Deep Learning in NLP for Anomalous HTTP Requests Detection

Manh Tien Anh Nguyen*‡, Van Tong*, Sondes Bannour Souihi§, and Sami Souihi†
*School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam
†LISSI-TincNET Research Team University Paris-Est Creteil, France
§ Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
‡TexPECT, France

***Abstract*—Techniques for Deep Learning (DL) and Natural Language Processing (NLP) are rapidly advancing. In addition, we notice that the access and utilisation of web applications is expanding in almost all fields in conjunction with related technologies. Web applications include a wide range of use cases involving personal, financial, military, and political data. This renders web-based applications a desirable target for cyber-attacks. To address this problem, we propose, in this study, a novel model capable of differentiating normal HTTP requests from different types of anomalous HTTP requests. Our model combines NLP techniques, the Bidirectional Encoder Representations from Transformers (BERT) model, and DL techniques. The pre-trained BERT model is able to operate on unprocessed data and therefore does not require manually extracted features. Our experimental results show that the proposed method achieves an F1 score of more than 98.90% in the classification of multiple categories of anomalous requests and normal requests on CAPEC dataset. Furthermore, we leverage Transfer Learning in order to detect new types of anomalous requests or new attack patterns that are similar to training anomalous patterns. With Transfer Learning techniques, our proposed model achieves an F1-score of 61.50% on unseen types of anomalous HTTP requests.**

***Index Terms*—Web attack detection, Deep Learning, Natural Language Processing (NLP), Anomalous HTTP request**

I. Introduction

Web applications play a significant role in people's daily lives, mainly because individuals are migrating their applications and confidential data to the cloud. The prevalence of web applications and the large quantities of sensitive user data they store make them vulnerable to attacks [1]. Therefore, it is essential to protect web applications from intrusions. The majority of vulnerabilities come from HTTP request-related flaws that can be exploited by transmitting specially crafted requests. These requests are intended to circumvent the Web Application Firewall (WAF) and get unauthorized access to the sensitive data. There are hundreds of web attack categories. According to [2], injection attacks are the third most critical threats to web applications security. In addition, the rapid development of web-based applications has resulted in the emergence of an expanding number of new attack types deriving from well-known attacks. Therefore, web attack detection systems have now to be able not only to detect known attack patterns with high accuracy, but also to adapt effectively to new attack patterns.

Given the prevalence of this threat, the cybersecurity community has responded to web application attacks in a variety of ways [4]. Generally, there are two approaches to detect attacks mentioned above. The first is the signature-based method, which detects malicious patterns by searching for a known identity. ModSecurity [5] is one of the most widely used engine for implementing the signature-based method. This platform provides a rule configuration language called 'SecRules' for real-time monitoring, logging, and filtering of HTTP requests according to user-defined rules. However, these rules can still be breached because the majority of signature-based methods rely on regular expression-based filters constructed from known attack signatures and require extensive configuration by security experts. The second approach is the anomaly-based method consisting in establishing normal request profiles so that abnormal requests can be distinguished from regular ones. The anomaly-based approach can be considered as a pattern classification problem with the objective of distinguishing anomalous from benign patterns. Recently, Machine Learning (ML) and Deep Learning (DL) techniques have proved, in use cases that rely on pattern recognition, to be superior alternatives especially for difficult attack detection scenarios [14]–[17]. Since malicious attacks are inherently repetitive and involve codes with similar patterns, DL approaches are highly effective at recognizing such patterns. However, both approaches suffer from a lack in detecting unseen attack patterns or new types of anomalous HTTP requests related to previous attack patterns. These problems can be overcome by leveraging Transfer Learning or Domain Adaptation strategies [11].

In this paper, we propose a novel method for detecting web attacks mentioned in the CAPEC dataset [3]. Note that we explicitly consider non-encrypted web requests (e.g., requests from users behind a firewall). Concretely, our system serves as a Host-based Intrusion Detection System [8] to control access to a web server. HTTP requests between end-users and the web server are strictly managed by our system which has the ability to detect abnormal requests that can be harmful to the server. We use state-of-the-art methodologies in DL and NLP. We leveraged the SecBERT pre-trained model [6] to obtain the vectors corresponding to the HTTP sequence in the word vector space. SecBERT is a variant of BERT model [9] trained on cybersecurity texts, which makes it efficient at learning cybersecurity knowledge and improving downstream tasks (e.g., Named Entity Recognition, Text Classification, Semantic Understanding) in the cybersecurity domain. Our proposed

978-3-903176-59-1 ©2023 IFIP

framework takes URLs as inputs from HTTP requests. The BERT tokenizer is then used to tokenize these URLs, which are then input into the pre-trained SecBERT model to obtain their corresponding word vectors. Based on resulting word vectors, we have trained a classifier with feed-forward neural networks and used the softmax activation function. Furthermore, we used Transfer Learning or Adversarial Domain Adaptation [12] to detect new anomalous requests or attack patterns that are similar to previously training anomalous patterns. We implement a non-generative approach in Adversarial-based Domain Adaptation strategy where a domain confusion loss produced by the domain discriminator helps learn the domain invariant representations. This strategy encourages the training requests space and the unseen requests space to follow the same distribution. The proposed model is consequently able to detect new anomalous requests with greater accuracy than the baseline method. According to our experiments, the proposed method with Domain Adaptation strategy achieves an F1 score exceeding 98.90% and 61.50% respectively on testing dataset and on new anomalous HTTP requests.

