Hindawi
Security and Communication Networks
Volume 2021, Article ID 6661124, 11 pages
https://doi.org/10.1155/2021/6661124

Research Article

An Improved Feature Extraction Approach for Web Anomaly Detection Based on Semantic Structure

Zishuai Cheng,[1,2] **Baojiang Cui**,[1,2] **Tao Qi**,[3] **Wenchuan Yang**,[1,2] **and Junsong Fu**[1,2]

[1]*School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*The National Engineering Laboratory for Mobile Network, Beijing 100876, China*
[3]*School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Baojiang Cui; cuibj@bupt.edu.cn

Received 1 December 2020; Revised 4 January 2021; Accepted 27 January 2021; Published 11 February 2021

Academic Editor: Zhe-Li Liu

Anomaly-based Web application firewalls (WAFs) are vital for providing early reactions to novel Web attacks. In recent years, various machine learning, deep learning, and transfer learning-based anomaly detection approaches have been developed to protect against Web attacks. Most of them directly treat the request URL as a general string that consists of letters and roughly use natural language processing (NLP) methods (i.e., Word2Vec and Doc2Vec) or domain knowledge to extract features. In this paper, we proposed an improved feature extraction approach which leveraged the advantage of the semantic structure of URLs. Semantic structure is an inherent interpretative property of the URL that identifies the function and vulnerability of each part in the URL. The evaluations on CSIC-2020 show that our feature extraction method has better performance than conventional feature extraction routine by more than average dramatic 5% improvement in accuracy, recall, and F1-score.

1. Introduction

Web attack still is one of the largest IT security threats with many types of Web attacks (e.g., SQL injection, cross-site scripting, and Web-shell) in the rapid development of 5G, IoT, and cloud computing. Web-based applications provide various services, such as e-commerce, e-government, e-mail, and social networking, for individuals and organizations [1, 2]. Users usually store their sensitive data on these applications. The importance and sensitivity of the Web-based application make it into an attractive target for attackers. Defending Web-based applications from attacks is a challenging task because cyber-defence is asymmetric warfare as the attackers have great advantage than defenders [3]. The intrusion detection system continuously identifies attacks relying on the up-to-date signature or model, while attacker only needs a single vulnerability for victory. Unknown attacks, specifically *Zero-day*, are difficult to identify by the signature-based intrusion detection system and can cause great damage to individuals and organizations.

To detect unknown attacks, a great number of anomaly-based intrusion detection methods have been proposed by researchers in recent years. The anomaly detection method can detect unknown attacks by identifying their abnormal behaviours that obviously deviate from the normal behaviours which have been modelled in the training phase [4, 5]. No matter which specific algorithm (i.e., support vector machine, hidden Markov model, and random forests) was used to profile the normal behaviours, feature extraction is essential to the anomaly-based detection model. The widely

used feature extraction methods can be classified into two types: expert knowledge-based models and NLP-based models as follows:

(i) In expert knowledge-based approaches, researchers design a set of handcrafted rules to describe the normal or malicious behaviour of HTTP request, such as whether exits sensitive keyword, the length of each value, and whether contains special character [6, 7]

(ii) In NLP-based approaches, researchers extract contiguous sequences of n characters from the URL [8–11]

Although these methods have achieved a good performance, they roughly treat HTTP request URL as a general string that consists of letters and pay average attention to each character.

Semantic structure is a knowledge that is comprised of a set of information entities, such as the deserving Web resource, number and sequence of logical parts, and the property of each logical part (trivial or salient) [12]. A resource is a function that provides a type of interaction for users by Web application. Consider an e-commerce application, the function can be register, login, view products, or order products. In general, URLs requesting same resource (or function) have the identical semantic structure although the values of logical parts are variable. In a request URL, each logical part plays different roles. Salient logical parts are mostly be used to indicate requesting resource. Values of these parts are stationary or only have a few numbers of values. On the contrary, trivial logical parts are always used to deliver users' input payloads to the server-side program, such as username, page number, delivery address, or product ID.

To the best of our knowledge, the utilization of semantic structure for feature extraction has not been investigated. We see a good reason to believe that the insights gained in the semantic structure carry over to feature extraction. In general, the attacker always manipulates the values of trivial logical parts to attack the Web-based application. On the contrary, the values of salient logical parts are rarely be used to launch attacks. Thus, we should pay more attention to the values of trivial logical parts rather than pay average attention to every logical parts in intrusion detection.

In our preliminary work [13], we introduced an anomaly detection method based on the semantic structure. However, it has some limitations in HTTP request imbalance. Hence, in this paper, we proposed an improved feature extraction approach that efficiently uses the semantic structure. This approach helps the anomaly-based detection model pay more attention to sensitive trivial parts which are more likely used by the attacker to launch attacks. A method that can automatically learn semantic structure by observing training dataset is proposed in this paper. We further eliminate the request imbalance by using skeleton structure to improve the accuracy of the semantic structure. Request imbalance is a serious problem which is caused by the fact that some functions are requested more frequently than others, such as viewing product function is more likely to be requested than ordering product function. The evaluation results show the anomaly-based detection models with the semantic structure outperform other models that were built with conventional feature extraction procedure.

To learn the semantic structure and use it to help build a detection model, we first define a notion of skeleton structure for the URL and classify URLs into several subgroups based on their skeleton structure. Then, we propose a statistical-based algorithm to learn the semantic structure from each group, respectively, and then combine these independent semantic structures into an entire semantic structure. Pattern-tree which is proposed by Lei et al. is used to encode the semantic structure [12]. After that, we build the anomaly-based detection model for each trivial logical part by observing their values. Finally, we introduce how to detect anomaly attacks based on the semantic structure and the built detection model.

Based on the semantic structure, the anomaly detection model can pay more attention to detect the values of trivial logical parts. Thus, the detection model using semantic structure is more sensitive and precise to detect attacks.

