A Distributed Deep Learning System for Web Attack Detection on Edge Devices

Zhihong Tian[1], Chaochao Luo[2], Jing Qiu[1,*], Xiaojiang Du[3], Mohsen Guizani[4]

***Abstract*—With the development of Internet of Things (IoT) and cloud technologies, numerous IoT devices and sensors transmit huge amounts of data to cloud data centers for further processing. While providing us considerable convenience, cloud-based computing and storage also bring us many security problems, such as the abuse of information collection and concentrated web servers in the cloud. Traditional intrusion detection systems (IDS) and web application firewalls (WAF) are becoming incompatible with the new network environment, and related systems with machine learning or deep learning are emerging. However, cloud-IoT systems increase attacks against web servers, since data centralization carries a more attractive reward. In this paper, based on distributed deep learning, we propose a web attack detection system that takes advantage of analyzing URLs. The system is designed to detect web attacks and is deployed on edge devices. The cloud handles the above challenges in the paradigm of the Edge of Things (EoT). Multiple concurrent deep models are used to enhance the stability of the system and the convenience in updating. We implemented experiments on the system with two concurrent deep models and compared the system with existing systems by using several datasets. The experimental results with 99.410% in accuracy, 98.91% in TPR and 99.55% in DRN demonstrate the system is competitive in detecting web attacks.**

***Index Terms*— Distributed Deep Learning, Distributed System, Edge of Things, Web Attack Detection**

I. Introduction

Information has existed everywhere around us since the development of information technology, and the Internet has changed the traditional ways of our daily life. At the same time, web applications that are widely applied in different fields have been the most popular applications of the Internet, and the rise of cloud technologies have made web services faster and more convenient. For convenience, users can get web services from providers by inputting simple Uniform Resource Locator addresses (URLs) in browsers. However, it is possible for attackers to hack into servers with well-designed URLs sent to the servers.

Gartner has reported that 75 percent of attack behaviors were discovered in the application layer and that web servers have been the primary targets of hackers during the cyber security incidents that have happened in recent years. There are two reasons why hackers are inclined to hack into web servers. On the one hand, millions of users' private information (e.g., ID numbers, addresses, bank accounts) are stored in the databases of servers, and hackers will get a considerable profit by selling the data. On the other hand, hackers can inject malicious code into the source documents of servers, and users will get hacked when they browse or download these documents. Hackers are then able to do what they want on victims' computers or smart phones. Common web attacks include SQL injections, code injections, sensitive data exposures, cross-site scripting (XSS) path traversal and more. Injection attacks, including SQL injections and code injections, rank first and XSS ranks 7th in the report of the Top 10 security threats released by OWASP in 2017.

In the cloud, common Intrusion Detection Systems (IDS), such as snort and Web Application Firewalls (WAFs), are used to protect against web attacks, but they are still penetrable because most WAFs rely on regular expression-based filters created from known attack signatures, and they require much expert configuration. Deep learning has been implemented in many fields with prominent achievements. For example, deep learning can be used in automatic translation machines to improve the reliability, in recommendation systems to suggest what customers are interested in or want to buy next, and in image recognition systems to detect objects. In the meantime, deep learning has been applied to cyber security in some studies due to its capability of analyzing and self-learning. Detecting web attacks within URLs from normal users and attackers using deep learning is challenging, and there are four major problems:

- An effective way to transform all kinds of URLs into representations is important in view of the various ways that different attacks hide in their URLs.
- Different attacks show different signatures in their URLs, and thus feature selection is not easy.
- Most applications of deep learning in cyber security have only one model to do their detection, and it is not easy to

Funding: This work is supported by the National Natural Science Foundation of China (U1636215, 61871140, 61872100, 61572153), the National Key research and Development Plan (2018YFB0803504), and the Guangdong Province Key Research and Development Plan (2019B010137004).

1 Cyberspace Institute of Advanced Technology, GuangZhou University, GuangZhou, China (tianzhihong@gzhu.edu.cn, qiujing@gzhu.edu.cn)

2 Institute of Computer Application, China Academy of Engineer Physics, Mianyang, Sichuan, China (luochaochaoforaffair@gmail.com)

3 Department of Computer and Information Sciences, Temple University, Philadelphia PA 19122, USA (dxj@ieee.org).

4 Department of Computer Science and Engineering, Qatar University, Qatar (mguizani@gmail.com).

* Corresponding author: Jing Qiu, Email: qiujing@gzhu.edu.cn

Manuscript received April. 19, 2019;

update the system.

In the environment of an IoT cloud, its centralization feature affects the application of distributed services, such as the network security mechanism for IoT applications. Novel security models, controls, and decisions distributed at the edge of the cloud are queried by new IoT applications in the new paradigm named the EoT. In this paper, we proposed a distributed system for web attack detection from URLs by utilizing deep learning techniques. These include detectors based on novel deep learning architectures such as Convolutional Neural Networks (CNNs) and models in Natural Language Processing (NLP) [1]. Specifically, our work makes the following contributions:

- We find a method to represent all kinds of URLs. Briefly, we replace every word or character with words from dictionaries that are previously defined in the light of different types of attacks.
- The system can distinguish anomalous requests and normal ones by automatically learning features.
- To enhance the stability of the detection system, multiple concurrent models are applied in our system.
- We propose a generic distributed web attack detection system on edge devices of the cloud.

