IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 31 March 2022, accepted 15 June 2022, date of publication 23 June 2022, date of current version 1 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3185748

An Attack Detection Framework Based on BERT and Deep Learning

YUNUS EMRE SEYYAR[1], ALI GÖKHAN YAVUZ[2], AND HALIL MURAT ÜNVER[1], (Member, IEEE)
[1]Department of Computer Engineering, Graduate School of Natural and Applied Sciences, Kırıkkale University, 71451 Kırıkkale, Turkey
[2]Department of Computer Engineering, Graduate School of Natural and Applied Sciences, Turkish-German University, 34820 Istanbul, Turkey

Corresponding author: Yunus Emre Seyyar (yunusemre.seyyar@gmail.com)

ABSTRACT Deep Learning (DL) and Natural Language Processing (NLP) techniques are improving and enriching with a rapid pace. Furthermore, we witness that the use of web applications is increasing in almost every direction in parallel with the related technologies. Web applications encompass a wide array of use cases utilizing personal, financial, defense, and political information (e.g., wikileaks incident). Indeed, to access and to manipulate such information are among the primary goals of attackers. Thus, vulnerability of the information targeted by adversaries is a vital problem and if such information is captured then the consequences can be devastating, which can, potentially, become national security risks in the extreme cases. In this study, as a remedy to this problem, we propose a novel model that is capable of distinguishing normal HTTP requests and anomalous HTTP requests. Our model employs NLP techniques, Bidirectional Encoder Representations from Transformers (BERT) model, and DL techniques. Our experimental results reveal that the proposed approach achieves a success rate over 99.98% and an F1 score over 98.70% in the classification of anomalous and normal requests. Furthermore, web attack detection time of our model is significantly lower (i.e., 0.4 ms) than the other approaches presented in the literature.

INDEX TERMS Anomalous request, BERT, deep learning, web attack, multilayer perceptron, natural language processing.

I. INTRODUCTION

Internet related technologies have been an integral and indispensable aspect of our lives to the extent that Internet has become as important as other utilities such as water, gas, and electricity. Use of web applications has benefited from the increase in popularity of the Internet. Web applications are utilized via Word Wide Web (WWW). In fact, WWW can be considered as a distributed and massive information system, which is based on the client-server model. Browsers are programs that regulate the relationships between clients and servers. Uniform Resource Locator (URL) textualizes IP (Internet Protocol) addresses the client uses when communicating with the server. Each device on the network has a unique address (i.e., the IP address). Since IP addresses are not easily memorizable, URL textualizes IP addresses that the clients use when communicating with the server. The acronyms used in this paper and their definitions are presented in Table 1.

When a Hypertext Transfer Protocol (HTTP) server is running, it is open to all HTTP requests. To provide access to the server, the HTTP gate (port) is left open in network firewalls. HTTP requests can contain malicious pieces of codes, because they appear to be valid HTTP requests, they are accepted by traditional firewalls and are not thoroughly investigated.

Attackers, generally, target web systems via HTTP protocol [1]. A web server responds with web pages whenever a request is received. Web servers have tasks such as storing, serving, and rendering web pages to clients. Communication between a web server and web pages is facilitated by HTTP [2].

One of the most used protocols in the WWW is the HTTP protocol and its secure extension, the HTTP Secure (HTTPS) protocol. As with many protocols, HTTP and HTTPS protocols also have vulnerabilities. Attackers exploit these vulnerabilities and perform attacks such as Man in the Middle (MITM), brute force, Distributed Denial of Service (DDoS), SQL Injection (SQLI), and Cross Site Scripting (XSS) attacks [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Luca Cassano.

In MITM attack, the traffic between the user and the gateway is routed and one of the intermediate routers acts as the real target while it is not. It sends signals to WiFi network by combining Address Resolution Protocol (ARP) spoofing and Secure Sockets Layer (SSL) stripping techniques. HTTPS is modified by appending the SSL header and HTTP packet of the user computer to the transferred data [4].

In order to consume the resources of the target, adversaries perform a Time-Triggered Protocol (TTP) flooding attack from the bots using the GET or POST methods of the HTTP protocol. Using certain tools, the application source code is accessed and a DDoS attack is performed to interrupt the service [5].

It has been reported that despite the inherent security measures implemented in HTTPS, information such as web page fingerprints, packet sizes, and timing information are leaked by HTTPS. Attackers, on the other hand, perform brute force attacks to predict the passwords of users with special lists prepared using such information, which is a particularly important type of attack especially for weakly protected web sites [3].

Sharing personal information on the Internet via web applications whets the appetite of attackers. These web applications communicate with databases, which store personal information belonging to users using Structured Query Language (SQL). According to Open Web Application Security Project (OWASP) top 10 vulnerabilities in 2021, in order of popularity, SQLI takes the first place, and the second most prominent attack is XSS [6]. SQLI and XSS attacks result in retrieval of users' information from databases or modification of information on web pages, among others. There are various successful classical web attack detection methods. However, there are problematic use cases, which cannot be satisfactorily addressed by traditional and rule-based approaches.For example, it is possible to create many different SQLI attacks and countering every conceivable attack would require too many rules to be constructed, which is a prohibitively cumbersome undertaking. Although, many SQLI attacks can be countered by using relatively simple measures still there are SQLI attacks, which are not easy to thwart by employing rule-based counter measures only. Recently, Machine Learning (ML) and Deep Learning (DL) techniques in use cases that rely on recognizing patterns have proven themselves as better alternatives for particularly challenging attack detection scenarios. Since malicious attacks are intrinsically repetitive and involve codes that revolve around similar patterns, DL approaches are highly successful in recognizing these patterns [3]–[9].

In current ML and DL based attack detection approaches, word embeddings such as autoencoder and word2vec are being utilized. It is shown that using word embeddings improves success rate in malicious attack detection tasks when compared to the traditional ML approaches such as rule-based models [7], [8]. Nevertheless, recent developments regarding transformers (e.g., Bidirectional Encoder Representations from Transformers (BERT), Robustly Optimized Bert Approach (RoBERTa)) and their remarkable success in text classification tasks make them promising candidates to be used in challenging attack detection use cases.

TABLE 1. Table of acroynms.

Acronyms	Description
ARP	Address Resolution Protocol
AUC	Area Under Curve
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
CSIC	Spanish Research National Council
DL	Deep Learning
DDoS	Distributed Denial of Service
HTTP	Hypertext Transfer Protocol
LSTM	Long Short-Term Memory
ML	Machine Learning
MITM	Man in the Middle
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NSP	Next Sentence Prediction
OWASP	Open Web Application Security Project
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SSL	Secure Sockets Layer
SQL	Structured Query Language
SQLI	SQL Injection
TTP	Time-Triggered Protocol
URL	Uniform Resource Locator
WAF	Web Application Firewall
WWW	Word Wide Web
XSS	Cross Site Scripting

Recent studies [1], [9]–[11] show how important web security is for the security and privacy of users' personal information. Although there is a wealth of studies in this area, we identified that there are still gaps to be filled. Considering the OWASP reports, we were inspired to work on SQLI and XSS, which are at the top of the list and are among the most dangerous for users.

While SQLI attacks are server-side vulnerabilities targeting web application databases, XSS attacks are client-side vulnerabilities targeting web application users. Since one of the aforementioned attacks (XSS) targets clients and the other (SQLI) targets servers, it is possible to state that it would be beneficial for both sides to classify HTTP requests, which include SQLI and XSS attacks, as normal and anomalous, respectively.