The contributions of this paper can be summarized as follows:

(i) An enhanced feature extraction approach is proposed for Web anomaly detection. This approach takes the advantage of semantic structure to pay more attention to trivial logical parts which are more vulnerable than salient parts. Compared with conventional feature extraction methods, the significant innovation is that we treat the URL as a combination of meaningful logical parts rather than meaningless string that consists of letters.

(ii) We proposed a notion of skeleton structure which is used to eliminate the request-imbalance problem. This method can improve the accuracy of the learned semantic structure.

(iii) We evaluate our approach on CSIC-2010 dataset [14]. Experimental results show that the semantic structure is vital to improving the performance of the anomaly-based intrusion model.

The rest of this paper is organized as follows. In Section 2, we introduce the related work focusing on anomaly-based detection and semantic structure. The framework of our approach and the details of how to learn semantic structure are separately introduced in Sections 3 and 4. The method that to build the anomaly-based detection model for each trivial logical part is described in Section 5. In Section 6, we illustrate how to use semantic structure and the built detection model to detect attacks. In Section 7, we report the simulation environment and experiment results. Finally, we draw conclusions and future points in Section 8.

2. Related Work

Since anomaly-based intrusion detection was firstly introduced in 1987 by Denning [15], research in this area has been

rapidly developed and attracted lots of attention. A great number of methods have been proposed by researchers in recent years. According to the types of algorithms that used to build the detection model, the anomaly-based WAF can be categorized into statistics, data mining, machine learning, and deep learning-based. No matter which specific algorithm is used, feature extraction always is an important part of building the anomaly-based detection model. The feature extraction methods can be widely divided into expert knowledge-based and NLP-based.

In the field using expert knowledge to extract features from URLs, Cui et al. proposed a feature extraction approach which extracts 21 features from HTTP request based on domain knowledge to describe the behaviour of HTTP request [7]. Then, they train a random forest (RF) classification model based on these features to classify HTTP request as normal and anomaly. Niu and Li extracted eight features with good classification effect to augment the original data [16]. Tang et al. proposed an approach that extracts behaviour characteristics of SQL injection based on the handcrafted rules and uses the long short-term memory (LSTM) network to train a detection model [17]. And, authors combined expert knowledge with N-gram feature for reliable and efficient Web attack detection and used the generic-feature-selection (GFS) measurement to eliminate redundant and irrelevant features in [18, 19]. Zhou and Wang proposed an ensemble learning approach to detect XSS attack [20]. The ensemble learning approach uses a set of Bayesian networks which is built with both domain knowledge and threat intelligence. More recently, Tama et al. proposed a stack of the classifier ensemble method which relies on the handcrafted features [21]. All these authors extract features mostly based on their expert knowledge. These handcrafted features have achieved a good performance in these datasets. However, there exists a strong difference between the network environments or the behaviours of Web applications. These selected features that perform well in one training dataset may not perform well in other Web applications.

To address the problem of expert knowledge-based feature extractions, lots of researchers use natural language processing (NLP) and neural network (NN) to automatically learn the significant features and build a powerful anomaly detection model. Kruegel et al. proposed an anomaly detection system for Web attacks, which takes advantage of the particular structure of HTTP query that contains parameter-value pairs [22, 23]. In this paper, authors built six models to detect attacks in different aspects such as attribute's length, character distribution, structural inference, token finder attribute presence or absence, and attribute order and separately output the anomaly probability value. The request is marked as malicious if one or more features' probability exceeds the defined threshold. Cho and Cha proposed a model which uses Bayesian parameter estimation to detect anomalous behaviours [24]. PAYL is proposed by Wang and Stolfo which uses the frequency of N-grams in the payload as features [25]. Tian et al. used continuous bag of words (CBOW) and TF-IDF to transform the HTTP request into vector [26, 27]. Both are the popular algorithms for text analysis in the field of NLP. Wu et al. exploited word embedding techniques in NLP to learn the vector representations of characters in Web requests [28]. Tekerek used bag of words (BOW) to produce a dictionary and convert HTTP request as a $200 \times 170 \times 1$ matrix [29]. If the payload matches an entry in the dictionary, the label is set to 1 that is represented with white pixel in image; if it does not, it is set to 0 that is represented with black pixel in image. Then, Tekerek used the conventional neural network (CNN) to learn the normal pattern of HTTP request and detects attacks. All these authors focus their efforts on solving the problem of how to build the behaviour models that can significantly distinguish abnormal behaviour from normal behaviour without a lot of human involvement. They ignore the semantic structure of HTTP request and treat the URLs as a general string that is comprised of letters and extract features directly from these URLs.

No matter using expert knowledge-based feature extraction methods or N-gram-based methods, the anomaly detection model will pay average attention to every letter or logical part. Thus, these models are taking the negative effects of some redundant letters or useless logical parts. Thus, it is necessary to use semantic structure to help the model pay more attention to those vulnerable logical parts.

To the best of our knowledge, there are few Web instruction detection methods that use the semantic structure of URLs. However, in other research areas, some researchers had taken advantage of it. Lei et al. proposed a concept of pattern-tree to learn the semantic structure of URLs [30]. They proposed a top-down strategy to build a pattern-tree and used statistic information of the values of logical parts to make the learning process more robust and reliable. Yang et al. further proposed an unsupervised incremental pattern-tree algorithm to construct a pattern-tree [31]. Our approach that is used to learn semantic structure is inspired by these works. However, in our approach, we take account of the negative effect of request imbalance that wildly exists in Web applications and we introduce a concept of skeleton to eliminate request imbalance.

3. Framework and Definition

Without loss of generality, we mainly analyse the HTTP requests using the GET method in this paper. Although we focus on GET requests here, our method can be extended to other methods easily by converting users' data to the parameter-value pairs format that is similar to GET.