This paper is organized in the following manner. Section II presents a brief review of related works. In Section III, we explain the architecture of our system and describe the details about our methodology. The datasets and settings of the experiments are provided in Section IV. We present our experimental results and have a short discussion in Section V. In Section VI, we conclude the paper and discuss our future work.

II. Related work

There have been many studies focused on utilizing deep learning to solve problems in cyber security [8-10]. Works in [2-4] concentrated on detecting malware using deep learning. Manually chosen features by analyzing the raw data were used to train the deep learning models in [2, 3] while Kim *et al*. [4] extracted the features from traffic with software instead of from the raw data. Particularly, in [4] a triggering relationship graph (TRG) was proposed to discover the behaviors of software and inputs into long short memory units (LSTM) to detect malware. Similar to the works above, the approach proposed by Du et al. [5] was able to perform malware detection. Not only that, their work can discover intrusive behaviors by using LSTMs to analyze operating system logs. Additionally, logs are also used by Rashid *et al*. [6]. To detect insider threats in a system they analyzed logs of applications with hidden Markov models (HMM). Other significant works were studied in [1] and [7]. In [1], authors were aimed at detecting malicious commands in powershells to ensure the security of operating systems, while Han *et al*. [7] focused on monitoring programmable logic controllers (PLCs) by utilizing signal form side channels.

In the domain of identifying malware traffic, Agarap *et al*. [13] proposed a new neural network architecture combining gated recurrent units (GRU) and support vector machines (SVM) for intrusion detection in network traffic data. They evaluated several models to do a comparison on the 2013 Kyoto University honeypot system’s network traffic data and KDD Cup 1999 dataset. Differently, Zhang *et al*. [22] concentrated on identifying encrypted malware traffic with contextual flow data. They extend from [19] with a data omnia approach and supervised machine learning models. Great results on real-world datasets in TLS, DNS, and HTTP showed its superiority.

A body of works, including [15-22], share a common goal with our work: detecting web attacks from URLs. Especially, in [23], authors used features selected by the GeFs [24] algorithm, manually and automatically using N-grams to train an SVM to learn those features and develop a capability of detecting web attacks. Later, Kim *et al*. [4] proposed a different way of analysis. The authors focused on more details in SQL injection detection instead of the many kinds of attacks in [23]. Queries from the database were parsed into query trees using a syntax tree, and they defined some features from the query trees and input them into an SVM classifier. Uwagbole *et al*. [25] extended the works above and proposed a novel method to detect SQL injections by modeling SQL queries with graphs that they defined. First, the authors replaced the queries with some predefined words, and then they generated a directed graph for every query and selected features from the graph. In the end, an SVM classifier trained by those features will detect a query as normal or an SQL injection attack.

It is obvious that deep learning provides new solutions for challenges in the field of cyber security. Especially, more studies will focus on extracting features automatically for the detection of web attacks. N-grams was widely used in automatic feature selection because of its great achievements in NLP tasks. There are two common limitations the existing works in web attack detection. On the one hand, only one discriminative model was chosen to do classifications in the existing works, and there is a risk that those systems using a single model are likely to be attacked by hackers, according to studies in [26, 27]. On the other hand, most approaches ignored how to update their systems. Systems would not detect changing attacks without updating in the real word. Different from existing works, in our work, we design our system by considering these two limitations.

III. Methodology

We now proceed to describe an overview of our proposed system and explain in detail every part of the system. In Subsection III-A, we describe the framework of our system and how the system takes advantage of distributed deep learning to detect web attacks from URLs on edge devices. We detail the method for processing the URLs collected online in Subsection III-B. Then, in Subsection III-C, we describe how we convert the processed URLs into vectors with models widely used in NLP. In Subsection III-D we introduce the detection models used in our system.

A. Architecture

Our system is composed of Data preparation, Discrimination and Actions. As shown in Fig. 1, URLs collected from edge devices of the cloud will be first sent to Data preparation, and then the processed data will be input into Feature Discriminator and Data Discriminator for detection. Actions will respond in accordance with the results from Discrimination.

- Data Preparation is used to convert raw URLs collected from edge devices into representations that can be input into Discriminators. First, raw URLs will be decoded and lowered in Processing, and then decoded data will be represented in a special format in data normalization, which will be discussed in Subsection III-B. Afterwards, the represented data will be sent to Data Discriminator. At the same time, in order to fit the inputs of Feature Discriminator, the represented data will also be converted into vectors in feature representation, which will be discussed in Subsection III-C.