In this paper, we propose a novel approach that detect web attacks. Note that we specifically consider web requests that are not encrypted (e.g., end user requests behind a firewall). We use state-of-the-art methodologies in DL and Neural Language Processing (NLP). We used the BERT model to obtain the vectors corresponding to the words in the word vector space. Then using these word vectors, we have trained several Multilayer Perceptron (MLP) models. Our architecture receives URLs in HTTP requests as inputs. We tokenize these URLs using the BERT tokenizer. Afterwards these tokens passed through the pre-trained BERT model. Resulting word vectors are then used to train our MLP model. To the best of our knowledge, this is the first example of BERT and MLP models are used together in order to detect web attacks.

Our contributions in this study are listed below;

- By incorporating NLP into our framework, we show that high performance classification, in terms of delay and accuracy, can be achieved without data normalization.
- We demonstrate that the BERT model, which is utilized, successfully, in NLP, can also be employed, successfully, for SQL language, which qualifies as a synthetic language.
- While previous studies used a 1:1 ratio for normal and anomalous request selections to determine model accuracy, we used ratios of 1:1, 1:10, and 1:20 for normal and anomalous request selections to determine the accuracy of our model, which is better representation of real world scenarios.
- Our performance evaluations reveal that requests can be rapidly tested as to whether they are anomalous or normal requests in as low as 0.4 ms per request, therefore, low latency provided by our approach is one of its defining and superior features.

The rest of the article is organized as follows. In Section II, we present a review of prominent studies in the literature on detecting web attacks. In Section III, we provide a detailed description of our proposed model. In section IV, performance evaluations of our model are provided. Conclusions of this study are drawn in Section V.

II. RELATED WORKS

In this section, we present a systematic overview of representative studies in the literature, which are most related to our study. First, we briefly review ML based attack detection approaches by using synthetic datasets in Section II-A. Second, we provide a concise overview of ML word embedding based attack detection approaches utilising real-life datasets in Section II-B. Third, we present word embedding based attack detection approaches in Section II-C. Fourth, we summarize studies employing BERT based techniques in Section II-D. Finally, we express the differences of this study from the literature in Section II-E.

A. ML BASED WEB ATTACK DETECTION USING SYNTHETIC DATASETS

Komiya *et al.* [7] proposed a ML method for the classification of malicious web codes. The proposed method utilizes Naive-Bayes, Support Vector Machines, and k nearest neighbor approaches. Accuracy of the proposed method by using synthetic datasests is reported to be over 98%. Shar *et al.* [12] utilized a synthetic dataset that they produced using Php-MinerII. WEKA is used for feature extraction, and the MLP model is used for classification. Performance evaluations reveal that accuracy obtained in the study is 80%. Fidalgo *et al.* [13] worked on the detection of a set of prominent SQLI attacks. The proposed model runs on PHP slices, which until recently was known as the most popular server-based language. They employed a dataset created with Software Assurance Reference Database (SARD). They used Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) in conjunction with Dropout layers models. It is reported that an accuracy of 95% is obtained by using the Rmsprop optimizer. Shadid *et al.* proposed a hybrid deep learning model with cookie analysis engine for web attacks and attacker profiling. They created a large dataset and trained CNN-based deep learning models. The accuracy of the model was reported to be 99.94% [17].

B. ML BASED WEB ATTACK DETECTION USING REAL-LIFE DATASETS

Hoang [8] investigated attack detection on web applications, which can access networks with vulnerabilities. It is advocated that existing methods work for static websites, therefore, a ML based model that works on both static and dynamic websites is proposed. The proposed method utilizes Naive Bayes and J48 Decision Tree methods. Performance evaluations with 300 anomalous websites from the Goldrake dataset and 100 normal websites from world universities' websites, shows accuracy over 93% and over 99%, respectively. Liang *et al.* [15] proposed a model that does not require feature selection unlike the studies overviewed in the preceding paragraph. The system design is based on two RNN models that use normal requests to detect attacks (i.e., one of the RNN models is a supervised learning model while the other one is an unsupervised learning model). The authors trained normal patterns of requests by using Web Application Firewall (WAF) and Spanish Research National Council (CSIC) datasets. The accuracy of this approach was found to be over 98%. Tian *et al.* [20] studied the detection of malicious web shells. Word2Vec was used for feature extraction in malicious web acceptance detection, and CNN was used for classification. A dataset consisting of 3,691 malicious web shell instances and 3,990 normal instances was used. It is reported that it is possible to achieve an F1-score over 98%. Gong *et al.* [14] proposed a model with the observation that annotation errors can be misleading in model training. They worked on model uncertainty that could help find annotation errors, as well as the misclassification caused by these errors. They used CSIC, Apache-2006, and Apache-2017 datasets. The F1-score of the model using CNN reached over 98% success. Yu *et al.* studied the detection of malicious requests to web servers. They used SVM in the last layer of their TextCNN text classification model. They used the CSIC 2010 dataset to train the model and achieved over 99% accuracy [18].

C. WORD EMBEDDING BASED WEB ATTACK DETECTION APPROACHES

Yu *et al.* [9] proposed a method against SQLI attacks, which pose a major security threat, extracted word vectors using Word2Vec on SQL queries, and classified them by using SVM, which has a detection time of 0.89 ms. Mac *et al.* [1] proposed a model for detecting malicious

TABLE 2. Related studies on web attack detection.

Study	Dataset type	Classifier	Performance metrics
Classification of Malicious Web Code by Machine Learning [7]	Synthetic	SVM, Naive-Bayes, k-Nearest	Accuracy>98%
A Website Defacement Detection Method Based on Machine Learning Techniques [8]	Real websites	Naive-Bayes, C4.5	Accuracy>97% F1-score>99%
Predicting Common Web Application Vurnerabilities from Input Validataion and Sanitization Code Patterns [12]	Synthetic	Nive Bayes, C4.5, MLP	Accuracy>83% Accuracy>88% Accuracy>91
Towards a Deep Learning Model for Vulnerability Detection on Web Application Variants [13]	Synthetic	LSTM	Accuracy>93%
Model Uncertainty Based Annotation Error Fixing for Web Attack Detection [14]	CSIC 2010, Apache-2006, Apache-2017	CNN	Accuracy>95% F1-score>87%
Anomaly-Based Web Attack Detection: A Deep Learning Approach [15]	CSIC 2010, WAF logs	LSTM	Accuracy>98%
A Distributed Deep Learning System for Web Attack Detection on Edge Devices [11]	CSIC 2010, FWAF, HttpParams	M-ResNet	Accuracy>99%
A novel architecture for web-based attack detection using convolutional neural network [16]	CSIC 2010v	CNN	Accuracy>96% F1-score>96%

patterns in HTTP/HTTPS requests. They used the CSIC 2010 dataset. The system model incorporates an autoencoder for feature extraction. Furthermore, ModSecurity is integrated with an autoencoder. Performance evaluations revealed that an F1-score of 94% is achieved, which has a detection time of 5.1 ms. Tian *et al.* [11] proposed a model on the detection of web attacks targeting cloud data centers, which facilitate fairly high volume of data transfers, exacerbated with the development of the Internet of objects. This team utilized two DL models working simultaneously. They used M-ResNet with Word2Vec for feature extraction and CNN for classification. They used CSIC 2010, FWAF, and HttpParams datasets. They achieved an accuracy of 99% by using the CSIC 2010 dataset. Tekerek [16] investigated web-based attacks. He used bag of words and CNN in his work. CSIC 2010 was used as the dataset. Accuracy and F1-score values achieved in the study is higher than 96%. Chen *et al.* proposed an SQLI detection system using Word2Vec, one of the NLP methods that does not rely on a background rule, and they used CNN for classification. They performed their tests with 4000 normal and 4000 SQLI samples. Accuracy and F1-score of the system is reported to be over 98% [19]. A set of studies on web attack detection with data sets, classifiers, and success metrics are shown in Table 2.