As shown in Figure 1, our approach is composed of three steps. In the learning step, we eliminate the request-imbalance problem, learn separate semantic structure from each subgroup, and merge these independent subsemantic structures into an entire semantic structure. Then, in the building anomaly-based detection model step, we build models for each trivial logical part by observing its values. In the detection step, the method that classifies new HTTP request as normal or abnormal based on the semantic structure and learned model is proposed.

Before introducing our model, we first define the training dataset of URLs as $U = \{u_1, u_2, \ldots, u_m\}$, in which u_i is the i^{th} request URL. According to HTTP protocol [32],

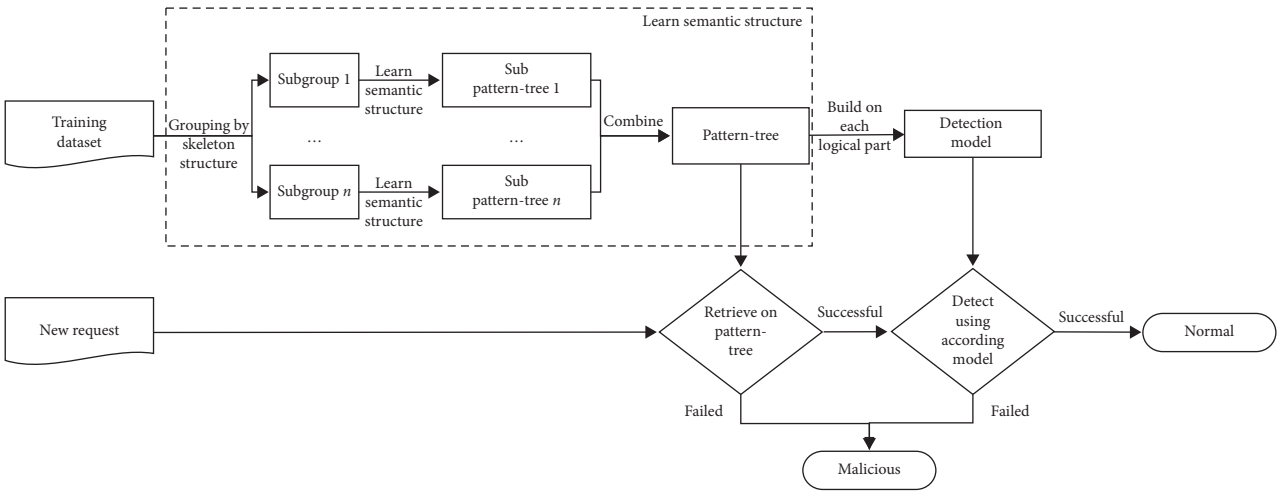

Figure 1: The framework of the improved feature extraction approach.

each request URL u can be decomposed into several components (e.g., scheme sch, authority auth, path path, optional path information component opinfo, and optional query string query) by delimiters like ":," "/," and "?." Components before "?" are called static parts (e.g., scheme, authority, path, and opinfo), and the rest of components (e.g., query) are called dynamic part. path can be further decomposed into a collection of parameter-value parts, also called logical parts, according to its hierarchical structure like $pv_{\text{path}} = \{(p_1, v_1), (p_2, v_2), \ldots, (p_n, v_n)\}$, where v_i is the i^{th} segment value in path split by "/" and p_i is the index of v_i represented as "path-i.". The dynamic part query is usually used to transmit the values submitted by end-users to the server-side program. query can further be decomposed into a collection of parameter-value parts or logical parts like $pv_{\text{query}} = \{(p_1, v_1), (p_2, v_2), \ldots, (p_n, v_n)\}$, in which p_i is the name of the i^{th} parameter in query split by "?" and v_i is the corresponding value of the i^{th} parameter. Finally, we combine pv_{path} and pv_{query} into a parameter-value collection pv.

However, the confusion between the function of logical parts in path and query proposes a challenge to determine a logical part is trivial or salient. Path path not only identifies the requesting resource but also sometimes contains the values submitted by end-users. query also can contain the identifier that indicates the requesting resource. Especially, the rapid development of search engine optimization (SEO) aggravates the confusion problem [33, 34]. Thus, we propose a top-down method to infer the function and semantic of each logical part in path and query and learn the semantic structure. This method will be introduced in detail in the next section.

4. Learn Semantic Structure Information

Our method can automatically learn semantic structure in three major steps: eliminating request-imbalance problem, learning semantic structure form each subgroup, and merging all independent part semantic structures into an entire semantic structure.

4.1. Eliminating Request-Imbalance Problem. As noticed before, request imbalance presents a major challenge of learning semantic structure accurately. For example, in an e-commerce website, users are more likely to choose and order products compared with register or login. Thus, logical parts contained in choose and order functions are requesting more frequently than others and have more appearance frequency than others. Thus, these logical parts are more likely determined as salient even if it is trivial.

As we all know, each URL has its basic structure (e.g., scheme, authority, depth of path, number, and sequence of logical parts in query). URLs which request same resource have same basic structure. Thus, we can split URLs into several subgroups based on their basic structures. For a Web application, the scheme and authority are mostly invariant. And thus, in this paper, we mainly use the priorities of path and query to divide URLs into subgroups.

To spilt URLs into subgroups, we firstly extract pv_{path} and pv_{query} for each URL u. Then, we construct a hash key using the size of pv_{path} and the parameter sequence of pv_{query} to split URLs into subgroups. The URLs with the same size and parameter sequence are classified into one subgroup. As shown in Figure 2, we split URLs showed in Table 1 into four subgroups according to their basic structure. After that, we can separately learn semantic structure from each group.

Splitting URLs in subgroups cannot change the fact that there has a request-imbalance problem. However, we can limit the imbalance between URLs to the imbalance between subgroups and ensure the URLs in each subgroup are request-balance. Thus, this method eliminates the impact of the request imbalance on the accuracy of the learned semantic structure.