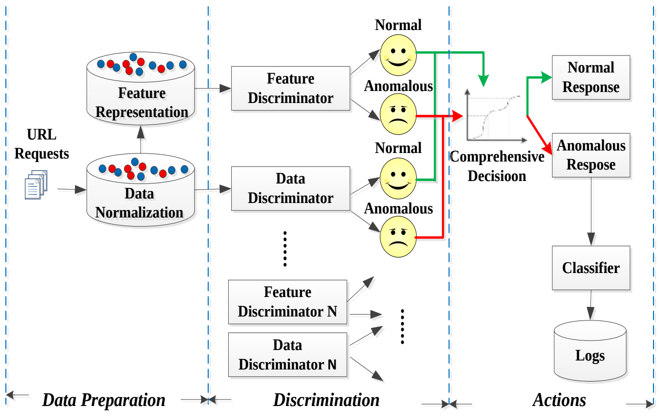

Fig. 1. The framework of proposed system

- Discrimination is responsible for distinguishing normal URL requests and Anomalous URL requests containing web attacks. We proposed multiple models in our system. Actions will perform a response directly only when all discriminators make the same decision (both are normal or anomalous) or it will respond according to the results from the Comprehensive decision. Specially, in this paper we test two concurrent models, which will be detailed in Subsection III-D, for the detection task, and the two models are trained and deployed separately. Moreover, we add results from the two discriminators in varying proportions for comprehensive decision.
- Actions are designed for responding according to the decisions from Discrimination. Actions will take a normal response if a URL is detected as normal. If not, it will take an anomalous response. At the same time, a URL will be classified by a classifier (we use the same model as Feature Discriminator) and its information will be saved in logs.

On the one hand, there are three advantages of our system: first, Discrimination with multiple concurrent models can decrease the numbers of underreporting. Second, the system can to some extent protect from attacks against deep learning. Third, it is convenient to update the system. Particularly, we

TABLE I
EXAMPLES OF KEYWORDS DEFINED

Description	Keywords
SQL	select update delete insert create alter drop order by group by truncate replace commit rollback savepoint transaction set distinct all desc null limit top …
html	doctype a abbr acronym address applet area article aside audio b base basefont bdi bdo big blockquote body br button …
Javascript	section select small source span strike strong style sub summary sup table tbody td textarea tfoot th thead time title tr track …
…	…
punctuations	/ + ? & ; = , ‘ “ ( ) < > * ! $ # | ^ { } \ -- % ~ @ . ` [ ] :

will discuss the second and the third points in Section V. On the other hand, in Actions, the Classifier and logs will provide detailed information about anomalous URLs, will help to discover unseen attacks and to will some extent help to update the system.

B. Data Normalization

In distributed nodes in IoT environments URL requests toward servers are in different formats because of the different configurations and languages in servers. In addition, attackers construct URL requests with various annotations, which change the structure or parameters of URLs to launch attacks. To detect web attacks (specially, SQL injections, XSS, and command injections) from these URL requests in various formats, we used a method to convert these URLs into the same representation. First, we define a special set of keywords consisting of SQL keywords and functions, JavaScript keywords and functions, html keywords etc. Specially, these keywords are selected by experts from those are most commonly used in daily works or attacks. Some of them are presented in Table I. Particularly, the defined set will be updated according to configurations of new servers in this cloud service. We then defined a transformation scheme showed in Table Ⅱ that transforms other words into the same expressions, eventually converting URLs into the same format utilizing the keywords and transformation scheme. For example, we collect a URL request processed in Processing as follows:

/ zabbix / php / local . php ? parameters = 1 ') or 4411 = (select count (*) from sysusers as sys1 , sysusers as sys2 , sysusers as sys3 , sysusers as sys4 , sysusers as sys5 , sysusers as sys6) and (' edcm ' like ' edcm

And it will be converted into a special expression as follows:

/ PathString / PathString / PathString . php ? PureString = Numbers ') or Numbers = (select count (*) from PureString as MixString , PureString as MixString , PureString as MixString , PureString as MixString , PureString as MixString , PureString as MixString) and (' PureString ' like ' PureString

Words in keywords or punctuations are reserved because they play meaningful role in servers, while others are parameters and are replaced by the Transformation scheme in Table Ⅱ. Particularly, we consider a special situation that words in keywords did not appear in the training data but do appear in real-time data. “SenString” in Table Ⅱ is designed to solve this problem. The procedure in Data Normalization is explained in Fig. 2.

```
Algorithm Data Normalization
Input: URLs processed by Processing P, keywords K1,
keywords from training data K2
Output: String of Normalied URLs N
1:  U [ ] ← SPLIT(P, space)
2:  n ← |U|
3:  for i=0 to n do
4:      if U[i] is not space then
5:          if U[i] is Path data then
6:            N[i] ← “PathString”
7:          elseif U[i] in K2 then
8:              N[i] ← U[i]
9:          elseif U[i] in K1 then
10:             N[i] ← “SenString”
11:         else
12:             if U[i] is number data then
13:                 N[i] ← “Numbers”
14:             end if
15:             if U[i] is String then
16:                 N[i] ← “PureString”
17:             end if
18:             if U[i] is mixed data then
19:                 if U[i] is Unicode then
20:                     N[i] ← “UniString”
21:                 elseif U[i] is Hex code then
22:                     N[i] ← “HexString”
23:                 else U[i] ← “MixString”
24:                 end if
25:             end if
26:         end if
27:     end if
28: end for
29: return N
```