D. BERT BASED TECHNIQUES

Devlin *et al.* [20] designed a language representation model called BERT to pre-train left and right-sided presentments by co-conditioning unlabeled data, unlike traditional language models. This model achieved an F1-score over 83%. Farahani *et al.* [21] investigated BERT's performance in NLP for Persian due to its success and growing popularity in English. By training the BERT model for Persian language, they showed that it outperformed previous studies at operations such as text classification and sentiment analysis. Martin *et al.* [22] hypothesised that pre-trained models are insufficient for NLP tasks in French and conducted a feasibility study on this subject. They worked on large and small datasets obtained from the web and created a BERT model. They stated that the model they trained, achieved the best performance level to date. Antun *et al.* [23] studied the performance of the BERT model for Arabic, since Arabic is a morphologically rich language, and showed that the pre-trained BERT model performed well. Cui *et al.* [24] applied word masking in Chinese text due to the success of BERT in various NLP tasks. They demonstrated that the accuracy achieved by this model in assorted natural language operations such as emotion classification and sentence pair matching is high. Rojas [25] stressed that one of the strategies of the spam filter for deception is the choice of a synonym for the message or similar words. He experimented with decision tree, kNN, SVM, logistic regression, naive Bayes, and MLP with BERT. He achieved accuracy values over 96%. Wong *et al.* [26] investigated MITM attacks due to the increasing importance of IoT devices. They built a BERT-based model and tested various traditional machine learning methods. They showed that MLP gave better results. Li *et al.* [27] used BERT for replacing vulnerable words with similar ones according to semantics and grammar, which achieved accuracy values over 97.8%.

E. DIFFERENCE OF THIS STUDY FROM THE LITERATURE

As briefly outlined in the previous paragraphs there is a rich literature on web attack detection. However, attack

TABLE 3. Examples of normal and anomalous requests in CSIC 2010, FWAF, and httpParams datasets.

Dataset	Type	Sample
FWAF	Normal	http://localhost:8080/tienda1/index.jsp
	Anomalous	http://localhost:8080/tienda1/publico/pagar.jsp?modo=insertar&precioA=51&B1=Confirmar
CSIC 2010	Normal	/root/public/code/cp_html2txt.php?page=http://192.168.202.118:8080/moclyxlwqyfjnp?
	Anomalous	/examples/jsp/num/index.php?id='union/**/select/**/0,0,1369222205,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0–
HttpParams	Normal	"cauifield@tuviaje.com.af","24","norm","norm"
	Anomalous	"1'));select (case when (8601=1220) then 8601 else 8601*(select 8601 from mysql.db) end)#","88","sqli","anom"

detection time is as important as the accuracy of the proposed approaches. We hypothesized that word placement models can be utilised in attack detection to improve detection time without sacrificing the detection accuracy. As such, in this study, we employed BERT to process HTTP/HTTPS requests for attack detection. Note that BERT has been successfully utilised in various languages (e.g., English, French, Persian, Arabic languages) and NLP processes, therefore, we proposed that SQL can also be considered as a synthetic language, which can be processed by BERT. In this context we considered each HTTP request as a word and proposed a hybrid model that uses the BERT model for these word vectors and the MLP model for classification. To the best of our knowledge, BERT and MLP models are used together for the first time to detect web attacks in the study. Our results show the success of the proposed model in synthetic languages in terms of both processing time and accuracy.

III. MATERIALS AND METHODS

The system architecture we propose for detecting web attacks is based on the integration of the BERT model and the MLP model. The BERT model is utilised to transform the words into vectors. For the classification of HTTP requests into normal and anomalous sets the MLP model is employed. CSIC 2010 [28], FWAF [29], and httpParams [30] datasets constitute the inputs of the system. System performance is characterized by using the accuracy and F1-score metrics, which are widely utilised in literature. In the rest of this section, first, we introduced the datasets so that our ultimate goal in classification can be comprehended fairly easily. Second, we present the DL part of the architecture. Third, the evaluation metrics used in this study are elaborated. Fourth, the overall system architecture is described in detail.

A. DATASETS

Selection of the datasets to be used in classifying web attacks is of utmost importance. In fact, the main performance metric of this study is the correct classification of normal and anomalous requests. In this study we opt to utilize HttpParams, CSIC 2010, and FWAF datasets, which consist entirely of web attack patterns and have been employed in many studies in the literature, therefore, we can compare the performance of our approaches against many other solutions. The first dataset we utilize is the CSIC 2010 dataset, which includes 36,000 normal requests and more than 25,000 anomalous requests. The dataset was automatically generated, and it targeted an e-Commerce web application's traffic. This dataset contains different attack types (e.g., SQLI, XSS, etc.). The second dataset is generated from HTTP traffic recorded by the Web Attacks Firewall (WAF). In order to obtain a larger dataset without duplicate elements, URLs from different domains are combined from names, source paths, and attribute keys through domain merging. It consists of 1,290,000 normal requests and 48,000 anomalous requests. The third dataset is the HttpParams Dataset on GitHub. This dataset was produced with different tools, and it contains 19,304 normal requests, and 11,763 anomalous requests. All the datasets are shown in Table 4

TABLE 4. Details of datasets.

Dataset Types	Total	Requests Types	
		Normal Requests	Anomalous Requests
FWAF	1,338,000	1,290,000	48,000
CSIC 2010	71,000	36.000	25.000
HttpParams	31,067	19,304	11,763

URLs consist of several mandatory and optional parts such as protocol, domain name, top-level domain, folder, file name, and file extension. A URL created from these parts is considered a normal request. However, a URL created by injecting code from SQL or script languages (e.g., SELECT, UNION, ALERT) are considered anomalous requests. Examples of normal and anomalous requests extracted from the datasets we used in our study are shown in Table 3.

B. DEEP LEARNING

DL can be defined as a class of ML techniques that uses a plurality of nonlinear hidden layers for feature extraction, transformation, pattern analysis, and classification [31]. DL based solution approaches have found widespread applications in many domains such as industrial applications, medical informatics, robotics, computer vision, predictive maintenance, finance, text processing, and classification problems in many domains. DL methods have given very successful results in processing many data types such as video, audio, and text [32]. DL includes computational models with multiple layers of processing so that the available data can be represented at multiple levels of abstraction [32]. Constituents of the utilized deep neural network that we used can be listed as: ***perceptron***,

activation function*, *cost function*, *and fully connected layers, which are explained in the rest of this subsection.

1) PERCEPTRON

Deep neural networks are based on the operating logic of the human brain. Indeed, deep neural networks are computer programs that imitate biological neural networks. The atomic building blocks of these programs are known as perceptrons [33].

The perceptron has input and output layers. All input values are multiplied by their weights, their sums are taken, and a bias value is added to this sum. The result is given to an activation function, the resulting error is input to an optimization function, and consequently the weights are updated to minimize error.

2) ACTIVATION FUNCTION

The activation function outputs a value in response to an incoming input value. It is important that the derivative of the activation function can be easily calculated because the derivative of the activation function is used in backpropagation, therefore, it is imperative to choose a function whose derivative is easily calculated so that the calculation does not slow down [34].