The remainder of this paper is structured as follows. In Section II, we provide a literature review of notable studies on web attack detection and Domain Adaptation techniques. We present a thorough description of our proposed framework in Section III. In Section IV, we evaluate the performance of our model. Finally, Section V draws the conclusions of this study.

II. Related Work

In this section, we provide a systematic review of representative studies that are the most relevant to our research. Firstly, we provide a brief overview of ML/DL-based attack detection techniques. Secondly, we summarize studies employing Domain Adaptation, a specific strategy of Transfer Learning. Finally, we explicit the differences between our proposal and previous research.

Mac et al. [14] proposed a model for identifying malicious HTTP/HTTPS request patterns. They used CSIC 2010 data. The proposed model includes an autoencoder for feature extraction. In addition, ModSecurity is integrated with an autoencoder. Performance evaluations revealed an F1-score of 94%, with a detection time of 5.1 milliseconds. Tian et al. [15] proposed a model for detecting web attacks targeting cloud data centers that have facilitated data transmissions made difficult by the growth of the Internet of Things. The authors used CNN for classification, with M-ResNet and Word2Vec for feature extraction. They conducted their experiments on the CSIC 2010, FWAF, and HttpParams datasets achieving 99% of accuracy on the CSIC 2010 dataset. Tekerek et al. [16] conducted research on web-based attacks and based their work on CNN and bag of words techniques. They achieved more than 96% of Accuracy and F1-score on the CSIC 2010 dataset. Chen et al. [17] proposed an SQLI detection system using Word2Vec, a non-rule-based NLP technique, and CNN for classification. They conducted their experiments using 4000 normal samples and 4000 SQLI samples. The system's accuracy and F1-score are reported to exceed 98%.

According to [18], Domain Adaptation (DA) approaches can be categorized into two different categories, namely, (i) Domain Invariant feature learning approach and (ii) Adversarial-based Domain Adaptation approach. The most prevalent deep DA approach relies on decreasing domain discrepancy in a latent feature space and learning domain invariant feature representations. The key component of these strategies is selecting an appropriate divergence measure to accomplish this. MMD [19] measures the distribution divergence of observed samples using the hypothesis of a two-sample statistical test. In particular, the mean of a smooth function relative to the samples from two domains is contrasted, with a larger mean difference indicating a greater domain disparity. In practice, the alignment component proceeds similarly to a task classifier. MMD can then be computed and minimized between the outputs of the classifiers' layers. CDD [20] is proposed to integrate the class label into MMD in order to achieve class-conditioned distribution alignment. While within-class divergence is lowered by lowering CDD, cross-class divergence is widened. Contrastive adaptation networks (CAN) [20] have been proposed as an alternative to estimate the target domain label using clustering, while reducing CDD, in the event that the label in a target domain is lacking in UDA. Rather than a divergence measure such as MMD, the Adversarial based domain adaptation approach focuses on adaptive learning of a divergence measure. Domain adversarial neural network (DANN) [12] uses the gradient reversal layer as a domain discriminator. In addition, a method for initializing the target model with training from the source domain and then performing adversarial adaptation to produce the target domain-specific classifier has been proposed: Adversarial Discriminative Domain Adaptation (ADDA) [21].

As mentioned succinctly in the preceding paragraph, there is a wealth of literature on web attack detection. However, the majority of proposals are designed for binary tasks (such as normal and anomalous requests detection). With only binary detection, it is challenging to analyze and investigate further attack behaviors. In addition, we hypothesized that HTTP request could be regarded as synthetic information that can leverage a pre-trained BERT-based model to extract contextual information for subsequence tasks. Moreover, to the best of our knowledge, Domain Adaptation's application to web attack detection has not been the subject of any research for detecting new anomalous requests that are similar to training anomalous requests.

III. Proposed methodology

Figure 1 illustrates the proposed system architecture for detecting web attacks. Firstly, the SecBERT [6] pre-trained model is used to transform the HTTP sequences in text form into dense vectors. SecBERT [6] is a BERT model [9] trained on a vast corpus of cybersecurity-related text, and is thus capable of representing text-based cybersecurity information (e.g. HTTP requests) as input for subsequent tasks. Secondly, a Multi-layer Perceptron (MLP) classifier model is trained to classify HTTP requests into normal and various anomalous requests groups. In

