine

The characters in URLs are converted into lower case. This is due to domain names are case insensitive and differentiating between capital and small letters may cause regularization issue. Otherwise, the models has to be run for more number of epochs to learn the patterns of all possibilities of characters exist in the URLs. The Data set 1 corpus contains 150 unique characters, dictionary size and the maximum length of the URL is 2,307. The Data set 2 random-split and time-split corpus contains 42 unique characters and the maximum length of the URL is 246. The URL which is lesser than the maximum length is padded with 0. The detailed architecture and configuration details of DUD is shown in Table III and Table IV for Data set 1 and Data set 2 random-split and time-split respectively. In DUD, character level Keras embedding contains 128 as embedding size, as each character is mapped into 128 dimension. This helps to learn the similarity among characters by mapping the semantics of similar characters to similar vectors. All models contains character level Keras embedding as URL representation method and the dimensionality of the embedding size is set to same size to conduct a fair comparative evaluation strategy. To know the effectiveness of character level Keras embedding, 3-gram text representation method is mapped into domain names. The features of 3-grams are hash it into a vector of length 1,000 using feature hashing. These 1,000 dimensional vectors are passed into DNN for optimal feature extraction and classification, the detailed configuration parameter details of DNN is available in Table V. To avoid overfitting and speed up the training process, dropout and batch normalization is used respectively in between the DNN layers. Followed by embedding layer, DUD contains convolution layer. This layer contains 64 filters with filter length 5, activation function $ReLU$. Convolutional layer follows maxpooling with pool length 4, LSTM with 70 memory blocks. Finally the optimal features are passed into fully connected layer which contains $sigmoid$ non-linear activation function which results in 0 for legitimate and 1 for spam. The loss function is binary cross entropy and defined mathematically as given below.

$$loss(pd, ed) = -\frac{1}{N}\sum_{i=1}^{N}[ed_i \log pd_i + (1 - ed_i)\log(1 - pd_i)] \tag{5}$$

where pd is a vector of predicted probability for all samples in testing data set, ed is a vector of expected class label, values are either 0 or 1.

TABLE III: Detailed configuration parameter information of DUD for Data set 1

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 2307, 128)	19200
conv1d_1 (Conv1D)	(None, 2306, 128)	32896
max_pooling1d_1 (MaxPooling1	(None, 1153, 128)	0
lstm_1 (LSTM)	(None, 70)	55720
dense_1 (Dense)	(None, 1)	71
activation_1 (Activation)	(None, 1)	0
Total params: 107,887		
Trainable params: 107,887		
Non-trainable params: 0		

TABLE IV: Detailed configuration parameter information of DUD for Data set 2 random-split and time-split

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 246, 128)	5376
conv1d_1 (Conv1D)	(None, 245, 128)	32896
max_pooling1d_1 (MaxPooling1	(None, 122, 128)	0
lstm_1 (LSTM)	(None, 70)	55720
dense_1 (Dense)	(None, 1)	71
activation_1 (Activation)	(None, 1)	0
Total params: 94,063		
Trainable params: 94,063		
Non-trainable params: 0		

VIII. Proposed Architecture - DeepURLDetect (DUD)

The proposed architecture for malicious URL detection in an Ethernet level is shown in Fig. 3. It is typically called as DeepURLDetect (DUD). DUD is a christened hybrid of convolution and long short term memory in-house model. This module can be added to the existing scalable framework for

TABLE V: Detailed configuration details of DNN

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 128)	128128
batch_normalization_1 (Batch	(None, 128)	512
activation_1 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 96)	12384
batch_normalization_2 (Batch	(None, 96)	384
activation_2 (Activation)	(None, 96)	0
dropout_2 (Dropout)	(None, 96)	0
dense_3 (Dense)	(None, 64)	6208
batch_normalization_3 (Batch	(None, 64)	256
activation_3 (Activation)	(None, 64)	0
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 32)	2080
batch_normalization_4 (Batch	(None, 32)	128
activation_4 (Activation)	(None, 32)	0
dropout_4 (Dropout)	(None, 32)	0
dense_5 (Dense)	(None, 16)	528
batch_normalization_5 (Batch	(None, 16)	64
activation_5 (Activation)	(None, 16)	0
dropout_5 (Dropout)	(None, 16)	0
dense_6 (Dense)	(None, 1)	17
Total params: 150,689 Trainable params: 150,017 Non-trainable params: 672		