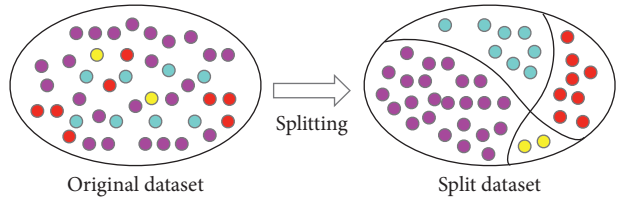

Figure 2: An example that illustrates the processing of splitting. By using this method, we divide those four types of URLs (as shown in Table 1) into four subgroups. Each URL in subgroups has the same basic structure and requests the same resource. Thus, the URLs in each subgroup are balanced.

Table 1: Four examples of URL and each URL represents a type of HTTP requests.

No.	URL	Parameter sequence	Semantic structure
1	/question/search?q = docker	Path 1, Path 2, q	question/search?q = *
2	/question/top?q = windows server &page = 10	Path 1, Path 2, q, page	question/top?q = *&page = *
3	/user/news?page = 1	Path 1, Path 2, page	/user/news?page = *
4	/teams/create/Linux	Path 1, Path 2, Path 3	/teams/create/*

Parameter sequence is the parameter-sequence extracting from pv_{path} and pv_{query}. Semantic structure is the semantic structure information of each type of HTTP requests. The "*" in semantic structure denotes the corresponding part is trivial, and other symbols mean the corresponding segments are salient.

4.2. Learn Semantic Structure and Construct Pattern-Tree. The crucial thing in learning semantic structure is to determine the logical part whether is trivial or salient. In this section, we will introduce the method about learning semantic structure in detail.

According to our observation, different logical parts (or components) play different roles and have distinct different appearance frequencies. In general, salient parts denoting directories, functions, and document types only have a few numbers of values, and these values have high appearance frequencies. In contrast, trivial parts denoting parameters such as usernames and product IDs have quite diverse values, and these values have low appearance frequencies.

Thus, we proposed an approach to determine the property of the logical part based on its entropy and the number of distinct values. The entropy for a logical part is defined as $H(K) = \sum_{i=1}^{V} -(v_i/N)\log(v_i/N)$, where V is the number of distinct values of this part, v_i is the frequency of the i^{th}value, and N is the number of total values. We determine the logical part whether is trivial according to the following equation:

$$\text{logical part } i = \begin{cases} \text{trivial}, & \text{others}, \\ \text{salient}, & \text{if } H(k) < \lambda \log V \text{ or } V, \end{cases} \quad (1)$$

where $\lambda \in [0, 1]$ and $\gamma \in \mathbb{N}$ are two hyperparameters to control the sensitivity of the learned semantic structure.

As shown in Algorithm 1, we proposed a top-down algorithm to recursively split the URLs into subgroups and build a pattern-tree in the meantime. We determine the logical part whether is salient or trivial according equation (1) in each splitting process. Values are reserved in V^* if the logical part is salient. Otherwise, values will be generalized as "*" and V^* is set as {'*'}. * is a wildcard character that represents any characters. According to the values in V^*, we can split URLs into subgroups. Then, we further determine the next logical part as salient or trivial on each subgroup recursively. This determining and splitting process is repeated until the subgroup is empty. Finally, we learn a pattern-tree $\mathcal{N}_i$ form subgroup U_i. Each path from the root to leaf in this tree is a piece of semantic structure information. Each node in the pattern-tree represents a logical part of the URL. And, the type of node is identified by to its value.

After applying the construct pattern – tree algorithm to each subgroup, we finally get several independent pattern-trees. Then, we can merge these independent pattern-trees into an entire patten-tree that describes the whole semantic structure of Web application. Figure 3 shows the processing of learning semantic structure and constructing pattern-tree. There are four pattern-trees separately learned on U_1, U_2, U_3, and U_4 using the construct pattern – tree algorithm. Then, we merge these four independent trees into an entire pattern-tree $\mathcal{T}$ as shown on the right. The entire pattern-tree describes the whole semantic structure and is used to build the anomaly detection model and detect attacks.

The entire pattern-tree can be retrieved using parameter-value collection KV. For example, for a request URL "/question/search?q = docker," we retrieve a path on pattern-tree according to parameter-value collection $pv = \{(\text{path}_1, \text{question}), (\text{path}_2, \text{search}), (q, \text{docker})\}$.
Firstly, we examine the first parameter-value pair $(\text{path}_1, \text{question})$ on pattern-tree. If the parameter-value pair exists, it indicates that this parameter-value pair is valid and we further examine the next parameter-value pair on the corresponding child-tree. Otherwise, we replace the value of this parameter-value pair with "*" and re-examine it. This process is repeated until all parameter-value pairs in pv are examined or subtree is null or parameter-value pair not exists. For this request URL, the successful retrieval path is shown in Figure 3 and is marked with the red dash arrow. This path shows that the semantic structure is "/question/