Fig. 2. Data Normalization Algorithm

C. Feature Representation

Word2vec [28] was proposed to do the word embedding job, which is using vectors to represent words. This has been widely applied NLP studies. We choose to utilize CBOW, one of the word2vec models, to represent words in normalized URLs with vectors. As shown in Fig. 3, the model takes one-hot vector $X_i \in R^v$ for every word in the text as an input, a hidden vector h $\in R^v$ and a one-hot vector $Y_i \in R^v$ for the output. For the k vectors of those words that are closest to the i^{th} word will be the input, the i^{th} word will be the label. The first part is to map the input k vectors to a hidden representation h $\in R^N$ through a simple mapping $\mathrm{h} = \frac{1}{C} W \cdot (\sum_{i=1}^{k} X_i)$. W $\in R^{V\times N}$ is a matrix of weights between the input layer and the hidden layer. There is another simple mapping between the hidden layer and the output layer, and $Y_i = \mathrm{softmax}({W'}^T \cdot h)$. The error needs to be minimized, which can be formulated as follows:

$$\mathrm{E} = \arg \min_{(W,W')} L(X_i,\ Y_i)$$

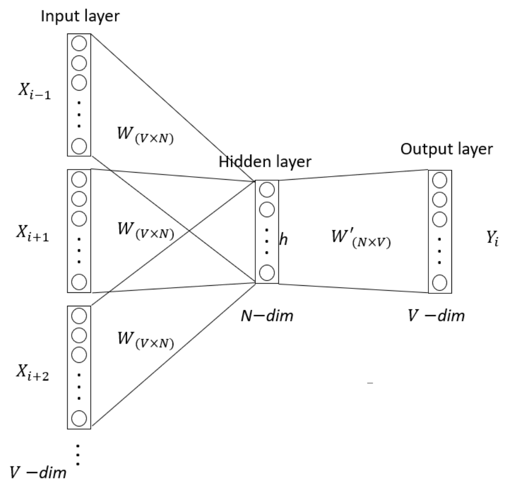

Fig. 3. Model of CBOW

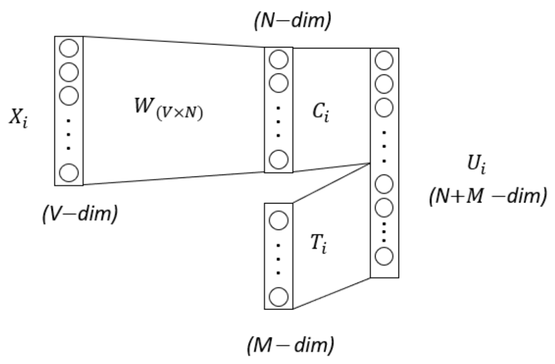

Fig. 4. Concatenating CBOW vector and TF-IDF feature together

where L is the loss function and every word X_i is mapped to a Y_i. Every word will be converted into a unique vector $C_i \in R^N$ ($C_i = X_i \cdot W$) after training. TF-IDF means the term

TABLE II
TRANSFORMATION SCHEME

Transformation	Description
PathString	Replacing the path parts in URLs. Example: http://example.com/main/index.php? will be replaced as http://example.com/PathString/PathString?
Numbers	Replacing all pure numbers in URLs.
PureString	Replacing strings which are made up of a-z and ‘-’.
MixString	Replacing strings which contain characters expect a-z and ‘-’
UniString	Replacing all Unicode data.
HexString	Replacing all Hex data
SenString	Replacing keywords didn’t appear in training data but appear in real-time data.
Punctuations are reserved	

frequency–inverse document frequency has been widely used in text classifications in NLP. We concatenate vector C_i and TF-IDF vector T_i together and get the feature vector U_i for one word, as illustrated in Fig. 4.

D. Deep learning Models

We take advantage of convolution in Feature Discriminator. A Convolutional Neural Network (CNN) is a great learning architecture widely used in computer vision. As filters slide the 2D input matrix, the dot product between filters and the same size of corresponding windows in the input are computed for feature maps. Generally, a Relu function, pooling layers and dropout layers are used in CNN architectures to optimize the neural network. Particularly, we present an amendment to a Residual network (ResNet) [29], a special CNN, by adding factors for every branch, see Fig. 5. The procedure is as follows:

$$F_1(\mathbf{x}) = \mathbf{x}$$
$$F_2(\mathbf{x}) = pool(\mathbf{x})$$
$$F_3(\mathbf{x}) = Relu(Conv(\mathbf{x}))$$
$$H(\mathbf{x}) = pool(\alpha F_1(\mathbf{x}) + \beta F_2(\mathbf{x}) + \gamma F_3(\mathbf{x}))$$

where α, β and γ are factors for every branch and will be optimized with the whole network. We call it M-ResNet in the following. The previous works have proven that CNN has a fantastic capability of performing text classification jobs. URLs are a kind of special text data. It stands to reason to use ResNet to detect anomalous URLs. We implemented eight blocks in Fig. 5 in Feature Discriminator whose structure is depicted by Fig. 6. A matrix composed of feature vectors U_i is input into Feature Discriminator. Four M-ResNet blocks are used for the length of the matrix and another four blocks are used for the height and each followed by a normalization layer and a dropout layer (omitted in Fig. 6). That is, followed by three full-connected layers, each followed by a dropout layer with a probability of 0.5 (omitted in Fig. 6), and a final output layer.