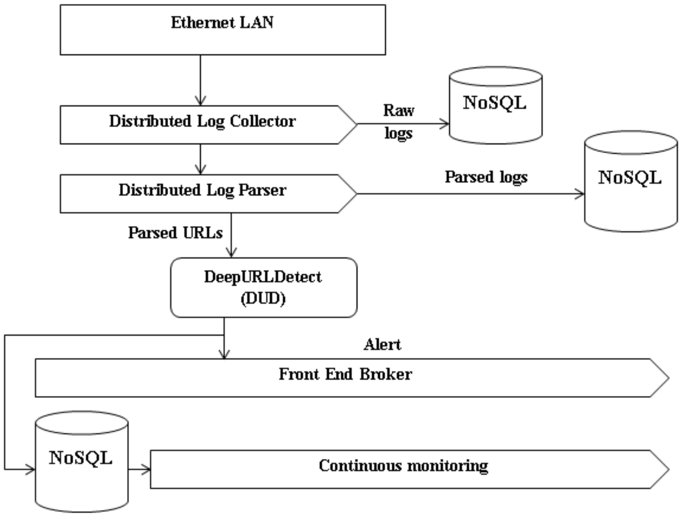


Fig. 3: Proposed Architecture - DeepURLDetect (DUD)

cyber threat situational awareness in order to enhance the malicious detection rate [32]. The architecture consists of three main modules (1) Data collection (2) Identifying malicious URL (3) Continuous monitoring

Distributed log collector collects URL logs from various different sources inside an Ethernet LAN in a passive way and passed into distributed data base. Following, the URLs are parsed using distributed log parser and fed into deep learning module. This classify the URLs into either malicious or legitimate. A copy of the preprocessed URLs are stored in distributed data base for further use. The deep learning module has Front End Broker to display the detailed information about the URL analysis. The framework contains continuous monitoring module which monitors detected malicious URLs. This continuously monitors the targeted URL once in 30 seconds. This helps to detect the malicious URL which are generated using Digitally Generated Algorithms (DGA).

IX. Performance Measures

The main objective of this work is classify whether the URL is either benign or malicious. To identify the performance of the deep learning architectures, we have used the following statistical metrics.

- True positive (TP) : malicious URL that is correctly classified as malicious URL.
- True negative (TN) : benign URL that is correctly classified as benign URL.
- False positive FP : benign URL that is incorrectly classified as malicious URL.
- False negative (FN) : malicious URL that is incorrectly classified as benign URL.

The above metrics are obtained from confusion matrix. Confusion matrix a matrix representation where each row indicates the URL samples in a predicted class and each column indicates the URL samples URL samples in an actual class. We estimate the statistical measures such as Accuracy, Precision, Recall, F1-score, true positive rate (TPR) and false positive rate (FPR) from confusion matrix and they are defined mathematically as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$\mathrm{Re}call = \frac{TP}{TP + FN} \tag{7}$$

$$\Pr ecision = \frac{TP}{TP + FP} \tag{8}$$

$$F1 - score = \frac{2 * \mathrm{Re}call * \Pr ecision}{\mathrm{Re}call + \Pr ecision} \tag{9}$$