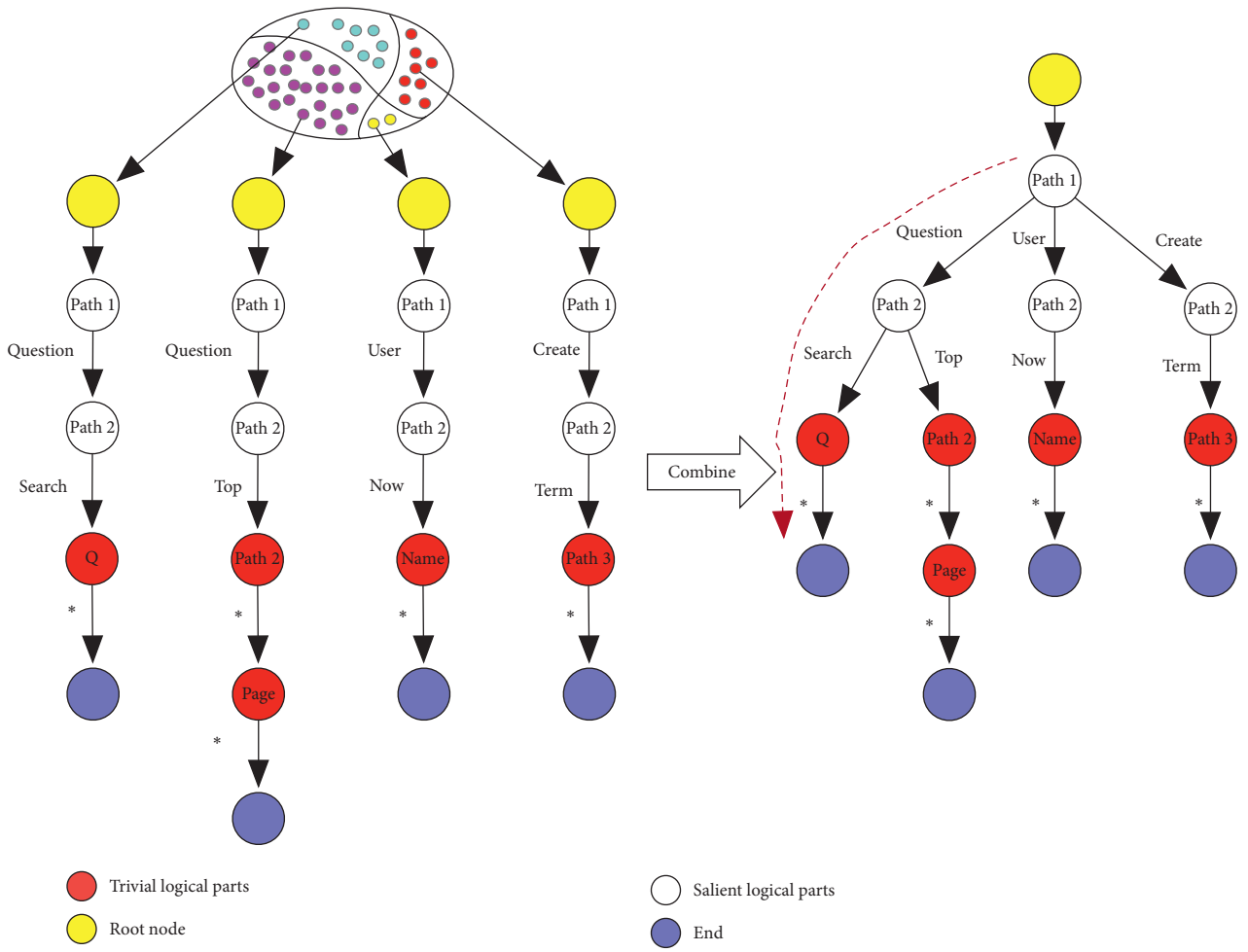

Figure 3: The processing of learning semantic structure and constructing pattern-tree. We first separately learn semantic structure and construct pattern-tree on each subgroup. Then, we merge these independent semantic structures into an entire pattern-tree which describes the whole semantic structure of Web application. The entire pattern-tree is used in building the anomaly detection model and detecting malicious.

search?q = ∗," where the parameter q is trivial and the value of parameter q is more vulnerable than others.

5. Build Anomaly Detection Model

As mentioned earlier, the values of trivial logical parts change frequently and depend on users' input. Values of these trivial logical parts are mostly crafted by attackers to attack Web application. Thus, in anomaly detection, we can pay more attention to trivial logical parts to improve the accuracy and efficiency of the detection model.

We firstly split HTTP request URLs U into several subsets $U_1, \cdots$, and U_n according to pattern-tree $\mathcal{T}$, where n is the number of semantic structure pieces in pattern-tree (also is the number of paths from the root to leaves). The subset U_i has the following characters:

(i) $\forall u \in U_i$, u has the same semantic structure

(ii) $\forall i \neq j, U_i \cap U_j = \varnothing$

(iii) $U_1 \cup U_2 \cup \cdots \cup U_n = U$

We further extract the value of each trivial part from a URL u and combine them as a vector $v_{\text{trivial}} = \{v_1, \ldots, v_q\}$, where v_i is the value of i^{th} trivial logical part. Furthermore, we combine v_{trivial} of each u in U_i as a $m \times q$ matrix $\mathrm{P}_{\text{trivial}}$, as shown in equation (2), where m is the numbers of URLs in U_i. The j^{th} column $[v_{1j}, v_{2j}, \ldots, v_{mj}]$ is the value of j^{th} trivial part for all URLs in U_i:

$$P_{\text{trivial}_i} = \begin{bmatrix} v_{11} & \cdots & v_{1q} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{mq} \end{bmatrix}. \tag{2}$$

We build anomaly-based intrusion detection models for each logical part by observing the corresponding column values in P_{trivial_i}. Finally, each node of pattern-tree that represents a logical part maps a detection model. The entire

anomaly detection model M of this Web application is composed of several submodels $\{m_{11}, \ldots, m_{1q_1}, \ldots, m_{21}, \ldots, m_{2q_2}, \ldots, m_{nq_n}\}$, where m_{ij} is built by observing the values of j^{th} column in P_{trivial_i}.

The specific algorithm used to build the anomaly-based detection model is beyond the scope of this paper. Our method can integrate with any anomaly-based detection algorithm to build more precious model for detection attacks.

6. Detect Malicious Attacks

In this section, we will introduce the approach to detect malicious attack according to the pattern-tree $\mathcal{T}$ and anomaly-based detection model M. The URL is detected in the following two levels. (a) Semantic structure level: we retrieve the URL on pattern-tree $\mathcal{T}$ to determine whether the new request matching the exiting semantic structure; (b) Value level: we then detect the values of each trivial logical part whether is anomaly using the corresponding learned anomaly-based detection model M. As long as the new request does not follow the existing semantic structure or any value of trivial logical parts, it will be classified as an anomaly. Otherwise, we determine it as benign.