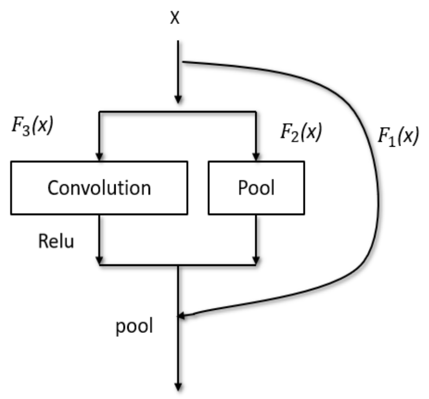

Fig. 5. The architecture of modified ResNet

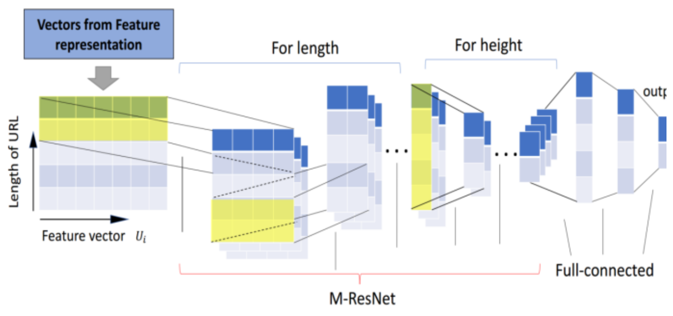

Fig. 6. The structure of Feature Discriminator

We use FastText [30] for Data Discriminator. FastText is a deep learning model for text classification created by Facebook's AI Research (FAIR). It should be made clear that FastText is used because a URL is a kind of text. This model is similar to CBOW mentioned above. Based on CBOW, the Softmax functions, N-gram features and other amendments are implemented. The results in [30] show its great capability for text classification.

IV. EXPERIMENTS

To validate our system, we implement experiments on the system with two concurrent models. In addition, we run all experiments under an environment with an Intel Core i7-6900k, 4 Kingston DDR4 16 G, 2 CUDA-enabled MSI GTX 1080Ti and Ubuntu 16.04. The structure of the system is described in Fig. 1 and uses Python 3.6.3-64 bit and the TensorFlow 1.8.0 library from Google. In the following section, we have a brief overview of the datasets used in our experiments and evaluation.

A. Datasets

Our experiments are based on three datasets: HTTP Dataset CSIC 2010, FWAF and HttpParams Dataset.

1) *HTTP DATASET CSIC 2010* has been most widely used for studies on web intrusion detection. This dataset was collected automatically on a web application in the authors' department. Significantly, more than 36000 normal requests and 25000 anomalous requests are provided in the dataset. Attacks in this dataset contain buffer overflows, information gathering, SQL injection, files disclosure, XSS (Cross-site scripting), CRLF injection, parameter tampering etc. In our experiment, we only used URLs of HTTP GET requests from the dataset.

2) *FWAF* is a dataset created by a community named FSECURITY for detecting malicious web queries with Machine learning driven web application firewalls. They collected more than 1290000 normal requests and 48000 anomalous requests from the Internet.

3) *HttpParams Dataset* is a well-known dataset on Github. It was created with several tools, include sqlmap, xssya, Vega Scanner and the FuzzDB repository. It contains 19304 normal requests and 11763 anomalous requests. Particularly, every piece of data was labeled as 'norm' meaning normal or 'anom'

meaning anomalous. Additionally, the data labeled 'anom' was marked with its type of attack ('sqli' refers to SQL injection, 'xss' refers to Cross-Site Scripting, 'cmdi' refers to Command Injection and 'path-traversal' refers to Path Traversal attacks).

We used all HTTP GET requests from CSIC 2010 in the beginning. After removing duplicated data and incorrectly labeled data from these three datasets, we extract 4812 normal requests and 5382 anomalous requests from CSIC 2010, 49969 normal requests and 44244 anomalous requests from FWAF and 18828 normal requests and 8620 anomalous requests from HttpParams Dataset. We mix these data together because of the variety of requests in distributed edge devices. It should be noted that we only use anomalous data that includes SQL injection, XSS and command injection.

B. *Settings*

We consider three main problems experimentally:

- **P1** How to transform a URL into a vector while retaining most of its information in the URL.
- **P2** Which deep learning model is suitable for extracting features from vectors of URLs and detecting anomalous requests.
- **P3** How to optimize the detecting model to get high accuracy and low false alarms.

For P1, we try several methods to convert URLs into vectors. These methods include encoding characters with their ASCII code, encoding characters with their vectors from the word2vec model, encoding words in URLs with the word2vec model and encoding words in URLs with the methods in Subsection III-B and Subsection III-C. For P2, we conduct some experiments on several machine learning and deep learning algorithms, including Naïve Bayes, X-means, SOM, C4.5, RNN etc. For P3, we propose a system described in Subsection III-A and compare it with some existing methods for detecting web attacks.

In our experiments, we evaluate the results with accuracy, recall, FP and precision, which are typical performance metrics to evaluate models used in machine learning or deep learning. In the following, TP refers to "true positive", TN refers to "true negative", FP refers to "false positive" and FN refers to "false negative".