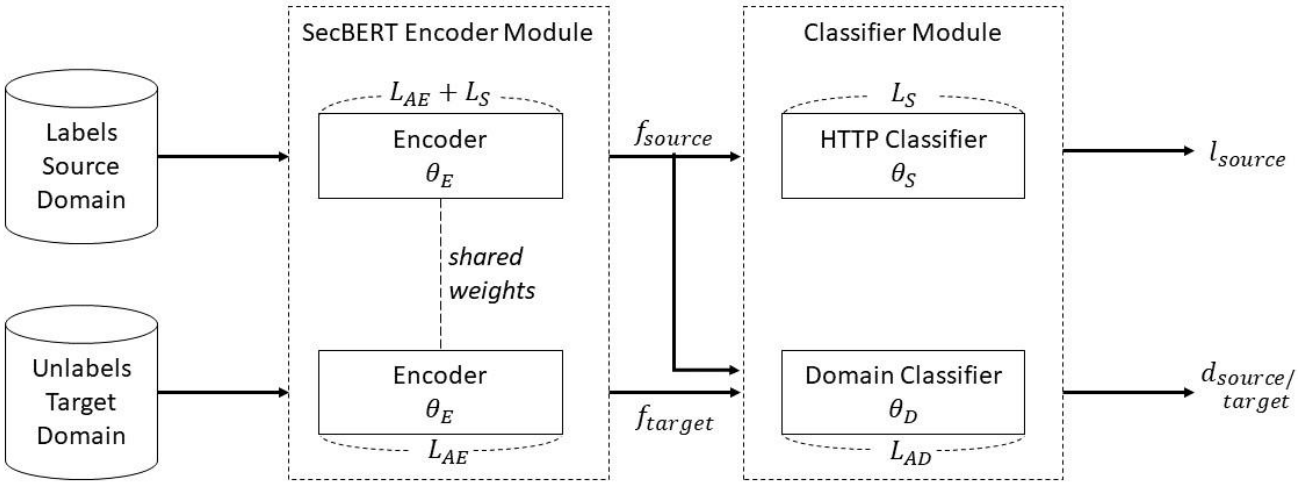

Fig. 1. Proposed system architecture.

addition, Domain Adaptation strategy is used to identify new requests or attack patterns that are comparable to previously trained anomalous patterns. Concretely, we implement a non-generative strategy in Adversarial-based Domain Adaptation in which a domain confusion loss generated by the domain discriminator is used to detect new anomalous requests or new attack patterns that are similar to training anomalous patterns. We classify unknown web attacks into pre-defined categories (e.g., injection) that have the same label as correlating training attack patterns. We trained and evaluated the proposed model on CAPEC dataset. In the remainder of this section, we first present the BERT and SecBERT pre-trained models for our feature extraction and classification task. Next, Adversarial Domain Adaptation strategy [12] is introduced to adapt our proposed model with new types of anomalous HTTP requests. Finally, the overall system architecture is described in detail.

A. BERT and SecBERT pretrained models

The bidirectional encoder representations from transformers (BERT) is a language representation model that was recently introduced by [9]. BERT is used for supporting feature extraction in natural language processing (NLP). NLP is a subfield of computer science concerned with the ability of machines to comprehend written and spoken language in the same manner as humans. In natural language processing (NLP), rule-based human language modeling (computational linguistics) is combined with statistical, machine learning, and deep learning models. The BERT model is employed to produce pre-trained deep bidirectional representations from unclassified text by mutually training on left and right context in all layers. This means that BERT is able to acquire contextual embeddings for words or text form. The pre-trained BERT model may be fine-tuned by adding an output layer to generate a variety of models for a wide range of use cases, such as natural language processing tasks. However, some domains, for example cybersecurity, actually remain highly confidential because of the critical nature of data they deal with, which makes them particularly exposed to cyber threats. As a result, the automatic processing of cybersecurity text is in need of a robust and trustworthy framework. Cybersecurity terms are either unusual in standard English (e.g., web attacks, HTTP requests, and keylogger) or have a variety of implications (homographs) depending on the target domain (e.g., virus, patch, handshake and honeypot) [7]. These differences in the structure of language and semantic contexts complicate text processing and suggest that the standard language model trained on general corpus may not be able to accommodate the vocabulary of cybersecurity texts, resulting in a limited or restricted understanding of the implications of cybersecurity.

SecBERT [6] is a pre-trained cybersecurity language model with a fundamental comprehension of both the character-level, word-level and sentence-level semantics in the cybersecurity field, and therefore effective in automating many crucial cybersecurity tasks which would otherwise require expertise from humans and time-consuming laborious efforts. A vast corpus of cybersecurity-related text has been used to train SecBERT. To make SecBERT effective not only at retaining general English comprehension, but also when implemented to text with cybersecurity consequences, [6] developed a custom tokenizer and a method for modifying pre-trained weights. The SecBERT is evaluated using the standard Masked Language Model (MLM) test in addition to two other standard NLP tasks. The evaluation studies demonstrate that SecBERT outperforms comparable existing models, validating its capacity to solve crucial NLP-related cybersecurity tasks.

Unless modified by adversaries, a typical HTTP request has a standard format. In the literature, URLs are treated as regular sentences in order to analyze their standard structure. As a result, analysis of HTTP requests (e.g., word prediction, word classification) can be carried out using the extensive repertoire of NLP techniques. Word embedding techniques are among

the most efficient NLP methods [10]. In this work, we leverage the capacity of SecBERT pre-trained model to extract features from HTTP requests in order to learn contextual embeddings for HTTP sequences.

B. Adversarial Domain Adaptation (ADA)

The primary objective of ADA is to optimize the performance of both the labeled source domain and the unlabeled target domain. In our work, the training dataset represents the source domain, while the dataset that contains new types of anomalous HTTP requests represents the target domain. Due to the possibility of a domain shift between the source and target domains, this method attempts to reconcile the marginal feature distributions independently of labels for both domains.

Therefore, we train the classification module and the SecBERT encoder module to maximize the probability of source labels given source inputs. In addition, in order to optimize the model's performance in the target domain, we train the SecBERT encoder to confuse a discriminator that attempts to estimate the domain of a data sample and functions as a density model.