More specifically, we first convert the URL u into parameter-value collection pv before detecting the HTTP request. Then, we retrieve pattern-tree $\mathcal{T}$ using pv. We simultaneously detect the value of trivial logical part whether is abnormal in the retrieve process. If a value is determined as an anomaly, we stop further retrieving and directly report this HTTP request as abnormal. If the URL of new request does not fit the expectation of $\mathcal{T}$ (e.g., there exists any parameter-value pair that has not examined when a null subtree is reached, and the subtree is not null after all parameter-values pairs are examined), we report the HTTP request as abnormal. Only if the new request satisfies both the semantic structure and anomaly-detection model, we classify it as normal.

7. Experiments

To evaluate the effectiveness of our approach, we implemented a prototype of our proposed method sketched in Figure 1. The components are implemented in Python and Scikit-learn 0.23. And, the dataset used in evaluation experiments is CSIC-2010 [14].

7.1. Experimental Settings

7.1.1. Dataset Description. CSIC-2010 is a modern Web intrusion detection dataset introduced by the Spanish National Research Council which includes two classes: normal and anomalous. It contains thousands of Web requests automatically generated by creating traffic to an e-commerce Web application using Paros proxy and W3AF. The dataset consists of three subsets: 36,000 normal requests for training, 36,000 normal requests, and 25,000 anomalous requests for testing. There are three types of anomalies in this dataset: static attack that requests for the hidden (nonexistent) resource, dynamic attack that craftily modifies the value of logical part to attack Web application, and unintentional illegal request that does not follow the normal semantic structure; however, it has no malicious payload. The dataset consists of HTTP requesting for several resources with two request methods: GET and POST.

7.1.2. Metrics. There are numbers of performance metrics that can be used to evaluate the performance of the anomaly detection system. The most commonly used metrics in this field are precision, recall, F1-score, and accuracy (ACC). In this paper, we also use these metrics to evaluate our approach:

(i) Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives given as follows:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}. \quad (3)$$

(ii) Recall is defined as the percentage of positive cases you caught given as follows:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \quad (4)$$

(iii) F1-score is the harmonic mean of precision and recall taking both metrics into account given as follows:

$$F_1 = 2^* \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \quad (5)$$

(iv) Accuracy measures in percentage form where instances are correctly predicted given as follows:

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{false positives} + \text{true negatives} + \text{false negatives}}. \quad (6)$$

8. Results and Discussion

The hyperparameters λ and γ play significant role to control the accuracy of pattern-tree. With the best λ and γ, the learned pattern-tree achieves an appropriate tradeoff between the size and integrity. With the increase in λ or γ, the policy to determine logical part whether is trivial or salient is getting more tolerant and more parts are determined as salient.

To choose the best λ and γ, we trained several pattern-trees with different parameters λ from 1 to 9 with step 1 and γ from 0.1 to 0.9 with step 0.1 on all GET method URLs in training dataset. As shown in Figure 4, it is obvious that with

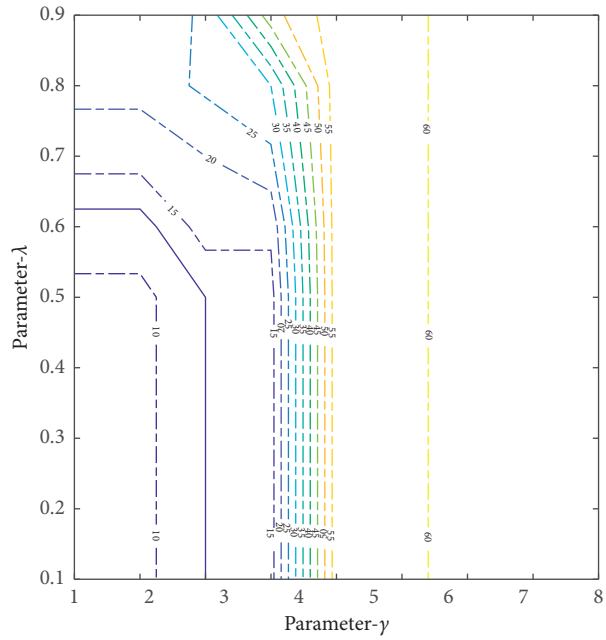

FIGURE 4: The number of resources recognized in training dataset on different parameters λ and γ.

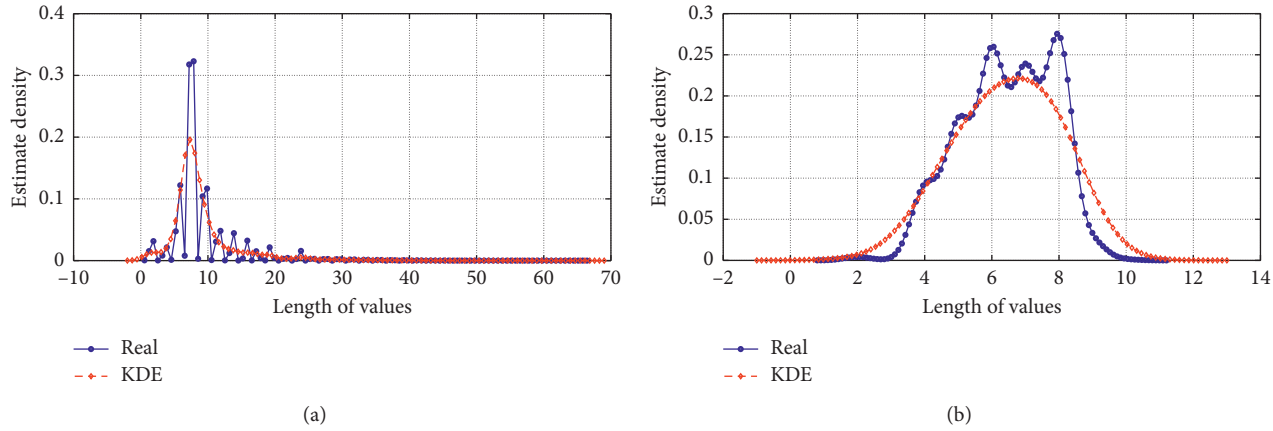

FIGURE 5: The difference of kernel density estimate (KDE) of length between all values and values of an example trivial logical part. (a) The KDE of length for all request values. (b) The KDE of length for request values for an example trivial logical part. It is obvious that the length distribution using semantic structure is more regular and learnt.