Accuracy (ACC) indicates the proportion of all requests that are correctly detected over all the data as follows:

$$\text{ACC} = \frac{TP + TN}{TP + FP + TN + FN}$$

Recall (TPR) is the ratio of real attacks that are detected as anomalous over all attacks as follows:

$$\text{TPR} = \frac{TP}{TP + FN}$$

The FPR rate is the ratio of normal requests that are detected as attacks over all normal requests as follows:

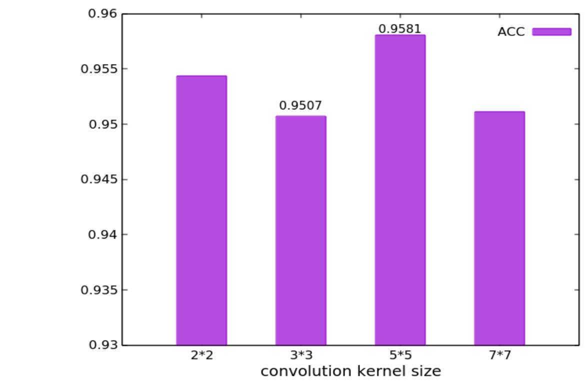

Fig. 7. Results for CNN with ASCII code

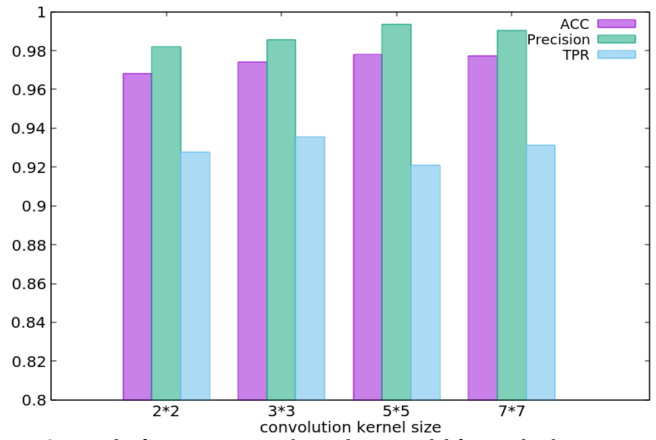

Fig. 8. Results for M-ResNet with word2vec model for single char

$$\text{FPR} = \frac{FP}{FP + TN}$$

Precision is defined as the ratio of anomalous predictions that are correct as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Particularly, we define the rate of real normal requests that are detected as normal over all normal requests (DRN) as follows:

$$\text{DRN} = \frac{TN}{FP + TN}$$

V. Results and Discussions

Fig. 7-9 and Table Ⅲ give our experimental results on the data from CSIC 2010, from which we could see the word2vec model's capability of representation for URLs. Figure 10 and Figure 11 compare our system with other proposed approaches

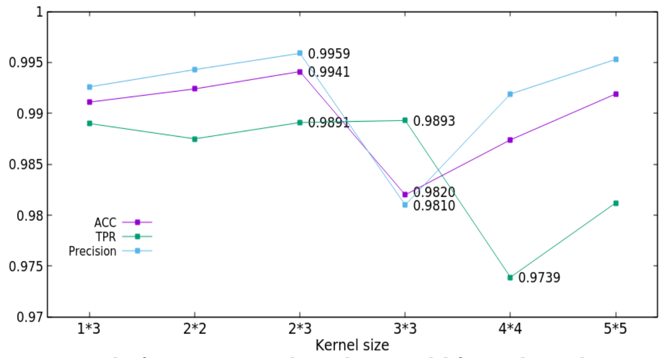

Fig. 9. Results for M-ResNet with word2vec model for single word

and show the results on a mixed dataset composed of the three datasets mentioned above.

To find an effective way to transform URLs into

TABLE III
COMPARISON ON DATASET CSIC2010 (*: REFERRING TO [20], †: REFERRING TO [22] AND ‡: REFERRING TO [21])

Model	ACC	TPR	DRN
ModSecurity with CRS*	0.5520	0.436	0.9941
EM*	0.7486	0.7516	0.7478
X-means*	0.7493	0.6837	0.9865
Naïve Bayes*	0.8408	0.5235	0.9268
SOM*	0.9282	0.9497	0.9242
C4.5*	0.9650	**0.9914**	0.8697
Model with RNN in [20]*	0.8515	0.7403	0.9546
Model with GRU in [20]*	0.9788	0.9722	0.9846
Model with LSTM in [20]*	0.9856	0.9888	0.9832
Model in [22]†	0.9649	0.9335	0.9864
Model in [21]‡	0.9881	0.9835	0.9912
M-ResNet with word2vec for word	**0.9941**	**0.9891**	**0.9955**

representations for P1, we conduct experiments with three methods. We replace characters with their ASCII code and use CNN as a discriminator, and the accuracy fluctuates approximately 0.955, as shown in Fig. 7. We then test M-ResNet with encoded characters by using word2vec's model. The results are provided in Fig. 8. We see that the average accuracy is much higher than it is in Fig. 7. It seems that the word2vec model has a better capacity of representing characters in URLs while retaining the information in the URL. It should be added that the TPR values that fluctuate approximately 0.93 are not good enough, and they are the most important metrics in detecting web attacks.