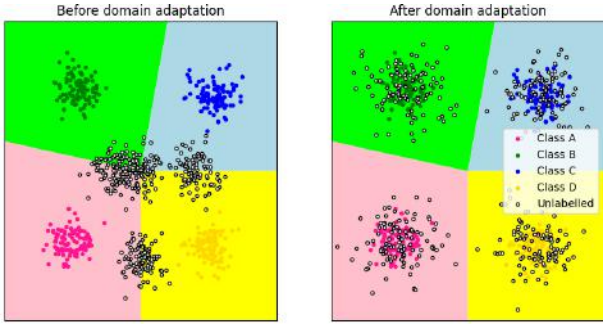

Fig. 2. An example of a domain shift in which the distribution of the unlabeled target domain differs from the labeled source domain distribution despite the fact that both distributions have similar structural characteristics. The supervised task's decision boundary (shown by color) is better aligned across both domains by aligning the two domains.

The marginal feature distributions of the source and target domains are aimed at being aligned during the adversarial training phase. The conditional distributions given the labels are implicitly aligned as a result of the partial structural similarity between the two domains. As a result, as shown in Figure 2, ADA improves the accuracy of a supervised module with decision bounds adapted for the training domains by aligning the distributions.

Figure 1 depicts the division of our proposed architecture into two functional components known as the *SecBERT Encoder Module* and *Classifier Module*. Let $E : \mathbb{R}^n \to \mathbb{R}^m$ represent the encoder that converts a vector embedding f_i from an input HTTP request i_i, which is then used as input for both HTTP classifier phase $S : \mathbb{R}^m \to \mathbb{R}^c$ for determining label l_i and the domain classifier phase $D : \mathbb{R}^m \to \mathbb{R}$ for producing domain label d_i. We can omit the domain classifier while testing since it is essentially an extra module that determines the encoder objective, hence reducing the memory footprint of the model.

The model is optimized to minimize HTTP classifier and adversarial losses, respectively L_S and L_A, simultaneously. The adversarial loss is further subdivided into encoder L_{AE} loss and domain classifier L_{AD} loss (Equations 1 and 3). The goal of the HTTP classifier is to minimize the cross-entropy loss calculated by Equation 2. While both the adversarial encoder and domain classifier losses rely on the encoder parameters w_E and the domain parameters w_D, each loss only gets utilised to the corresponding component to implement the adversarial training procedure. The filters of the HTTP classifier w_S are only optimized with respect to the loss of the HTTP classifier for the data of the source domain. The strength ratio between the HTTP classifier and the adversarial target is determined by the factor λ. In this setting, the encoder is motivated to derive features that strike a balance between the HTTP classifier task and maximizing domain invariance.

The adversarial training approach is realized by applying each loss exclusively to the corresponding module, despite the fact that the adversarial encoder and discriminator losses simultaneously rely on the encoder parameters w_E and the discriminator parameters w_D. Only the HTTPS classifier loss for data from the source domain is taken into account when optimizing the filters of the HTTP classifier w_S. The HTTP classifier and adversarial objectives' respective strengths are determined by the factor λ. This structure allows the encoder to extract features that strike a compromise between the HTTP classifier task's relevance and the maximization of domain invariance.

$$L(w_S, w_D, w_E) = L_S(w_S, w_E) + \lambda L_A(w_D, w_E) \quad (1)$$

$$L_S(w_S, w_E) = \mathbb{E}_{l=S(E(i,w_E),w_S),i\sim S}[-\log(l)] \quad (2)$$

$$L_A(w_D, w_E) = L_{AD}(w_D) + L_{AE}(w_E) \quad (3)$$

$$\begin{aligned} L_{AD}(w_D) = \mathbb{E}_{f=E(i),i\sim S}[-\log(D(f, w_D))] + \\ \mathbb{E}_{f=E(i),i\sim T}[-\log(1 - D(f, w_D))] \end{aligned} \quad (4)$$

This work characterises the adversarial encoder loss L_{AE} independently of the domain classifier loss using two prevalent formulations. Similar to the initial generator loss in the GAN framework [13], the encoder can be learned to maximize domain confusion in ADA:

$$\begin{aligned} L_{AE}(w_E) = -\mathbb{E}_{f=E(i,w_E),i\sim S}[\log(1 - D(f))] - \\ \mathbb{E}_{f=E(i,w_E),i\sim T}[\log(D(f))] \end{aligned} \quad \text{(confusion loss)}$$

In contrast to the GAN framework, this loss is applicable to samples from both domains, not just generated samples. A comparable approach is the minimax formulation, which negates the discriminator loss [12]. Minimax loss can be represented as follows when gradient reversal [12] is utilized:

$$\begin{aligned} L_{AE}(w_E) &= -L_{AD} \\ &= \mathbb{E}_{f=E(i,w_E),i\sim S}[\log(D(f))] + \\ &\mathbb{E}_{f=E(i,w_E),i\sim T}[\log(1 - D(f))] \end{aligned} \quad \text{(minimax loss)}$$

In our work, minimax loss is used in Adversarial Domain Adaptation to improve performance.

C. System Architecture

The architecture proposed for web attack detection using BERT and Transfer Learning is shown in Figure 1. The Encoder Module is based on the SecBERT pre-trained model to capture the method, absolute path and query parameters from the HTTP/HTTPS requests. Both labeled data from source domain and unlabeled data from target domain are fed into the SecBERT encoder module with the same weights for feature extraction. After that, features of source domain and target domain data are transmitted to Classifier Module. There are two classifiers with different objectives in the classification Module: HTTP Classifier and Domain Classifier. All the features of the source data and their corresponding labels are fed into the HTTP Classifier to detect normal requests and multiple anomalous requests. In our work, we incorporated a feed-forward neural network (i.e. MLP) for requests classification task. In contrast, Domain Classifier receives both source features and target features to distinguish the target domain input from the source domain input. It is trained with a gradient reversal layer that multiplies the gradient by a particular negative constant during backpropagation. This is executed in order to create the feature representation domain invariant. Finally, with this training strategy, our model not only recognizes the attack pattern in the source dataset, but also in the target dataset.