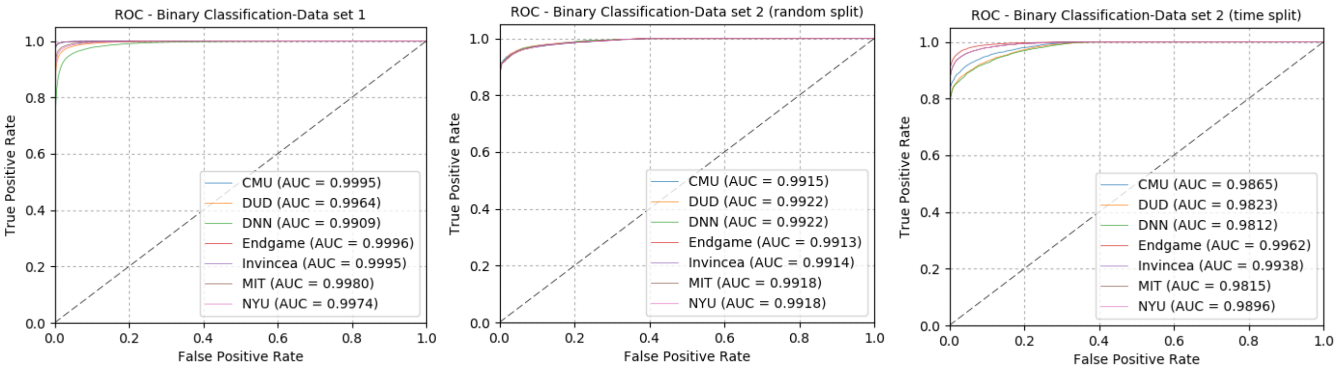
$$TPR = \frac{TP}{TP + FN} \tag{10}$$

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

The accuracy estimates the ratio of the total number of correct classifications. The precision estimates the number of correct classifications penalised by the number of incorrect classifications. The recall estimates the number of correct classifications penalised by the number of missed entries. Recall is also called as sensitivity or true positive rate. The F1-score estimates the harmonic mean of precision and recall, which serves as a derived effectiveness measurement. Receiver operating characteristic (ROC) curve denotes the performance of the classifier and plotted using TPR on Y axis and FPR on X axis. Generally, the ROC curve is used when the classes are balanced and for imbalanced classes precision-recall curve

TABLE VI: Test results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Data set 1 (both train and test from public sources)				
Invincea [22]	99.0	99.5	98.4	98.9
NYU [35]	97.7	98.0	96.8	97.4
MIT [36]	97.9	98.8	96.6	97.7
CMU [34]	99.1	99.2	98.7	99.0
Endgame [28]	99.1	99.3	98.7	99.0
DeepURLDetect (proposed)	97.2	97.4	96.4	96.9
3-gram with DNN 5 layer (proposed)	95.4	96.8	93.0	94.9
Data set 2 random-split				
Invincea [22]	96.4	97.9	93.1	95.4
NYU [35]	96.1	97.9	92.2	95.0
MIT [36]	96.0	96.1	93.6	94.9
CMU [34]	95.4	95.1	93.3	94.2
Endgame [28]	96.6	97.2	94.1	95.6
DeepURLDetect (proposed)	95.4	97.4	90.8	94.0
3-gram with DNN 5 layer (proposed)	95.0	96.2	90.9	93.5
Data set 2 time-split				
Invincea [22]	96.1	95.9	94.4	95.1
NYU [35]	95.0	95.8	91.3	93.5
MIT [36]	93.3	97.7	85.2	91.0
CMU [34]	94.1	95.9	89.0	92.3
Endgame [28]	97.1	97.6	95.0	96.3
DeepURLDetect (proposed)	93.1	94.5	87.8	91.1
3-gram with DNN 5 layer (proposed)	93.0	96.3	85.6	90.7



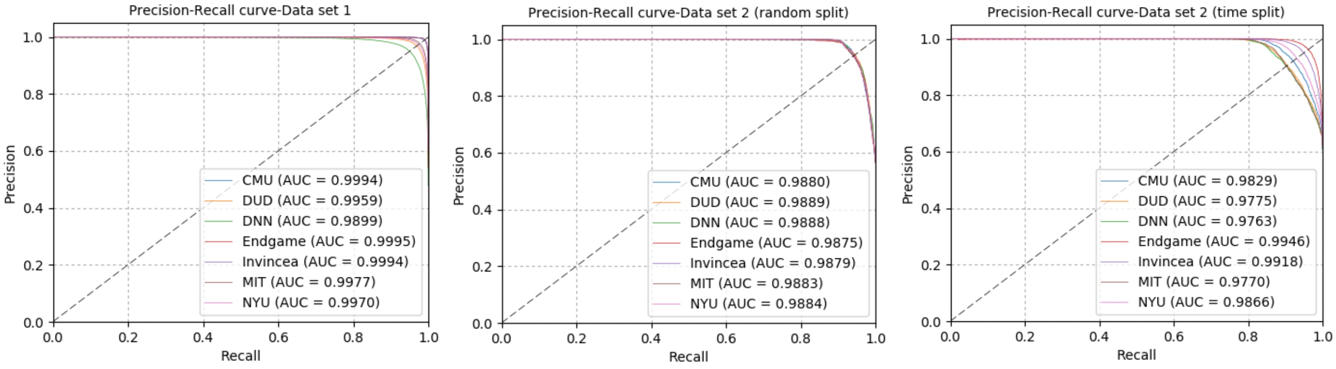
(a) (b) (c)

Fig. 4: ROC curve for (a) Data set 1, (b) Data set 2 (random-split) (c) Data set 2 (time-split)

used. To generate precision-recall curve, we estimated the trade-off between the precision and recall across varying threshold in the range of $[0, 1]$.