Inspecting the curves in Fig. 9, we see that the averages of these three metrics are higher than the averages in Fig. 8. Obviously, the curves of the TPR, ACC and Precision have the same trend. The results of kernel size=2×3 achieve an ACC of 0.9941 and a precision of 0.9959. They outperform the others. The best score for TPR comes from the result of a kernel size=3×3, which has only 0.0001 more than the result of a kernel size=2×3. It stands to reason that we set 2×3 as our kernel size for our model moving forward. Table Ⅲ compares several methods on dataset CSIC 2010 and shows that M-ResNet with word2vec for single words outperforms the other listed models. It can be seen from this that M-ResNet with the word2vec model for single words has the best capability of representing URLs and detecting web attacks. It should be added, however, that while the words represented by the word2vec model come from URLs, the dictionary of words is too big, and it is not easy to tackle unknown words.

To solve the problem of unknown words, we utilize the method in Subsection III-B and implement experiments with the existing models on the mixed dataset for a comparison. Particularly, we used two concurrent discriminators for the experiments below. In Fig. 10 our system, composed of Feature Discriminator and Data Discriminator, performs closely to Feature Discriminator and better than Data Discriminator. This proves that our system has the capacity to take advantage of the two concurrent models. The comparison with existing systems is shown in Fig. 10 and Fig. 11. We see that Precision histograms are almost the same while ACC and TPR histograms show clear distinctions; therefore, the model DBPF in [2] performs poorly on this dataset, while the other models perform closely to each other. The model DBPF gets a Precision score close to our system, and a better FPR score than our system. Our system performs much better than it in ACC and TPR scores, which are the two most important metrics for detecting web attacks. The model CLCNN in [20] performs the best, achieving 0.9959 in TPR, 0.9815 in ACC and 0.0299 in FPR. Our system, which gets scores very close to CLCNN, achieves 0.9890 in TPR, 0.9777 in ACC and 0.0321 in FPR. The model SDCNN in [22] and our system get close ACC scores, but our system achieves a much higher TPR score than CLCNN while it gets an FPR score much better than our system. Interestingly, quite different performances between DBPF and CLCNN are reported even though they have very similar architectures. It may be one of the reasons that the length of vectors for every character in CLCNN is 128, while it is 32 in DBPF. It seems that vectors in high dimensions can retain more information about URLs. Maybe our system will achieve better scores if we increase our vector size from 48 to 128.

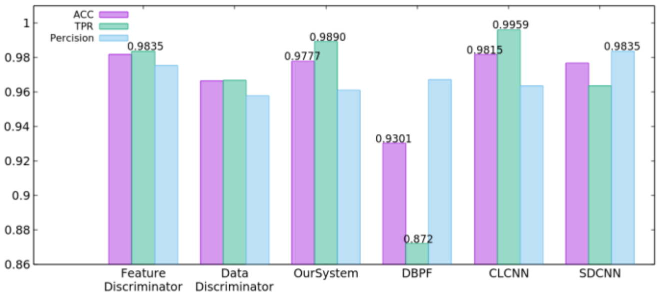

Fig. 10. ACC, TPR, Precision: comparison for different models on mixed dataset

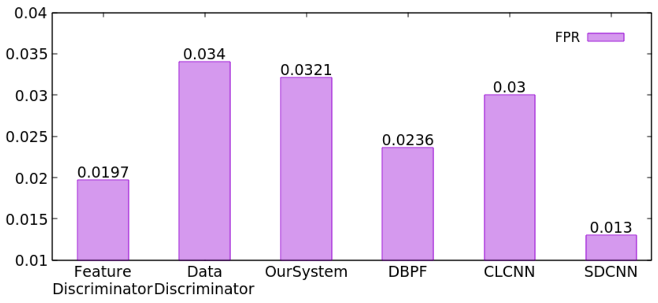

Fig. 11. FPR: comparison for different models on mixed dataset

These comparison results prove that our system can sufficiently represent URLs and detect web attacks on edge devices and is competitive with existing approaches. Second, it is convenient to update our system when required. First, discriminators will be retrained with data newly collected offline. Next, discriminators will be replaced by retrained models in turn. The system will be working all the time during updating because one model is getting updated while the others are still working. For instance, if Feature Discriminator need to be updated, the comprehensive decision will make a judgment according to other discriminators. Significantly, attacks against deep learning models have been proposed in [26, 27]. In fact, a model could be bypassed by adversarial samples well-designed by the attackers. In our system, we use several completely different models; if one model is bypassed, other models will largely recognize it and the administer can update the system according to the adversarial sample's information saved in the logs.

VI. Conclusions and Future work

In this work, we proposed a distributed system for web attack detection from URLs by utilizing deep learning techniques. This system is designed to protect multiple web applications in the distributed environment of the EoT. Several deep models will be trained separately and deployed in different servers. This system will normalize the URLs of edge devices and will respond according to the detection results from concurrent models. The experimental results show the word2vec model's capacity for representing URLs and the system's sufficient capability of detecting web attacks. In addition, this system has the potential to be useful in the real world because of its automatic feature selection, convenience of updating and stability of protecting from attacks against distributed deep models.

Numerous directions for our future work are considered. Primarily, we will try more distributed deep models such as decision trees, LSTM, HMM and etc. in the discrimination. Second, we will optimize the algorithm for comprehensive decisions, which at the present is adding add probabilities from discriminators in varying proportions. Stacking and bagging are most famous techniques in combing deep models. Third, optimizing the methods of feature presenting which include SVD, PCA and Auto-encoder to gain better performance.