the increasing of γ, the number of semantic structure pieces encoded in $\mathcal{T}$ is increasing rapidly. The cause of this phenomenon is that γ pays more significance than λ in controlling the tolerance of determining a logical part whether is trivial. The solid blue line in Figure 4 is the ground true number of resources in this Web application. In this paper, we chose hyperparameter λ as 0.3 and γ as 3.

To demonstrate how semantic structure helps to build a more precise anomaly-detection model, we compared the distribution of length feature which is separately extracted with and without using semantic structure. Length feature is a common feature to measure the length of value and is widely used in many anomaly detection types of research.

Figure 5 shows the comparison of these two distributions. The probability distribution and kernel density estimation (KDE) of the original length feature observed from all URLs are shown in Figure 5(a). In contrast, Figure 5(b) shows the probability distribution and KDE which are observed from an example logical part. It is obvious that the distribution shown in Figure 5(a) is more regular than Figure 5(b) and is further easy to be profiled by the anomaly-based detection model. This experiment shows that the semantic structure has significant point to improve the learning ability and accuracy of the detection model.

Finally, we further implemented an experiment to demonstrate that the semantic structure can extremely improve the performance of the detection model. We construct two types of models. One is using the conventional routine that directly extracts the features on the dataset using the feature proposed in [7] and trains the anomaly detection

(i) **Input**: given a subgroup U obtained by Section 5.1 and initialize j as 1
(ii) **Output**: a tree node $\mathcal{N}$ for URLs in U
(1) Create a new node $\mathcal{N}$ and extract parameter-value collection kv for a random URL
(2) **if** $j >$ the size of kv, **then**
(3) **return** the node $\mathcal{N}$
(4) **end if**
(5) extract pv for each URL in U, and combine the value of j^{th} parameter into collection K
(6) calculate $H(K)$ of K
(7) **if** $H(K) < \lambda \log V$ or $V < \gamma$, **then**
(8) $V^* =$ the set of distinct values in K
(9) **else**
(10) $V^* = \{' * '\}$
(11) **end if**
(12) further split U into several subgroups $\{U_1, \ldots, U_t\}$ according to V^*
(13) **for** all subgroup U_i **do**
(14) child = construct pattern − tree$(U_i, j + 1)$
(15) add child as a child of node $\mathcal{N}$
(16) **end for**
(17) **return** the node $\mathcal{N}$

Algorithm 1: Construct pattern − tree(U, j).

Table 2: The performance comparison of difference models.

Algorithm	Precision	Recall	F1-score	Accuracy
Random forest	0.8899	0.7860	0.8348	0.8169
Random forest*	0.9154	0.8508	0.8819	0.8376
Decision tree	0.8646	0.8075	0.8351	0.8124
Decision tree*	0.9169	0.8454	0.8797	0.8352
Support vector machine	0.6746	0.9090	0.7745	0.6912
Support vector machine*	0.9636	0.8541	0.9056	0.8731
K-neighbours	0.8879	0.7764	0.8292	0.8109
K-neighbours*	0.9335	0.8491	0.8893	0.8493

The algorithms with * are trained with semantic structure, and others are trained with conventional routine.*

model. Other is trained within the semantic structure. The specific machine learning algorithms used in this experiment are random forest, decision tree, support vector machine, and K-neighbours. The hyperparameters of these models are not tuned but only used the default parameter value initialized in Scikit-learn.

Table 2 shows the comparison results. It is obvious that the performance of the detection model is briefly enhanced by using semantic structure. In random forest, decision tree, and K-neighbour-based detection model, the F1-score has considerable average 5% improvement. Especially in the support vector machine-based model, F1-score has dramatic 13% improvement. The significant improvements in precision, recall, F1-score, and accuracy in different machine learning algorithms strongly suppose the importance of semantic structure.

As highlight earlier, there exist three types of anomalies in CSIC-2010. Our anomaly detection model can efficiently perform detection than traditional models. In conventional scenarios, no matter static attacks, dynamic attacks, or unintentional attacks, the anomaly detection model has to inspect each value or character in the requesting URL. However, in our method, most of static and unintentional attacks can be detected by semantic structure because these URLs seriously violate the learned semantic structure (e.g., the value of salient logical part that has not observed in training dataset presents and there still exits pair that has not been inspected in kv when semantic structure tree has reached the bottom). Moreover, our method pays more attention to the values of vulnerable logical parts and builds a more precise detection model. Because our method detects little volume of URLs and has more precise model than conventional models, we achieve a significant lower false positive and higher accuracy.

9. Conclusion and Future Work

We introduced an enhanced feature extraction method for Web anomaly detection that uses semantic structure of request URLs. We propose to use skeleton structure to eliminate the request-imbalance problem. By using semantic structure, the detection model is able to pay more attention to the vulnerable logical parts and produces a precise model.

The feature distribution comparison demonstrates the reason why the semantic structure can help to improve the performance of the detection model. And, the improvement

of performance shown in Table 2 also indicates the value of semantic structure.