X. Evaluation Results and Observations

All deep learning architectures are implemented using TensorFlow [29] with Keras [30] and conventional machine learning algorithms are implemented using Scikit-learn [31]. Graphical processing unit (GPU) enabled machine are used for experimental purpose. Initially, all the models are trained with Data set 1. To evaluate the performance of the trained model on Data set 1 is tested with the test data set of Data set 1. Likewise, same approach is followed for Data set 2 random-split and Data set 2 time-split. Most of the models performed well on Data set 1 in comparison to the Data set 2 random-split and time-split. Moreover, the performance of various models on Data set 2 random-split is good in comparison to the Data set 2 time-split. This is due to the fact that the samples of test data of Data set 2 time-split is unseen during training. The detailed results are reported in Tables VI and VII. The receiver operating characteristic (ROC) curve for various models on different test data sets are shown in Fig. 4a for Data set 1, Fig. 4b for Data set 2 random-split and Fig. 4c for Data set 2 time-split with comparing two operating characteristics such as true positive rate and false positive rate across varying threshold in the range [0.0 - 1.0]. Likewise the precision-recall curve for various models on different test data sets are shown in Fig. 5a for Data set 1,



(a) (b) (c)

Fig. 5: ROC curve for (a) Data set 1, (b) Data set 2 (random-split) (c) Data set 2 (time-split)

TABLE VII: Test results. TPR and FPR are w.r.t. a threshold 0.5

Model	TPR (%)	FPR	AUC
Data set 1 (both train and test from public sources)			
Invincea [22]	88.9	0.087	0.9995
NYU [35]	89.5	0.10	0.9974
MIT [36]	85.9	0.124	0.9980
CMU [34]	87.7	0.105	0.9995
Endgame [28]	87.1	0.081	0.9996
DeepURLDetect (proposed)	87.6	0.116	0.9964
3-gram with DNN 5 layer (proposed)	87.9	0.121	0.9909
Data set 2 random-split			
Invincea [22]	93.9	0.142	0.9914
NYU [35]	94.2	0.143	0.9918
MIT [36]	95.0	0.146	0.9918
CMU [34]	94.4	0.145	0.9915
Endgame [28]	93.6	0.139	0.9913
DeepURLDetect (proposed)	95.3	0.146	0.9922
3-gram with DNN 5 layer (proposed)	95.3	0.143	0.9922
Data set 2 time-split			
Invincea [22]	85.9	0.023	0.9938
NYU [35]	83.3	0.031	0.9896
MIT [36]	82.7	0.027	0.9815
CMU [34]	89.5	0.027	0.9865
Endgame [28]	92.1	0.034	0.9962
DeepURLDetect (proposed)	79.9	0.032	0.9823
3-gram with DNN 5 layer (proposed)	71.4	0.042	0.9812

Fig. 5b for Data set 2 random-split and Fig. 5c for Data set 2 time-split with comparing two operating characteristics such as precision and recall across varying threshold in the range [0.0 - 1.0]. Obtaining better AUC in precision-recall curve indicates that the models predicts more accurately. The performance of all models have marginal difference in terms of accuracy and AUC and thus voting methodology can be employed to distinguish whether the URL is legitimate or malicious. This can further enhance the malicious URL detection rate. This is remained as one of the significant direction towards future work. They all used character level Keras embedding as text representation method and performed well over 3-gram text representation method with DNN. Deep learning with embedding based malicious URL detection can be robust solution over hand crafted features with conventional machine learning based solutions. This is due to malicious author can utilize domain knowledge to learn the hand crafted features with the aim to evade detection. They can make use of generative adversarial networks in deep learning, the details are discussed in [28].