IV. Experiments

In this section, we evaluate the proposed method compared to multiple baseline methods including machine learning and deep learning models along with different feature extraction techniques for text data. All our experiments were carried out on a machine running Ubuntu 20.04 LTS having 10 cores, 16 threads Intel(R) Core(TM) i5-13400F CPU at 2.50GHz, 64GB of RAM and NVIDIA RTX 3080 graphics card. Source code of the present work uses Python, Scikit-learn and Pytorch libraries.

A. Data specification

The proposed method is evaluated and benchmarked on the CAPEC dataset [3]. The dataset consists of web requests collected during 12 days in July 2020 by a web server (WordPress) deployed on a virtual machine and accessible via Internet. CAPEC is a multi-label dataset designed specifically for web attack detection, containing 907,814 requests, of which 525,195 are normal requests and 382,619 are anomalous requests, with each record containing 24 distinct features and a set of 13 labels. Table I shows the detailed of typical attributes of normal web request and attack web request on CAPEC dataset.

However, for the purpose of transfer learning phase, CAPEC dataset is converted to a multi-class dataset. We achieve this by removing multi-label class samples. This trick had no significant impact on the dataset beacause these samples represent only 1% of the total dataset. Additionally, similar attack patterns are grouped together and represented by a single label. We identify similar attack patterns by determining the correlation between each attack's behavior (e.g. OS Command Injection and SQL Injection are combined and represented as Injection attacking type). By combining the relevant attacking types, we can generate source and target datasets for evaluation in transfer learning task (e.g. SQL Injection in source domain, OS Command Injection in target domain). Table II provides detailed information about how often a web request is classified under a specific CAPEC heading in both source and target datasets. Note that for the final result, duplicate requests have been eliminated from the original CAPEC dataset. In our experiment, the CAPEC source dataset was used to train all models, while the CAPEC target dataset was used to evaluate the performance of these models on the Transfer Learning task.

B. Evaluation measures

In general, a classifier can be evaluated based on the overall accuracy. However, in multiclass imbalanced data, this metric is no longer applicable, as the dominant classes have much greater impacts than the minor ones. This paper adopts precision, recall, and F1-score metrics to give a better idea of the accuracy achieved within a given class. These measures are calculated using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values.

Precision is a metric that indicates how many of the predicted positive values are actually positive, as shown in Equation 5.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall is used to quantify what proportion of anticipated positive values are actually predicted as positive, as presented in Equation 6.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

As shown in Equation 7, the F1-score is the harmonic mean of the Precision and Recall values.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

C. Results

In this section, we discuss our experimental results for both request classification task and Transfer Learning task.

1) Request classification performance: In table III, we compare our proposed method with other baseline methods, including ModSecurity engine [5] and many ML/DL approaches. ModSecurity engine [5] was the first method we used as a benchmark. The OWASP ModSecurity Core Rule Set (CRS) is a collection of signatures that can defend web applications against a wide range of attacks. The ModSecurity Core Rule Set (CRS) can take action based on the anomaly scores associated with every HTTP request. The action may be passing or declining the request. In our experiment, ModSecurity performs the worst on the CAPEC dataset compared to other methods. The precision, recall and F1-score are respectively 0.8175, 0.7284 and 0.7704. In fact, ModSecurity relies on a predefined core rule set not covering the whole CAPEC dataset, which explains the poor results achieved by the system.

TABLE I
DETAILED OF TYPICAL ATTRIBUTES OF NORMAL WEB REQUEST AND ATTACK WEB REQUEST ON CAPEC DATASET.

Attribute	Normal web request	Attack web request
method_value	GET	POST
http_request_value	/blog/index.php/comments/feed/	/blog/index.php/my-account/edit-profile/post.php/ ?action=edit&post=%7B%7B%20data.id%20%7D%7D
agent_value	Mozilla/5.0 (Windows NT 5.1;...)	Mozilla/5.0 (X11; Linux x86_64,...)
host_value	test-site.com	test-site.com
content_type_value	nan	application/x-www-form-urlencoded
body_value	nan	username=drearySyrup7& password=0C20y& user-registration-login-nonce=6eac0e2d5f...
http_status_code_value	200	200
http_status_message_value	OK	OK
response_content_length_value	424	58110

TABLE II
NUMBER OF WEB REQUESTS IN CAPEC SOURCE DATASET AND CAPEC TARGET DATASET.

Label	CAPEC source		CAPEC target	
	Number of web requests	% of total requests	Number of web requests	% of total requests
Normal	225423	38.37%	1086	7.63%
Injection	261885	44.58%	9168	64.45%
Fake the source of Data	54574	9.29%	1408	9.90%
HTTP splitting	18748	3.19%	920	6.46%
Path traversal	16824	2.86%	938	6.59%
Manipulation	7993	1.36%	318	2.27%
Scanning software	1998	0.35%	384	2.70%
TOTAL	587445		14222	

TABLE III
THE PRECISION, RECALL AND F1-SCORE OF OUR PROPOSED METHOD AND OTHER BASELINE METHODS ON THE MULTICLASS CAPEC DATASET.