We plan to study how to learn the nonstationary semantic structure with an increment learning mechanism. To provide a better service for users, Web application is constantly evolved, such as adding new or removing old resources and changing the parameters of some resources.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

The previous version of this research work was published at the 6th International Symposium, SocialSec 2020. The content in this new version has been extended by more than 30%.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China (nos. 61941114 and 62001055) and Beijing Natural Science Foundation (no. 4204107).

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web services," in *Web Services*, pp. 123–149, Springer, Berlin, Germany, 2004.

[2] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: a review," *Computers & Security*, vol. 30, no. 6-7, pp. 353–375, 2011.

[3] W. Yurcik, J. Barlow, and J. Rosendale, "Maintaining perspective on who is the enemy in the security systems administration of computer networks," in *ACM CHI Workshop on System Administrators Are Users*, Fort Lauderdale, FL, USA, April 2003.

[4] D. M. Hawkins, *Identification of Outliers*, Vol. 11, Chapman and Hall, London, UK, 1980.

[5] V. Jyothsna, V. Rama Prasad, and K. Munivara Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011.

[6] R. Funk and N. Epp, "Anomaly-based web application firewall using http-specific features and one-class svm," *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação*, vol. 2, p. 1, 2018.

[7] B. Cui, S. He, X. Yao, and P. Shi, "Malicious URL detection with feature extraction based on machine learning," *International Journal of High Performance Computing and Networking*, vol. 12, no. 2, pp. 166–178, 2018.

[8] R. Pal and N. Chowdary, "Statistical profiling of n-grams for payload based anomaly detection for HTTP web traffic," in *Proceedings of the 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Indore, India, December 2018.

[9] A. M. Vartouni, S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect web attacks using stacked auto-encoder," in *Proceedings 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, Kerman, Iran, March 2018.

[10] W. Khreich, B. Khosravifar, A. Hamou-Lhadj, and C. Talhi, "An anomaly detection system based on variable N-gram features and one-class SVM," *Information and Software Technology*, vol. 91, pp. 186–197, 2017.

[11] M. Zolotukhin, T. Hämäläinen, T. Kokkonen, and J. Siltanen, "Analysis of HTTP requests for anomaly detection of web attacks," in *Proceedings 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, Dalian, China, August 2014.

[12] T. Lei, "A pattern tree-based approach to learning URL normalization rules," in *Proceedings of the 19th International Conference on World Wide Web*, 2010.

[13] Z. Cheng, B. Cui, and J. Fu, "A novel web anomaly detection approach based on semantic structure," in *International Symposium on Security and Privacy in Social Networks and Big Data*, pp. 20–33, Tianjin, China, August 2020.

[14] Giménez, C. Torrano, A. P. Villegas, and G. Álvarez Marañón, "HTTP data set CSIC 2010," Information Security Institute of CSIC, Spanish Research National Council, Madrid, Spain, 2010, https://www.tic.itefi.csic.es/dataset/.

[15] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[16] Q. Niu and X. Li, "A high-performance web attack detection method based on CNN-GRU model," in *Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, pp. 804–808, Chongqing, China, June 2020.

[17] P. Tang, W. Qiu, Z. Huang, H. Lian, and G. Liu, "Sql injection behavior mining based deep learning," in *Proceedings of the International Conference on Advanced Data Mining and Applications*, Nanjing, China, November 2018.

[18] H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrović, and K. Franke, "Application of the generic feature selection measure in detection of web attacks," *Computational Intelligence in Security for Information Systems*, pp. 25–32, Springer, Berlin, Germany, 2011.

[19] C. Torrano-Gimenez, H. T. Nguyen, G. Alvarez, and K. Franke, "Combining expert knowledge with automatic feature extraction for reliable web attack detection," *Security and Communication Networks*, vol. 8, no. 16, pp. 2750–2767, 2015.

[20] Y. Zhou and P. Wang, "An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence," *Computers & Security*, vol. 82, pp. 261–269, 2019.

[21] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K.-S. Kwak, "An enhanced anomaly detection in web traffic using a stack of classifier ensemble," *IEEE Access*, vol. 8, pp. 24120–24134, 2020.

[22] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington D.C. USA, October 2003.

[23] C. Kruegel, G. Vigna, and W. Robertson, "A multi-model approach to the detection of web-based attacks," *Computer Networks*, vol. 48, no. 5, pp. 717–738, 2005.

[24] S. Cho and S. Cha, "SAD: web session anomaly detection based on parameter estimation," *Computers & Security*, vol. 23, no. 4, pp. 312–319, 2004.

[25] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, Sophia Antipolis, France, September 2004.

[26] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, and Z. Tian, "A novel web attack detection system for internet of things via ensemble classification," *IEEE Transactions on Industrial Informatics*, 2020.

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

[28] J. Wu, Z. Yang, L. Guo, Y. Li, and W. Liu, "Convolutional neural network with character embeddings for malicious web request detection," in *Proceedings of the 2019 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking*, pp. 622–627, 2019.

[29] A. Tekerek, "A novel architecture for web-based attack detection using convolutional neural network," *Computers & Security*, vol. 100, p. 102096, 2021.

[30] T. Lei, R. Cai, J.-M. Yang, Y. Ke, X. Fan, and L. Zhang, "A pattern tree-based approach to learning URL normalization rules," in *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, April 2010.

[31] Y. Yang, "UPCA: an efficient URL-pattern based algorithm for accurate web page classification," in *Proceedings of the 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Zhangjiajie, China, August 2015.

[32] R. Fielding, J. Gettys, J. Mogul et al., *Hypertext transfer protocol–HTTP/1.1*, http://www.hjp.at/doc/rfc/rfc2616.html, 1999.

[33] J. Shi, Y. Cao, and X.-J. Zhao, "Research on SEO strategies of university journal websites," in *Proceedings of the 2nd International Conference on Information Science and Engineering*, Hangzhou, China, December 2010.

[34] M. Cui and S. Hu, "Search engine optimization research for website promotion," in *Proceedings of the 2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, vol. 4, Nanjing, China, 2011.