XI. Conclusion

In this work, a comparative analysis of various deep learning based character level embedding models for malicious URL detection is done. All deep learning architectures have marginal difference in terms of accuracy. Among 5 models, two are based on RNN, two are based on CNN and one is based on hybrid of CNN and LSTM. All the models performed well and achieved around 93-98% malicious URL detection rate with false positive rate of 0.001. This indicates that if the deep learning based character level embedding models able to catch 970 malicious URL; they label only one benign URL as malicious. For comparative analysis, DNN with n-gram is used. In all test cases, deep learning based character level models performed well in comparison to the other models. All deep learning based character level embedding based models have the potential to handle drifting of malicious URLs and supply as a robust method in an adversarial environment. Though deep learning has performed well, it is good to have conventional methods such as blacklisting using regular expression, signature matching method and conventional machine learning based solutions as an initial gateway followed by deep learning based character level embedding models. The christened DeepURLDetect (DUD) can be made more robust by adding auxiliary modules such as registration services, website content, network reputation, file paths and registry keys. This can be considered as one of the significant direction towards future work.

References

[1] Tran, K. N., Alazab, M., & Broadhurst, R. (2013, November). Towards a feature rich model for predicting spam emails containing malicious attachments and urls. In 11th Australasian Data Mining Conference, Canberra.

[2] Alazab, M., & Broadhurst, R. (2015). Spam and criminal activity.

[3] Alazab, M., Layton, R., Broadhurst, R., & Bouhours, B. (2013, November). Malicious spam emails developments and authorship attribution. In Cybercrime and Trustworthy Computing Workshop (CTC), 2013 Fourth (pp. 58-68). IEEE.

[4] Broadhurst, R., Grabosky, P., Alazab, M., Bouhours, B., & Chon, S. (2014). An analysis of the nature of groups engaged in cyber crime.

[5] Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2011, December). Zero-day malware detection based on supervised learning algorithms of API call signatures. In Proceedings of the Ninth Australasian Data Mining Conference-Volume 121 (pp. 171-182). Australian Computer Society, Inc..

[6] Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179.

[7] Felegyhazi, M., Kreibich, C., & Paxson, V. (2010). On the Potential of Proactive Domain Blacklisting. LEET, 10, 6-6.

[8] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009, June). 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 (pp. 1245-1254). ACM.

[9] Kolari, P., Finin, T., & Joshi, A. (2006, March). SVMs for the Blogosphere: Blog Identification and Splog Detection. In AAAI spring symposium: Computational approaches to analyzing weblogs (pp. 92-99).

[10] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009, June). Identifying suspicious URLs: an application of large-scale online learning. In Proceedings of the 26th annual international conference on machine learning (pp. 681-688). ACM.

[11] Chiba, D., Tobe, K., Mori, T., & Goto, S. (2012, July). Detecting malicious websites by learning IP address features. In Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on (pp. 29-39). IEEE.

[12] Chiba, D., Tobe, K., Mori, T., & Goto, S. (2012, July). Detecting malicious websites by learning IP address features. In Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on (pp. 29-39). IEEE.

[13] McGrath, D. K., & Gupta, M. (2008). Behind Phishing: An Examination of Phisher Modi Operandi. LEET, 8, 4.

[14] Cao, J., Li, Q., Ji, Y., He, Y., & Guo, D. (2016). Detection of forwarding-based malicious urls in online social networks. International Journal of Parallel Programming, 44(1), 163-180.

[15] Choi, H., Zhu, B. B., & Lee, H. (2011). Detecting Malicious Web Links and Identifying Their Attack Types. WebApps, 11, 11-11.

[16] Lee, S., & Kim, J. (2012, February). WarningBird: Detecting Suspicious URLs in Twitter Stream. In NDSS (Vol. 12, pp. 1-13).

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

[18] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. 2010. Lexical feature based phishing URL detection using online learning. In Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security. ACM, 54–60.

[19] Pranam Kolari, Tim Finin, and Anupam Joshi. 2006. SVMs for the Blogosphere: Blog Identification and Splog Detection. In AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. 92–99

[20] Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Evaluating deep learning approaches to characterize and classify malicious URL's. Journal of Intelligent & Fuzzy Systems, 34(3), 1333-1343.