Method	Precision	Recall	F1-score
BOW + Naive Bayes	0.8471	0.8323	0.8396
BOW + Decision Tree	0.9632	0.9544	0.9588
BOW + Random Forest	0.9665	0.9602	0.9633
BOW + Logistic Regression	0.9575	0.9493	0.9534
BOW + GRU	0.9698	0.9681	0.9689
BOW + LSTM	0.9702	0.9711	0.9706
TF-IDF + Naive Bayes	0.8771	0.8719	0.8745
TF-IDF + Decision Tree	0.9594	0.9587	0.9590
TF-IDF + Random Forest	0.9693	0.9631	0.9662
TF-IDF + Logistic Regression	0.9554	0.9488	0.9521
TF-IDF + GRU	0.9667	0.9699	0.9683
TF-IDF + LSTM	0.9771	0.9723	0.9747
Word2Vec + Naive Bayes	0.4848	0.4972	0.4909
Word2Vec + Decision Tree	0.9493	0.9401	0.9447
Word2Vec + Random Forest	0.9604	0.9552	0.9578
Word2Vec + Logistic Regression	0.8021	0.7972	0.7996
Word2Vec + GRU	0.9796	0.9705	0.9750
Word2Vec + LSTM	0.9792	0.9748	0.9780
ModSecurity + CRS	0.8175	0.7284	0.7704
SecBERT + MLP + DA (Proposal)	**0.9904**	**0.9878**	**0.9890**

We have also experimented with multiple traditional ML and DL methods on the CAPEC dataset. Since the HTTP request input is a special kind of text form, we make use of NLP techniques for feature extraction. We implement three popular feature extraction methods in NLP: Bag-of-Words (BOW), Term frequency–inverse document frequency (TF-IDF) and Word2Vec algorithms. Feature extraction is performed at the character level instead of the word level to meet the input particularity. For each of the previously mentioned feature extraction techniques, multiple classifiers, including both traditional ML and DL approaches, were integrated to form a comprehensive system. Table III shows results of the preceding techniques. In general, the integration of NLP feature extraction techniques with conventional ML classifiers (e.g. Naive Bayes, Decision Tree,...) yields comparable results. The integration of the TF-IDF feature extractor with the Random Forest classifier gives the most accurate results, with precision, recall, and F1-score values of 0.9693, 0.9631, and 0.9662 respectively. In addition, the use of feature extractors with DL models (e.g. GRU, LSTM) in HTTP request classification task leads to superior results compared to traditional ML classifiers. In our experiments, DL classifiers achieve slightly higher accuracy than ML classifiers. Concretely, the precision, recall, and F1-score are respectively 0.9792, 0.9748, and 0.9780 for the Word2Vec feature extractor with LSTM classifier. More specifically, the DL classifiers outperform the ML classifiers in detecting all categories of attacks, while ML classifiers seem to be biased towards attack types with large training data.

Table IV shows the detailed result of detecting multiple web application security vulnerabilities on the CAPEC dataset using typical ML, DL, rules-based ModSecurity and our proposed method. With a precision, recall, and F1-score of 0.4823, 0.2054, and 0.2881 for the Manipulation attack, the ML classifier having the best accuracy (e.g. TF-IDF extractor with Random Forest classifier) gives limited performance. In contrast, best classifier of DL method (e.g. Word2Vec extractor with LSTM classifier) outperforms the traditional method in detecting Manipulation attack, with precision, recall, and F1-score of 0.8793, 0.4098 and 0.5395 respectively. The poor

TABLE IV
PRECISION(P), RECALL(R), F1-SCORE(F1), FOR EACH ATTACK, OF ML, DL, AND OUR PROPOSED METHOD.