References

[1] D. Hendler, S. Kels, and A. Rubin, "Detecting malicious PowerShell commands using deep neural networks", in *Proceedings of Asia Conference on Computer and Communications Security*, ACM, 2018. DOI: 10.1145/3196494.3196511.

[2] J. Saxe, and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *International Conference on Malicious and Unwanted Software (MALWARE)*, IEEE, 2015.

[3] Z. Yuan et al. (2014). Droid-sec: deep learning in android malware detection. ACM SIGCOMM Computer Communication Review. 44(4).

[4] M. Y. Kim, and D. H. Lee. (2014). Data-mining based SQL injection attack detection using internal query trees. Expert Systems with Applications. 41(11), pp: 5416-5430.

[5] M. Du et al, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2017.

[6] T. Rashid, I. Agrafiotis, and J. RC Nurse, "A new take on detecting insider threats: exploring the use of hidden markov models," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, ACM, 2016.

[7] Y. Han et al, "Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017.

[8] L. Zhu, X. Tang, M. Shen, X. Du, M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," IEEE Journal on Selected Areas in Communications, Vol.36, No.3, pp. 628-643.

[9] K. Zhang, S. Leng, X. Peng, L. Pan, S. Maharjan and Y. Zhang, "Artificial Intelligence Inspired Transmission Scheduling in Cognitive Vehicular Communications and Networks", IEEE Internet of Things Journal, Vol. 6, No. 2, pp. 1987-1997, Apr. 2019.

[10] M. Shen, B. Ma, L. Zhu, R. Mijumbi, X. Du, J. Hu, "Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection," IEEE Transactions on Information Forensics and Security, Vol.13, No.4, pp. 940-953.

[11] Y. Xiao, X. Du, J. Zhang, and S. Guizani. (2007, Nov.). Internet Protocol Television (IPTV): the Killer Application for the Next Generation Internet. *IEEE Communications Magazine*. *45(11)*, pp. 126-134.

[12] Z. Tian, S. Su, W. Shi, X. Du, M. Guizani and X. Yu. (2018, Dec.). A Data-driven Method for Future Internet Route Decision Modeling. *Future Generation Computer Systems*. *95*, pp: 212-220. DOI: 10.1016/j.future.2018.12.054

[13] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," in *Proceedings of International Conference on Machine Learning and Computing*, ACM, 2018.

[14] B. Anderson, and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of ACM Workshop on Artificial Intelligence and Security*, ACM, 2016.

[15] Z. Tian, W. Shi, Y. Wang, C. Zhu, X. Du, S. Su, Y. Sun and N. Guizani. (2019). Real Time Lateral Movement Detection based on Evidence Reasoning Network for Edge Computing Environment. *IEEE Transactions on Industrial Informatics*. 2019. DOI: 10.1109/TII.2019.2907754.

[16] X. Du, M. Guizani, Y. Xiao and H. H. Chen, Transactions papers. (2009, Mar.). A Routing-Driven Elliptic Curve Cryptography based Key Management Scheme for Heterogeneous Sensor Networks. *IEEE Transactions on Wireless Communications*. *8(3)*, pp. 1223-1229.

[17] C. Zhang, L. Zhu, C. Xu, X. Liu, K. Sharif, "Reliable and Privacy-Preserving Truth Discovery for Mobile Crowdsensing Systems," IEEE Transactions on Dependable and Secure Computing. (Online, DOI: 10.1109/TDSC.2019.2919517)

[18] X. Du and H. H. Chen. Security in Wireless Sensor Networks. (2008, Aug.). *IEEE Wireless Communications Magazine*. *15(4)*, pp. 60-66.

[19] Y. Xiao, V. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway. (2007, Sep.). A Survey of Key Management Schemes in Wireless Sensor Networks. *Journal of Computer Communications*. *30(11-12,)*, pp. 2314-2341.

[20] J. Liang, W. Zhao, and W. Ye, "Anomaly-Based Web Attack Detection: A Deep Learning Approach," in *VI International Conference on Network, Communication and Computing*, ACM, 2017.

[21] M. Ito, and H. Iyatomi, "Web application firewall using character-level convolutional neural network," in *International Colloquium on Signal Processing & Its Applications (CSPA)*, IEEE, 2018.

[22] M. Zhang, et al, "A deep learning method to detect web attacks using a specially designed cnn," in *International Conference on Neural Information Processing*, Springer, Cham, 2017.

[23] T. G. Carmen, et al. (2015). Combining expert knowledge with automatic feature extraction for reliable web attack detection. *Security and Communication Networks*. *8.16*, pp: 2750-2767.

[24] M. A. Hall. (1999, April). "Correlation-based feature selection for machine learning," PhD Thesis, University of Waikato.

[25] S. O. Uwagbole, W. J. Buchanan, and L. Fan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2017.

[26] M. E. Ahmed, and K. Hyoungshick, "Poster: Adversarial Examples for Classifiers in High-Dimensional Network Data," in *ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2017.

[27] N. Papernot et al, "Practical black-box attacks against machine learning," in *ACM on Asia Conference on Computer and Communications Security*, ACM, 2017.

[28] T. Mikolov et al., "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.

[29] K. He et al., "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016.

[30] A. Joulin et al., "Bag of tricks for efficient text classification," arXiv preprint arXiv:1607.01759, 2016.