3) LOSS FUNCTION

Training of a neural network is achieved by updating the weights. The update mechanism is actually a feedback based error minimisation operation. The error is the differences between the actual value and the prediction. The loss function's goal is to bring this error closer to zero. Optimization functions are used to minimize the error, if possible close to zero.

4) FULLY CONNECTED LAYER

In the fully connected layer all neurons in the relevant layer are connected to all the neurons in the previous layers. In classification problems, the number of neurons in the last layer is equal to the number of classes.

a: MLP

MLP type multilayer artificial neural networks consist of input, hidden, and output layers. The aggregate input to the neurons of each layer is obtained by summing the weighted neuron outputs in a lower layer. Neuron outputs are obtained depending on the activation function defined for a particular neuron. In an MLP network each node is fully connected to every node in the preceding and proceeding layers. Supervised learning is typically utilised in MLPs. The architecture of MLP in its simplest form is presented in Figure 1.

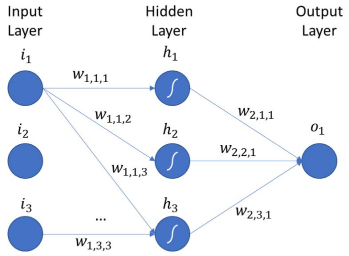


FIGURE 1. **MLP architecture.**

b: EVALUATION METRICS

There are several metrics used to evaluate the performance of the proposed model in terms of classification accuracy. The metrics we utilise in performance evaluations are: Accuracy, Precision, Recall, F1-score, which are computed by using TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) values. In the study we employ the F1-score and accuracy metrics. However, in order to obtain the F1-score metric, the other metrics need to be calculated. Accuracy is defined as the ratio of correct predictions to the total number of predictions as given in Eq. (1)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}. \tag{1}$$

Precision is a metric to express how many of the values predicted as positive are actually positive as presented in Eq. 2

$$Precision = \frac{TP}{TP + FP}. \tag{2}$$

Recall is used to quantify how much of the values that should be predicted as positive are predicted as positive. We would like to reiterate that True Positive Rate is another term used for recall. They both are utilised to measure how accurately the model predicts true positive values as presented in Eq. 3.

$$Recall = \frac{TP}{TP + FN}. \tag{3}$$

True Negative Rate is a measure of how accurately the model predicts true negative values as presented in Eq. 4

$$TNR = \frac{TP}{TP + FP}. \tag{4}$$

False Positive Rate is the ratio of those predicted to be 1 even though the value is 0 as given Eq. 5.

$$FPR = \frac{FP}{FP + TN}. \tag{5}$$

False Negative Rate is defined as the ratio of those predicted to be 0 even though the true value is 1, which is provided in Eq. 6.

$$FNR = \frac{FN}{FN + TP}. \tag{6}$$

The ROC (Receiver Operating Characteristic) is used to generate a Precision / TNR report as given in Eq. 7.

$$ROC = \frac{Precision}{TNR}. \tag{7}$$

The F1-score actually is the harmonic mean of the Precision and Recall values as given in Eq. (8)

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{8}$$

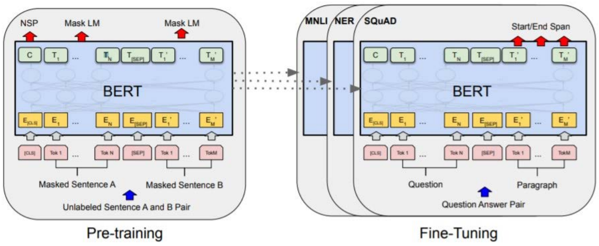
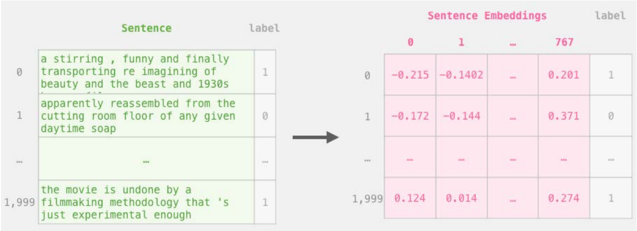


FIGURE 2. Overall pre-training and fine-tuning procedures for BERT.



	Sentence	label
0	a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s	1
1	apparently reassembled from the cutting room floor of any given daytime soap	0
…	…	…
1,999	the movie is undone by a filmmaking methodology that 's just experimental enough	1

	Sentence Embeddings				label
	0	1	…	767	
0	-0.215	-0.1402	…	0.201	1
1	-0.172	-0.144	…	0.371	0
…	…	…	…	…	…
1,999	0.124	0.014	…	0.274	1

FIGURE 3. BERT embedding. The sentences shown on the left panel are represented by the vector of weights assigned by BERT tokenizer on the right panel.

C. BERT

A typical URL has a standard structure unless it is modified by adversaries. In literature, to analyse URLs their standard structure is exploited by treating them as regular sentences. As such, analysis of URLs (e.g., word prediction, word classification) can be performed by utilising the rich repertoire of NLP techniques. Word embedding methods are among the most effective NLP approaches. One such method is BERT, which is reported to more successful than the other word embedding methods. BERT is an open-source, pre-trained NLP model developed by Google AI group researchers [20]. Note that, both autoencoders and word embeddings are utilized to efficiently represent the data through projections onto appropriate vector spaces. Indeed, like autoencoders, word embedding models learn a vector space embedding for some data.

By adding a suitable output layer to a pre-trained BERT model, significantly better results can be obtained in language processing tasks when compared to classical NLP methods [35]. BERT uses two basic learning strategies to overcome contextual constraints and facilitates a bidirectional association. Masked Language Modeling (MLM) is performed before the word strings are transferred to the BERT model. Note that, 15% of the word strings are replaced with the [MASK] token. In this manner, MLM attempts to predict the original value of masked words based on the context formed by other unmasked words in the sequence.

In Next Sentence Prediction (NSP) sentence pairs are taken as inputs to the model in the BERT training process. The objective is to train the model in such a way that the model can predict whether the second sentence in the pair is the next sentence in the document. In the training of the model, the second sentence in the original document is chosen for 50% of the inputs, and in the other 50%, the second sentence is randomly selected. In fact, a successfully trained model can determine that the sentence chosen randomly is not related to the first sentence. BERT architecture, which includes pre-training and fine-tuning procedures, is presented in Figure 2 [17].

BERT can be considered as a stack of encoders and decoders. However, in the traditional encoder-decoder architecture, some learned characteristics especially in the relatively distant history are forgotten as the input gets longer. As RNN evaluates incoming words sequentially, it preserves the integrity of words. However, as the input becomes longer, the relationship established between far separated words can diminished significantly. With the Attention mechanism, there have been significant developments in the field of NLP, such as the Transformer architecture and Google's BERT. Attention, the problem of decreasing the value of the leading words in RNN can be alleviated because the encoder transmits to decoder the hidden state information generated after processing each word. Transformers, which evaluate the incoming data left-to-right/right-to-left with parallel processing and multi-head attention mechanisms, have better performance in comparison to traditional encoder-decoder architectures. The main difference of BERT when compared to RNN, Attention, and Transformers is its double-sided examination of the text, which results in its improved handling of the relationship with the words on the right and left of the word under processing, and learning the content with MLM and NSP. The transformer network used by BERT consists of encoders and decoders that include self-attention mechanisms and feed forward networks. WordPiece coding [36], which is an encoder architecture, creates word vectors by comparing the positions of a word in sentences.