Attacks	Injection			Fake the source of data			HTTP splitting			Path traversal			Manipulation			Scanning software		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BOW + Naive Bayes	0.98	0.95	0.97	0.98	0.92	0.95	0.98	0.98	0.98	0.98	0.90	0.94	0.44	0.16	0.24	0.86	0.96	0.91
BOW + Decision Tree	0.97	0.96	0.97	0.97	0.91	0.94	0.98	0.97	0.98	0.97	0.91	0.94	0.42	0.14	0.22	0.87	0.95	0.91
BOW + Random Forest	0.99	0.96	0.98	0.98	0.91	0.94	0.99	0.98	0.99	0.98	0.92	0.95	0.45	0.20	0.26	0.88	0.98	0.93
BOW + Logistic Regression	0.97	0.95	0.96	0.98	0.92	0.95	0.98	0.96	0.97	0.98	0.98	0.98	0.43	0.17	0.24	0.85	0.95	0.90
BOW + GRU	0.97	0.97	0.97	0.99	0.90	0.94	0.99	0.98	0.99	0.96	0.94	0.95	0.84	0.38	0.52	0.91	0.93	0.92
BOW + LSTM	0.98	0.97	0.98	0.99	0.91	0.95	0.99	0.99	0.99	0.97	0.95	0.96	0.86	0.39	0.53	0.92	0.94	0.93
TF-IDF + Naive Bayes	0.99	0.96	0.98	0.99	0.93	0.96	0.99	0.99	0.99	0.99	0.92	0.96	0.48	0.20	0.28	0.88	0.98	0.93
TF-IDF + Decision Tree	0.98	0.95	0.97	0.98	0.92	0.95	0.98	0.98	0.98	0.97	0.92	0.95	0.46	0.17	0.27	0.87	0.97	0.92
TF-IDF + Random Forest	0.99	0.95	0.97	0.98	0.93	0.96	0.99	0.99	0.99	0.98	0.93	0.95	0.45	0.21	0.26	0.89	0.98	0.93
TF-IDF + Logistic Regression	0.96	0.96	0.96	0.98	0.91	0.94	0.98	0.95	0.96	0.98	0.97	0.98	0.42	0.18	0.24	0.85	0.95	0.90
TF-IDF + GRU	0.96	0.96	0.96	0.99	0.93	0.96	0.98	0.98	0.98	0.97	0.93	0.95	0.85	0.34	0.53	0.91	0.92	0.92
TF-IDF + LSTM	0.97	0.98	0.98	0.99	0.90	0.94	0.99	0.99	0.99	0.96	0.95	0.96	0.85	0.38	0.52	0.92	0.92	0.92
Word2Vec + Naive Bayes	0.98	0.96	0.97	0.98	0.92	0.95	0.99	0.98	0.99	0.98	0.91	0.95	0.44	0.16	0.24	0.86	0.96	0.91
Word2Vec + Decision Tree	0.98	0.97	0.98	0.97	0.91	0.94	0.98	0.96	0.97	0.97	0.90	0.93	0.42	0.15	0.22	0.87	0.95	0.91
Word2Vec + Random Forest	0.99	0.96	0.98	0.98	0.94	0.96	0.99	0.99	0.99	0.97	0.92	0.94	0.45	0.20	0.25	0.89	0.97	0.92
Word2Vec + Logistic Regression	0.98	0.95	0.97	0.98	0.93	0.96	0.98	0.96	0.97	0.98	0.97	0.98	0.43	0.18	0.23	0.85	0.95	0.90
Word2Vec + GRU	0.99	0.97	0.98	0.99	0.91	0.95	0.99	0.99	0.99	0.96	0.96	0.96	0.85	0.39	0.53	0.92	0.93	0.93
Word2Vec + LSTM	0.99	0.98	0.99	1.00	0.93	0.96	1.00	1.00	1.00	0.98	0.96	0.97	0.87	0.40	0.55	0.94	0.96	0.95
ModSecurity + CRS	0.83	0.76	0.79	0.82	0.74	0.77	0.81	0.73	0.76	0.80	0.71	0.75	0.62	0.51	0.56	0.78	0.72	0.74
SecBERT + MLP + DA (Proposal)	**0.99**	**0.99**	**0.99**	**1.00**	**0.95**	**0.98**	**1.00**	**1.00**	**1.00**	**0.99**	**0.98**	**0.99**	**0.96**	**0.89**	**0.95**	**0.98**	**0.97**	**0.98**

performance of the ML classifier can be explained by the fact that the Manipulation attack covers only 1.4% of the training data set. In fact, ML classifiers are biased towards attack categories having the largest number of training samples, whereas DL models are optimized over the entire data set, including attack types with few samples. Finally, our proposed model achieves the best results, with precision, recall, and F1-score values of 0.9904, 0.9878, and 0.9890, respectively. This result demonstrates that using SecBERT pre-trained model for extracting features of HTTP requests is extremely efficient. SecBERT is capable of capturing the specificities of each HTTP request, providing an optimal input for the MLP classifier. A reminder that we evaluated with other learning architectures other than MLP. For instance, we constructed CNN-based models of different configurations and obtained an average F1-score of 0.9887. Nevertheless, when compared to other classifiers, the MLP-based model that we developed offers the best accuracy.

2) Transfer Learning performance: The transfer learning performance of different methods is presented in Table V. As a baseline, we evaluated the performance of naive zero-shot transfer of multiple ML and DL methods. Each method is trained until convergence using only source domain data, and then zero-shot transferred to the target domain. While these models typically obtain high performance on data from the source domain, their zero-shot performance on the target domain can be considerably limited. For transfer from the CAPEC source domain to the CAPEC target domain, naive zero-shot can result in more than 60% of decrease in F1 score when compared to in-domain supervised training. Concretely, ML methods such as TF-IDF feature extractor with the Random Forest classifier have an F1-score of only 0.3697 on the CAPEC target domain, whereas the F1-score on the CAPEC source is 0.9662. DL approaches such as Word2Vec with LSTM classifier also witnessed a significant decrease, with an F1-score of 0.9780 and 0.3778 respectively on the CAPEC source and target domains. In our experiment, ModSecurity performs the Transfer Learning task with the lowest accuracy compared to other methods with an F1-score of only 0.2531 on the CAPEC target domain. This low result can be explained by the fact that ModSecurity detects web attacks using rules, therefore allowing new types of attacks to easily circumvent this system.

These experiments above demonstrates that models without Domain Adaptation techniques and rules-based method have a limited accuracy in detecting new attack patterns that are correlated with attack requests in the training dataset. Therefore, by leveraging Adversarial-based Domain Adaptation strategy, our proposed method significantly outperforms the naive zero-shot baselines in all the transfer tasks without degrading accuracy on the source domain. Our model achieves an F1-score of 0.6150 on the CAPEC target domain and keeps an F1-score of 0.9890 on the CAPEC source domain. Our method improved best result in baseline methods, represented by an F1-score of 0.4124 (TF-IDF feature extractor with Naive Bayes classifier), by around 49%.

TABLE V
THE F1-SCORE OF OUR PROPOSED METHOD AND OTHER BASELINE METHODS ON THE TRANSFER LEARNING TASK.