[21] Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., & González, F. A. (2017, April). Classifying phishing URLs using recurrent neural networks. In Electronic Crime Research (eCrime), 2017 APWG Symposium on (pp. 1-8). IEEE.

[22] Saxe, J., & Berlin, K. (2017). eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. arXiv preprint arXiv:1702.08568.

[23] Machine Learning: How to Build a Better Threat Detection Model By Madeline Schiappa, Available at https://www.sophos.com/en-us/medialibrary/PDFs/technicalpapers/machine-learning-how-to-build-a-better-threat-detection-model.pdf?la=en

[24] Sommer, R., & Paxson, V. (2010, May). Outside the closed world: On using machine learning for network intrusion detection. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 305-316). IEEE.

[25] Verma, R. (2018, March). Security Analytics: Adapting Data Science for Security Challenges. In Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics (pp. 40-41). ACM.

[26] Machine Learning: How to Build a Better Threat Detection Model By Madeline Schiappa, Available at https://www.sophos.com/en-us/medialibrary/PDFs/technicalpapers/machine-learning-how-to-build-a-better-threat-detection-model.pdf?la=en

[27] Hillary Sanders and Joshua Saxe, "Garbage In, Garbage Out: How purportedly great ML models can be screwed up by bad data"

[28] Anderson, H. S., Woodbridge, J., & Filar, B. (2016, October). DeepDGA: Adversarially-tuned domain generation and detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (pp. 13-21). ACM.

[29] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). Tensorflow: a system for large-scale machine learning. In OSDI (Vol. 16, pp. 265-283).

[30] Chollet, F. (2015). Keras.

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825-2830.

[32] Vinayakumar, R., Poornachandran, P., & Soman, K. P. (2018). Scalable Framework for Cyber Threat Situational Awareness Based on Domain Name Systems Data Analysis. In Big Data in Engineering Applications (pp. 113-142). Springer, Singapore.

[33] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[34] Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., & Cohen, W. W. (2016). Tweet2vec: Character-based distributed representations for social media. arXiv preprint arXiv:1605.03481.

[35] Vosoughi, S., Vijayaraghavan, P., & Roy, D. (2016, July). Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (pp. 1041-1044). ACM.

[36] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[37] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access, 7, 41525-41550.

[38] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2019). Robust Intelligent Malware Detection Using Deep Learning. IEEE Access, 7, 46717-46738.

[39] Vinayakumar, R., Soman, K. P., Poornachandran, P., Mohan, V. S., & Kumar, A. D. (2019). ScaleNet: scalable and hybrid framework for cyber threat situational awareness based on DNS, URL, and email data analysis. Journal of Cyber Security and Mobility, 8(2), 189-240.

[40] Vinayakumar, R., Soman, K. P., Poornachandran, P., & Menon, V. K. (2019). A Deep-dive on Machine Learning for Cyber Security Use Cases. In Machine Learning for Computer and Cyber Security (pp. 122-158). CRC Press.

[41] Vinayakumar, R., Alazab, M., Jolfaei, A., Soman, K. P., & Poornachandran, P. (2019, May). Ransomware triage using deep learning: twitter as a case study. In 2019 Cybersecurity and Cyberforensics Conference (CCC) (pp. 67-73). IEEE.

[42] Vinayakumar, R., Soman, K. P., Poornachandran, P., Akarsh, S., & Elhoseny, M. (2019). Deep Learning Framework for Cyber Threat Situational Awareness Based on Email and URL Data Analysis. In Cybersecurity and Secure Information Systems (pp. 87-124). Springer, Cham.

[43] Harikrishnan, N. B., Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2019). Time Split Based Pre-processing with a Data-Driven Approach for Malicious URL Detection. In Cybersecurity and Secure Information Systems (pp. 43-65). Springer, Cham.

[44] Vazhayil, A., Vinayakumar, R., & Soman, K. P. (2018, July). Comparative Study of the Detection of Malicious URLs Using Shallow and Deep Networks. In 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[45] Harikrishnan, N. B., Vinayakumar, R., Soman, K. P., Poornachandran, P., Annappa, B., & Alazab, M. (2019). Deep learning architecture for big data analytics in detecting intrusions and malicious URL. Big Data Recommender Systems: Algorithms, Architectures, Big Data, Security and Trust, 303.