A BERT tokenizer splits the sentence into tokens, which is illustrated in Figure 3. To solve the classification problem, tokens are inserted into the beginning (i.e., [CLS] token) and end (i.e., [SEP] token) of a sentence. The maximum length setting used in text processing also applies here. If the sentence is shorter than the maximum length, then the empty fields are filled with zeros. However, if it is longer than the maximum length then the excess part is removed. Upon completion of this task, a sentence is divided into tokens, and eventually the tokens are indexed. Hidden attention corresponding to only the first token is taken for simplicity. Then, for each word, the vector distance in the word space is determined [37].

D. SYSTEM ARCHITECTURE

There are many successful examples of classification approaches based on feature vectors obtained from text inputs in the literature. Inspired by the success of such studies, in our framework we also utilised a similar approach (i.e., features obtained by BERT tokenizer from text inputs are classified by using MLP).

The first step in the automated analysis of text data is to transform the text into a representation that can conveniently

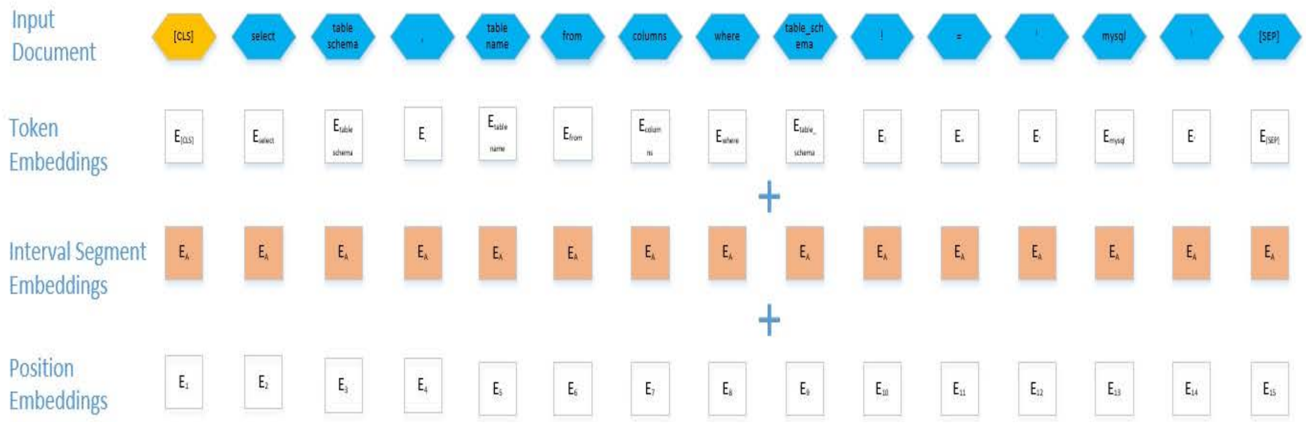


FIGURE 4. Tokenization of URLs. BERT symbol embeddings (i.e., token embeddings) and positions of words in sentences are utilized to express (positional) information and sentence pairings as input fields in the task. A distinction is made between the first and second sentences. A unique placement learning section (segment) contains additional embeddings for representation and input.

TABLE 5. Examples of 80 × 768 matrix with word vector weights representation after BERT Tokenizer process. Each row corresponds to a word and the maximum number of words in queries is limited by eighty.

	0	**. . .**	**767**
0	-0.3988637030124664	. . .	-0.420408129692
.	.	. . . .	.
.	.	. . . .	.
79	-0.5857216119766235	. . .	-0.376996338367462

be interpreted by a computer, which typically is accomplished by resorting to NLP techniques. Note that, in the last decade significant improvements in NLP techniques have been achieved due to the groundbreaking advances in DL techniques.

Since attackers typically try to modify the URLs to perform web attacks, it is, indeed, of utmost importance to preserve each word and character in URL strings. Therefore, to provide the integrity of URLs, in our framework, the words or sentences (i.e., each of the tokenized URLs is considered as a sentence) obtained from URLs are converted to numerical values, which are then merged to form word vectors in the word vector space.

As such our system architecture is built upon a text classification infrastructure. Therefore, as in many text classification schemes, tokenization of the text via BERT tokenizer is the primary step in our framework, which is illustrated in Figure 4.

The BERT tokenizer takes a set of words with a predetermined maximum length parameter, which is chosen as 80 in our framework. In fact, we empirically determined that using 80 as the maximum length parameter is sufficient for the vast majority of our URL queries. As stated earlier, to guarantee the uniformity of the set sizes, shorter sentences are padded with zeros. By processing the input URLs the BERT tokenizer outputs a feature metrics with a dimension of 80 × 768. An example 80 × 768 matrix with vector weights is presented in Table 5.

Each of the row vectors in the input matrix of size 80 × 768 corresponds to the specific words in the input query except for the first (i.e., [CLS] token) and the last (i.e., [SEP] token) rows. The first row is the classifier vector for the entire sentence (i.e., URL). However, probability values calculated by BERT are not sufficient to determine whether a URL is normal or anomalous with high performance. As a remedy we decided to incorporate a feed forward neural network (i.e., MLP) to our architecture, which accept the BERT output as its input. Hence, the row vectors (each of length 768) are utilised to train the MLP model for the classification of URLs. MLP architecture that we utilized consists of 6 fully connected layers (i.e., linear layers), each followed by batch normalization and Rectified Linear Unit (ReLU) layers. The last layer of the MLP architecture is a Softmax [38] layer that outputs predictions for a given URL query. While linear layers determine the weights and features, other layers are mainly used to regularize the outputs. The overall system architecture is illustrated in Figure 5. Unlike the other results in the literature, data pre-processing (other than BERT tokenizer) is not needed in our framework, which results in lower processing times. Indeed, BERT has proven to be highly effective in various NLP tasks such as information extraction, sentiment analysis and question answering [39].

Note that we experimented with other learning architectures other than MLP. For example, we built CNN-based models with various depths and obtained and average accuracy of up to 0.9579 accuracy. Nevertheless, the MLP-based model that we created give the best performance when compared to other learning models.

IV. RESULTS AND DISCUSSION

Computational experiments to explore the performance of our proposed framework is conducted by utilising a computing system, which includes two NVIDIA RTX 2080 TI GPUs, 64 GB of system RAM, and 16 cores of an AMD