Method	CAPEC source → CAPEC target
BOW + Naive Bayes	0.4013
BOW + Decision Tree	0.3945
BOW + Random Forest	0.3563
BOW + Logistic Regression	0.3774
BOW + GRU	0.3891
BOW + LSTM	0.3812
TF-IDF + Naive Bayes	0.4124
TF-IDF + Decision Tree	0.3871
TF-IDF + Random Forest	0.3697
TF-IDF + Logistic Regression	0.3733
TF-IDF + GRU	0.3896
TF-IDF + LSTM	0.3983
Word2Vec + Naive Bayes	0.2965
Word2Vec + Decision Tree	0.3555
Word2Vec + Random Forest	0.3378
Word2Vec + Logistic Regression	0.2874
Word2Vec + GRU	0.3880
Word2Vec + LSTM	0.3778
ModSecurity	0.2531
SecBERT + MLP + DA (Proposal)	**0.6150**

V. CONCLUSION

Web application vulnerabilities are exploited by adversaries through web requests. In this study, we proposed a novel approach for the detection of web attacks based on BERT model and DL techniques. SecBERT is a variant of BERT model trained on cybersecurity texts, making it effective at learning and representing cybersecurity knowledge. We used the SecBERT pre-trained model to have an appropriate representation of URLs and the MLP classifier to distinguish between normal and anomalous requests of different types. To go further, our system leverages Transfer Learning to detect unseen abnormal requests or new attack patterns that correlate with training requests. Our approach significantly outperforms the baseline models on both web attack classification and Transfer Learning tasks with an F1 score of 98.90% and 61.50% respectively. This promising result demonstrates the effectiveness of both feature extraction using a SecBERT pre-trained model and the Adversarial-based Domain Adaptation strategy. In future work, we will investigate different Domain Adaptation strategies and more complex classifiers to further improve the overall system accuracy. Additionally, we will have an emphasis on deploying and assessing the system in production.

REFERENCES

[1] Abdulsalam, Y. S., and Hedabou, M. (2021). Security and privacy in cloud computing: technical review. Future Internet, 14(1), 11.

[2] Top OWASP. 10. Application Security Risks-2021. Open Web Application Security Project (OWASP).

[3] Riera, T. S., Higuera, J. R. B., Higuera, J. B., Herraiz, J. J. M., and Montalvo, J. A. S. (2022). A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques. Computers and Security, 120, 102788.

[4] Salih, A., Zeebaree, S. T., Ameen, S., Alkhyyat, A., and Shukur, H. M. (2021, February). A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection. In 2021 7th International Engineering Conference "Research and Innovation amid Global Pandemic"(IEC) (pp. 61-66). IEEE.

[5] ModSecurity. https://github.com/SpiderLabs/ModSecurity (accessed April. 11, 2023).

[6] "SecBERT: pretrained BERT model for cyber security text, learned Cybersecurity Knowledge. https://github.com/jackaduma/SecBERT (accessed April. 12, 2023).

[7] Aghaei, E., Niu, X., Shadid, W., and Al-Shaer, E. (2022, October). SecureBERT: A Domain-Specific Language Model for Cybersecurity. In International Conference on Security and Privacy in Communication Systems (pp. 39-56). Cham: Springer Nature Switzerland.

[8] Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T., and Bahaj, S. A. (2022). Deep learning for intrusion detection and security of Internet of things (IoT): current analysis, challenges, and possible solutions. Security and Communication Networks, 2022.

[9] Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[10] Naseem, U., Razzak, I., Khan, S. K., and Prasad, M. (2021). A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. Transactions on Asian and Low-Resource Language Information Processing, 20(5), 1-35.

[11] Wilson, G., and Cook, D. J. (2020). A survey of unsupervised deep domain adaptation. ACM Transactions on Intelligent Systems and Technology (TIST), 11(5), 1-46.

[12] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., ... and Lempitsky, V. (2016). Domain-adversarial training of neural networks. The journal of machine learning research, 17(1), 2096-2030.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[14] Mac, H., Truong, D., Nguyen, L., Nguyen, H., Tran, H. A., and Tran, D. (2018, December). Detecting attacks on web applications using autoencoder. In Proceedings of the 9th International Symposium on Information and Communication Technology (pp. 416-421).

[15] Tian, Z., Luo, C., Qiu, J., Du, X., and Guizani, M. (2019). A distributed deep learning system for web attack detection on edge devices. IEEE Transactions on Industrial Informatics, 16(3), 1963-1971.

[16] Tekerek, A. (2021). A novel architecture for web-based attack detection using convolutional neural network. Computers and Security, 100, 102096.

[17] Chen, D., Yan, Q., Wu, C., and Zhao, J. (2021). Sql injection attack detection and prevention techniques using deep learning. In Journal of Physics: Conference Series (Vol. 1757, No. 1, p. 012055). IOP Publishing.

[18] Liu, X., Yoo, C., Xing, F., Oh, H., El Fakhri, G., Kang, J. W., and Woo, J. (2022). Deep unsupervised domain adaptation: A review of recent advances and perspectives. APSIPA Transactions on Signal and Information Processing, 11(1).

[19] Rozantsev, A., Salzmann, M., and Fua, P. (2018). Beyond sharing weights for deep domain adaptation. IEEE transactions on pattern analysis and machine intelligence, 41(4), 801-814.

[20] Kang, G., Jiang, L., Yang, Y., and Hauptmann, A. G. (2019). Contrastive adaptation network for unsupervised domain adaptation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4893-4902).

[21] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7167-7176).