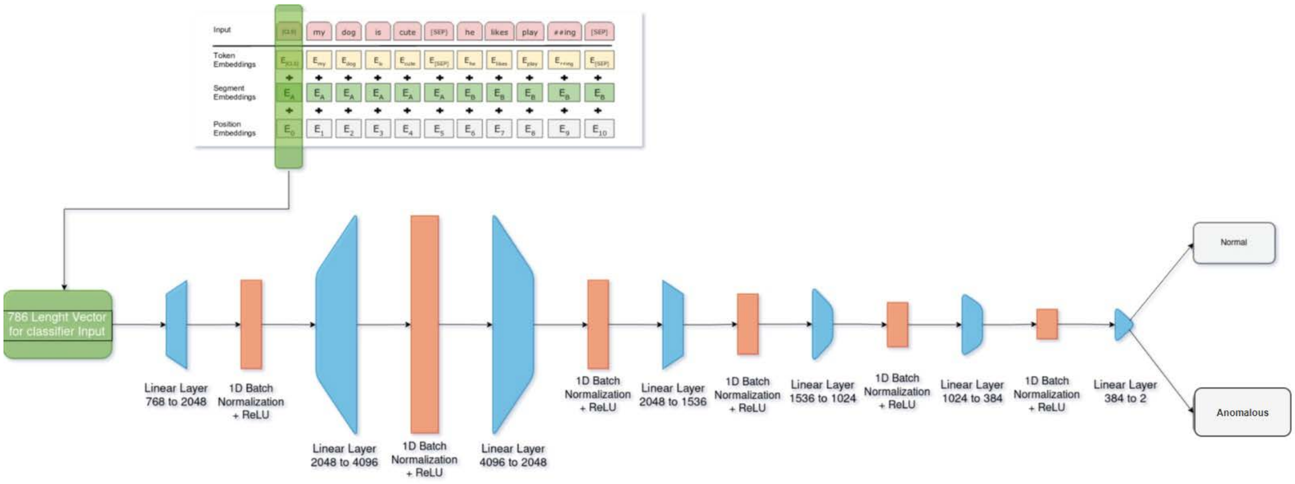


FIGURE 5. Proposed system architecture combining BERT tokenizer and six fully connected layers MLP model.

processor. The software employed in this study includes Python, PyTorch, and BERT library.

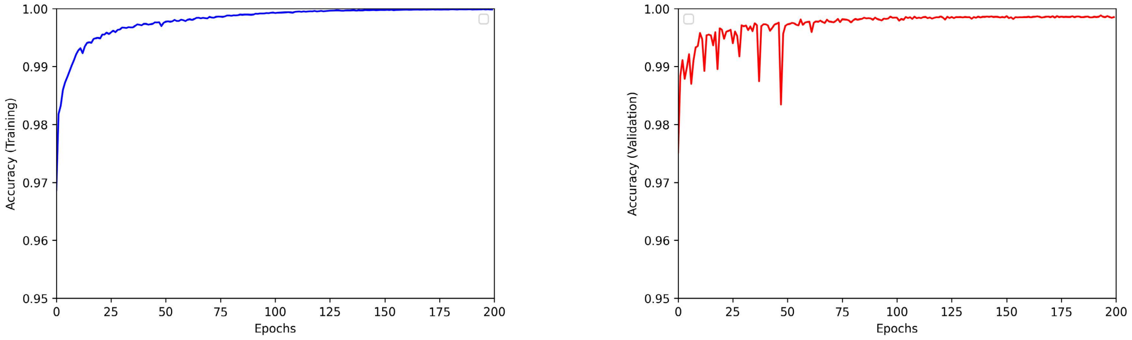
Training phase is comprised of epochs, each of which takes four minutes on the average. The most time consuming operation within each epoch is the processing of URLs by the BERT tokenizer. Instead of processing queries one by one, we first processed all queries together and save resulting vectors. To obtain the best MLP architecture we empirically explored the design space (e.g., number of layers, neurons, and loss functions). After experimenting with a large number of MLP architectures, all trained for 200 epochs, the best performing architecture is determined, which is elaborated in subsection III-D. Note that the selected model is then further fine-tuned until improvements in accuracy between epochs are reduced below 0.0001. Although, we continued the training process up until 350 epochs we observed that the training process almost always reach a plateau much earlier.

While various datasets for this problem is available (as explained in subsection III-A), we initially opted to utilise the CSIC 2010 dataset, which consists of 50,000 samples in seven different categories (i.e., normal queries and six different anamolus queries such as SQLI, XSS, CMDI, etc.,). Since our objective is to create a system for detection of anomalous queries, we re-labeled entries in this dataset as normal and anomalous samples (i.e., all six anamolous query categories are merged to form a single anamolous category). Our initial MLP architecture was a simpler artificial neural network with three hidden layers, which results in 99% accuracy when it is tested by data extracted from the CSIC 2010 dataset. However, when it was tested on another dataset (i.e., httpParams dataset) its performance in terms of accuracy was slightly better than random predictions. It was then hypothesized that samples between datasets were too dissimilar to each other to be considered comparable. In order to test this hypothesis, the same model was trained on the httpParams dataset and then tested with the CSIC 2010 dataset, which resulted in a similar outcome (i.e., slightly better accuracy then random predictions). Therefore, we decided to merge all three available datasets (i.e., CSIC 2010, FWAF, and HttpParams), which was to be used as our main and only dataset. To enable the uniformity of our unified dataset the eliminate attributes of queries which were not present in all of them. Nevertheless, performance evaluations of our architecture is done by using the unified dataset.

As explained in subsection III-B4.b, we utilised accuracy and F1-score as our performance metrics. We divided our dataset into training (85% of the data) and validation (15% of the data) parts.

Accuracy as a function of epochs for training and validation phases are presented in Figure 6a and Figure 6b, respectively. Accuracy in training phase surpass 98.00% after seventh epoch and it stays above 99.90% after ninety first epoch. After one hundred sixty second epoch accuracy is above 99.98%, which determines the convergence level of our framework. Accuracy in validation phase exceed 98.30% after forty seventh epoch and it never falls below 99.70% after forty ninth epoch. Beginning with one hundred and third epoch accuracy is always higher than 99.80%.

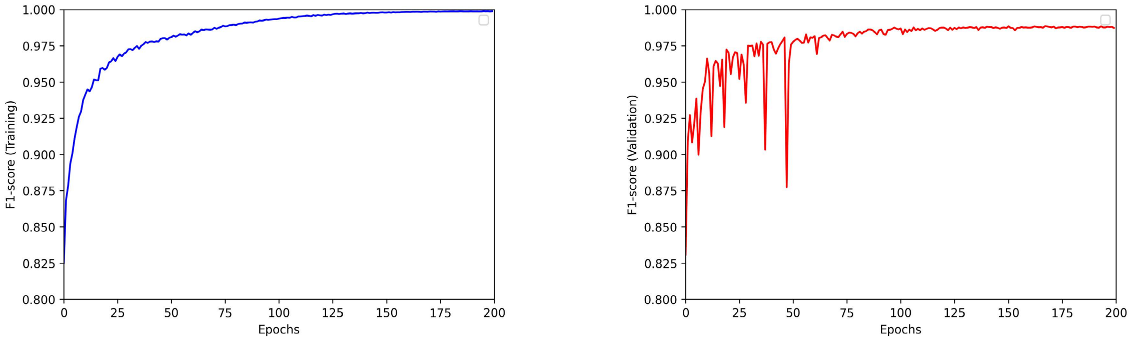
In Figure 7a and Figure 7b, F1-score as a function of epochs for training and validation phases are given, respectively. Beginning with the twenty first epoch F1-score values exceed 96.00% in training phase and after the eighty first epoch F1-score values are always higher than 99.80%. F1-score values settle around 99.80% once the number of epochs surpass one hundred fifty. In validation phase, F1-score values exceed 88.00% and 97.00% after forty seventh epoch and forty ninth epoch, respectively. F1-score converges to 98.70% after one hundred and seventy second epoch. Note that the relatively large variations in F1-score values are primarily due to the weighted loss used in F1-score computations.



(a) Accuracy (training)

(b) Accuracy (validation)

FIGURE 6. Accuracy as a function of epochs for training and validation. In training phase accuracy surpasses 98.00% after seventy epochs and never drops below 99.98% after one hundred sixty two epochs. Likewise, in validation phase forty seven epochs and one hundred and three epochs are the thresholds for 98.30% and 99.80% accuracies, respectively.



(a) F1-score (training)

(b) F1-score (validation)

FIGURE 7. F1-score as a function of epochs for training and validations. In training phase F1-score surpasses 96.00% after twenty epochs and never drops below 99.80% after one hundred fifty epochs. Likewise, in validation phase forty seven epochs and one hundred and seventy epochs are the thresholds for 88.00% and 98.70% F1-score, respectively.

TABLE 6. Table of k-cross validation for 10 fold.

	Train Acc.	Val. Acc.	Train F1-Score	Val. F1-Score
fold1	99.83%	99.79%	98.51%	98.21%
fold2	99.85%	99.74%	98.67%	97.59%
fold3	99.83%	99.73%	98.56%	97.66%
fold4	99.84%	99.80%	98.60%	97.15%
fold5	99.84%	99.78%	98.65%	98.16%
fold6	99.83%	99.76%	98.59%	98.98%
fold7	99.84%	99.72%	98.55%	98.56%
fold8	99.84%	99.78%	98.59%	98.15%
fold9	99.83%	99.77%	98.51%	98.09%
fold10	99.84%	99.66%	98.48%	98.15%
Average	99,84%	99,75%	98,57%	98,07%

This table is prepared with the 10-cross validation technique, and it is found that the training and validation accuracies for each fold, as well as the F1-scores for the training and validation. Finally, we take the highest values for each fold and calculate its average.

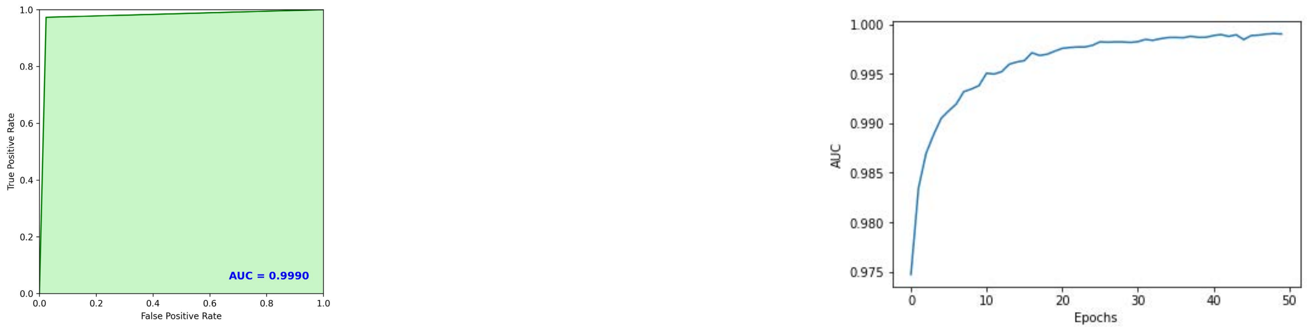
We present a 10-fold cross validation analysis in Table 6. For each folds, average of the highest values are obtained as 99.84%, 99.75%, 98.57%, 98.07% training accuracy, validation accuracy, training F1-score, and validation F1-score, respectively. Furthermore, AUC (Area Under Curve) ROC curve for 10-fold cross validation is presented in Figure 8a, whereas, the change in AUC for 50 epochs is plotted in Figure 8b. Our evaluations reveal that there is no robustness issue with our solution.

TABLE 7. Attack detection times.

Studies	Detection Time (ms)
Web attack detecting via autoencoder [1]	5.1
Text analysis based SQLI attack detecetion [9]	0.89
Proposed Model	0.4

In our framework the BERT tokenizer is utilised to create feature vectors from query sentences, which are than processed by the classifier. We first experimented with 1D convolutional model with a linear output for classification. Although initial tests with this approach reached a certain accuracy level, it was not enough to provide a satisfactory solution to our problem. Later we experimented with an MLP classifier and fine-tuned the MLP model (as explained in subsection III-D) to find the best compromise between speed and precision. As presented in section IV our architecture achieves 99.98% accuracy and 98.70% F1-score on validation. Furthermore, apart from the BERT feature extraction, it takes 0.4 ms to complete the entire classification operation, which is less than half of the delay values reported in the literature [1], [9], which is presented in Table 7.

Accuracy reached by our solution is similar to the accuracy values reported in [7], [8], [11], [15]. However, unlike the aforementioned studies, we report computation time values in



(a) AUC ROC curve (10-fold)

(b) AUC vs. epochs (k-fold)

(a): This graph shows AUC value, which is 0.9990,
(b): This graph shows the change in AUC values in 50 epochs.

FIGURE 8. **K-fold analysis: (a) AUC ROC curve and (b) AUC as a function of epochs.**

our manuscript, which is one of the contributions of our solution. Indeed, our computation times are significantly lower than the results reported in the literature [1], [9]. Furthermore, our solution does not require pre-processing unlike the solutions in [7], [8], [11], [15]. Moreover, present studies in the literature utilize datasets with approximately equal numbers of normal and anomalous requests, which does not reflect the actual occurrence frequency of anomalous request in real life (i.e., in real life the occurrence frequency of anomalous requests is much lower than the frequency of normal requests). Therefore, for the better representation of the real life we worked with datasets of which 50%, 10%, and 5% are anomalous requests and we show that the performance obtained with all three cases are the same.

In summary, this is the first study in the literature which shows that BERT can be successfully utilised for web attack detection with high accuracy. Moreover, incorporation of NLP to our framework results in high performance classification without the need for data normalization, which leads to extremely low delay.

V. CONCLUSION

Adversaries exploit web request queries to take advantage of vulnerabilities of web applications. In this study, we proposed a novel approach based on BERT and DL techniques for the detection of web attacks. We utilized BERT model for the representation of URLs and MLP classifier to discriminate the normal and anomalous queries. Experimental evaluations show that the representational capability of URLs by BERT model is high, which in turn leverages high performance web attack detection significantly. The novel contributions of this study are itemized as follows:

- We utilised an aggregate dataset by merging three different datasets (CSIC 2010, FWAF, httpParams), which is used in both training and validation phases. Therefore, our framework is capable of generalizing data from multiple datasets successfully. To the best of our knowledge, these datasets were utilised separately in other studies. Nevertheless, our framework is the first one that integrate all three datasets within the context of web attack detection through query classification.
- Data normalization is a process utilised in other studies on web attack detection, which increases the translational load. However, our framework does not require data normalization, yet our performance results are par with reported results on web attack detection in the literature. Nonetheless, our processing time is lower then the other studies.
- Performance evaluations reveal that our framework is capable of representing URLs successfully and results in web attack detection with an F1-score and an accuracy and as high as over 97%, 99%, respectively. As such our framework can be utilised in practical real world applications and scenarios.

In its current form our proposed model runs on a Linux platform. However, our model can also be modified to run on Android, IoS, and Mac platforms, which is among our future research agenda.

REFERENCES

[1] H. Mac, D. Truong, L. Nguyen, H. Nguyen, H. A. Tran, and D. Tran, "Detecting attacks on web applications using autoencoder," in *Proc. 9th Int. Symp. Inf. Commun. Technol. (SoICT)*, 2018, pp. 416–421.

[2] I. Kresna A. and Y. Rosmansyah, "Web server farm design using personal computer (PC) desktop," in *Proc. 10th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE)*, Jul. 2018, pp. 106–111.

[3] J. Luxemburk, K. Hynek, and T. Čejka, "Detection of HTTPS brute-force attacks with packet-level feature set," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 0114–0122.

[4] A. R. Chordiya, S. Majumder, and A. Y. Javaid, "Man-in-the-middle (MITM) attack based hijacking of HTTP traffic using open source tools," in *Proc. IEEE Int. Conf. Electro/Inf. Technol. (EIT)*, May 2018, pp. 0438–0443.

[5] S. Bishnoi, S. Mohanty, and B. Sahoo, "A deep learning-based methodology in fog environment for DDOS attack detection," in *Proc. 5th Int. Conf. Comput. Methodol. Commun. (ICCMC)*, Apr. 2021, pp. 201–206.

[6] D. Wichers, "Owasp top-10 2013," OWASP Foundation, Annapolis, MD, USA, Tech. Rep., Feb. 2013.

[7] R. Komiya, I. Paik, and M. Hisada, "Classification of malicious web code by machine learning," in *Proc. 3rd Int. Conf. Awareness Sci. Technol. (iCAST)*, Sep. 2011, pp. 406–411.

[8] X. D. Hoang, "A website defacement detection method based on machine learning," in *Proc. Int. Conf. Eng. Res. Appl.* Da Nang, Vietnam: Springer, 2018, pp. 116–124.

[9] L. Yu, S. Luo, and L. Pan, "Detecting SQL injection attacks based on text analysis," in *Proc. 3rd Int. Conf. Comput. Eng., Inf. Sci. Appl. Technol. (ICCIA)*, 2019, pp. 95–101.

[10] Y. Tian, J. Wang, Z. Zhou, and S. Zhou, "CNN-Webshell: Malicious web shell detection with convolutional neural network," in *Proc. VI Int. Conf. Netw., Commun. Comput.*, 2017, pp. 75–79.

[11] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1963–1971, Mar. 2020.

[12] L. K. Shar and H. B. K. Tan, "Predicting common web application vulnerabilities from input validation and sanitization code patterns," in *Proc. 27th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2012, pp. 310–313.

[13] A. Fidalgo, I. Medeiros, P. Antunes, and N. Neves, "Towards a deep learning model for vulnerability detection on WEB application variants," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Oct. 2020, pp. 465–476.

[14] X. Gong, J. Lu, Y. Zhou, H. Qiu, and R. He, "Model uncertainty based annotation error fixing for web attack detection," *J. Signal Process. Syst.*, vol. 93, pp. 187–199, Feb. 2020.

[15] J. Liang, W. Zhao, and W. Ye, "Anomaly-based web attack detection: A deep learning approach," in *Proc. VI Int. Conf. Netw., Commun. Comput.*, 2017, pp. 80–85.

[16] A. Tekerek, "A novel architecture for web-based attack detection using convolutional neural network," *Comput. Secur.*, vol. 100, Jan. 2021, Art. no. 102096.

[17] W. B. Shahid, B. Aslam, H. Abbas, S. B. Khalid, and H. Afzal, "An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling," *J. Netw. Comput. Appl.*, vol. 198, Feb. 2022, Art. no. 103270.

[18] L. Yu, L. Chen, J. Dong, M. Li, L. Liu, B. Zhao, and C. Zhang, "Detecting malicious web requests using an enhanced TextCNN," in *Proc. IEEE 44th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2020, pp. 768–777.

[19] D. Chen, Q. Yan, C. Wu, and J. Zhao, "SQL injection attack detection and prevention techniques using deep learning," in *Proc. J. Phys., Conf.*, 2021, vol. 1757, no. 1, Art. no. 012055.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[21] M. Farahani, M. Gharachorloo, M. Farahani, and M. Manthouri, "Pars-BERT: Transformer-based model for Persian language understanding," 2020, *arXiv:2005.12515*.

[22] L. Martin, B. Müller, P. Javier Ortiz Suárez, Y. Dupont, L. Romary, É. Villemonte de la Clergerie, D. Seddah, and B. Sagot, "CamemBERT: A tasty French language model," 2019, *arXiv:1911.03894*.

[23] W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based model for Arabic language understanding," 2020, *arXiv:2003.00104*.

[24] Y. Cui, W. Che, T. Liu, B. Qin, and Z. Yang, "Pre-training with whole word masking for Chinese BERT," 2019, *arXiv:1906.08101*.

[25] S. Rojas-Galeano, "Using BERT encoding to tackle the mad-lib attack in SMS spam detection," 2021, *arXiv:2107.06400*.

[26] H. Wong and T. Luo, "Man-in-the-middle attacks on MQTT-based IoT using BERT based adversarial message generation," in *Proc. KDD*, 2020, pp. 1–6.

[27] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "BERT-ATTACK: Adversarial attack against BERT using BERT," 2020, *arXiv:2004.09984*.

[28] C. T. Giménez, A. P. Villegas, and G. A. Marañón, "HTTP data set CSIC 2010," Inf. Secur. Inst. CSIC (Spanish Res. Nat. Council), Serrano, Madrid, Tech. Rep., 2010.

[29] F. Ahmad. (2017). *Web Application Firewall*. [Online]. Available: https://github.com/faizann24/Fwaf-Machine-Learning-driven-Web-Application-Firewall

[30] Morzeux. (2020). *Httpparams*. [Online]. Available: https://github.com/Morzeux/HttpParamsDataset

[31] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7 nos. 3–4, pp. 197–387, 2013.

[32] Y. Bengio, I. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2016.

[33] S. K. Vasudevan, S. R. Pulari, and S. Vasudevan, *Deep Learning: A Comprehensive Guide*. Boca Raton, FL, USA: CRC Press, 2022.

[34] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," in *Bio-Inspired Neurocomputing*. Singapore: Springer, 2021, pp. 203–224.

[35] K. Schachinger, "A complete guide to the Google rankbrain algorithm," *Search Engine J.*, 2020.

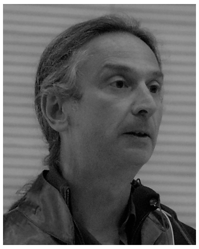
[36] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, and M. Norouzi, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.

[37] J. Alammar. *A Visual Guide to Using Bert for the First Time*. Accessed: Jun. 28, 2022. [Online]. Available: https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

[38] G. Pang, L. Cao, and C. Aggarwal, "Deep learning for anomaly detection: Challenges, methods, and opportunities," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 1127–1130.

[39] A. Roy and S. Pan, "Incorporating medical knowledge in BERT for clinical relation extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 5357–5366.

YUNUS EMRE SEYYAR was born in Kırıkkale, Turkey. He received the Graduate degree from the Computer Engineering Department, Erciyes University, Kayseri, in 2008, and the M.Sc. degree in computer engineering from Kırıkkale University, Kirikkale, where he is currently pursuing the Ph.D. degree in computer engineering. He is also employed as a Scientific Program Expert by The Scientific and Technological Research Council of Turkey (TUBITAK). His responsibilities at TUBITAK include coordination of the advisory committee, management of research and development projects on cyber security and networks, and contributing to research funding vision and policies of TUBITAK. His research interests include cyber security, deep learning, and natural languages processing.



ALI GÖKHAN YAVUZ received the Ph.D. degree in computer engineering from Yıldız Technical University, Istanbul, Turkey. He is currently a Professor and the Head of the Department of Computer Engineering, Turkish-German University. His current research interests include systems and network security, cloud computing, and big data.

HALIL MURAT ÜNVER (Member, IEEE) received the Ph.D. degree in machine engineering from Kırıkkale University, Kırıkkale, Turkey. He is currently an Associate Professor with the Department of Computer Engineering, Kırıkkale University. His current research interests include robotics, computer networks, and network security.