A Novel Web Anomaly Detection Approach Based on Semantic Structure

Zishuai Cheng(✉), Baojiang Cui, and Junsong Fu

School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100087, China
{chengzishuai,cuibj,fujs}@bupt.edu.cn

Abstract. In recent years, various machine learning, deep learning based models have been developed to detect novel web attacks. These models are mostly use NLP methods, like N-gram, word-embedding, to process URLs as the general strings composed of characters. In contrast to natural language which consist of words, the URL is composed of characters and hardly decomposes into several meaning segments. In fact, HTTP requests have its inherent patterns, which so-called semantic structure, such as the request bodies have fixed type, request parameters have fixed structure in names and orders, values of these parameters also have special semantics such as username, password, page id, commodity id. These methods have no mechanism to learn semantic structure. They roughly use NLP techniques like DFA, attention techniques to learn normal patterns from dataset. And, they also need a mount of dataset to train. In this paper, we propose a novel web anomaly detection approach based on semantic structure. Firstly, a hierarchical method is proposed to automatically learn semantic structure from training dataset. Then, we learn normal profile for each parameter. The experimental results showed that our approach achieved a high precision rate of 99.29% while maintaining a low false alarm rate of 0.88%. Moreover, even on a small training dataset composed of hundreds of samples, we also achieved 96.3% accuracy rate.

Keywords: Anomaly-based detection · Semantic structure · Machine learning

1 Introduction

Web-based applications are more and more popular and provide various services for individuals and organizations [2]. Daily tasks, such as E-commerce, E-government, E-mail, and social networking, are mostly processed via Web-based applications. Meanwhile, users usually store sensitive data in those applications. The importance and sensitiveness of Web-based applications attract a lot of interest from attacks. Web-based applications are suffering from many types of web attacks such as SQL injection, Cross-site scripting (XSS) attack, Web-Shell attack, etc. [1]

Y. Xiang et al. (Eds.): SocialSec 2020, CCIS 1298, pp. 20–33, 2020.
https://doi.org/10.1007/978-981-15-9031-3_2

Defending Web-based applications from attacks is a challenging task. Cyber-defense is an asymmetric warfare as attackers have great advantages [27]. Intrusion detection systems are continuously identifying attacks relying on the up-to-date signatures or models, while attackers only need a single vulnerability for victory. Anomaly-based intrusion detection approaches provide an ability to detect attacks by identifying abnormal behaviors with deviating from the normal behaviors which have been profiled in training phase [12].

A great number of anomaly-based detection methods have been developed by researchers in recent years. Igino et al. [5] model the sequence and values for each attribute of queriers based on Hidden Markov Model (HMM). Davide et al. [3] propose an intrusion detection approach based on HMM that models the character sequences of the HTTP payload. Wen et al. [9] propose an adaptive anomaly detection model based on HMM. Those researchers introducing the semantic structure in the anomaly detection approach, but these works use HMM mainly to learn the character sequence of URLs which is only a part of semantic structure.

Moreover, Deep learning technique has been used in anomaly-based detection model to learn higher-level features. Qin et al. [19] propose a model which learns semantic of malicious segments in payload using Recurrent Neural Network (RNN) with attentional mechanism. Yu et al. [26] propose a method that uses Bidirectional Long Short-Term Memory (Bi-LSTM) with attention mechanism to model HTTP traffic. Although the attention mechanism can learn the semantic of attack patterns, these models still have drawbacks. URLs are treated as the meaningless general string composed by characters and ignore the semantic structure in HTTP-request scenario. And also, training the deep learning-based models need lots of samples, but high-quality training data is difficult to obtain in the real word [28].

In this paper, we propose a novel anomaly detection approach based on semantic structure of URLs. Firstly, we propose an algorithm that automatically learns semantic structure information from training dataset. We use pattern-tree, logical parts and trivial parts to represent the semantic structure [17]. Each path of this tree is a piece of semantic structure which illustrates a specific structure. Next, we build anomaly detection model for each node of pattern-tree using machine learning technique based on length, characteristic distribution, structure inference. Finally, we classify URL as normal or abnormal using semantic structure and anomaly detection model. This approach considers entire semantic structure of URLs and produces a very precise normal behavior model. Our approach is very sensitive and is able to detect malicious such as web-shell, SQL intrusion, XSS. This approach has a very low false positive rate in despite the fact that it has high sensitiveness. Even on the small training dataset, it still has a good performance.

The contributions of this paper are summarized as follows.

- An efficient Web intrusion detection approach is proposed, based on semantic structure. Compared with previous research which treats the URL as

meaningless string composed by letters, we treat the URL as the meaningful combination of parts.
– We improved the Markov detection model to decrease the size and improve learning ability.
– We evaluated our approach on CSIC-2010 [11] dataset and achieve better performance than previously published results.

The rest of this paper is organized as follows. In Sect. 2 we introduce the related work, focusing on anomaly-based detection research and semantic structure research. The framework of our novel anomaly detection approach is introduced in Sect. 3. In Sect. 4, we report the simulation environment and results. Finally, we draw conclusions and future points in Sect. 5.

2 Related Work

Since anomaly-based intrusion detection was first introduced in 1987 by D. Denning et al. [8], the research associated with this field has been rapidly developed. Kruegel et al. [13], [14] proposed an anomaly detection system for Web-attacks, which takes advantage of the particular structure of HTTP queries that contains parameter-value pairs. kruegel et al. assemble separated models to detect attacks. Each model is built on different features, such as attribute's length, character distribution, structural inference, token finder attribute presence or absence and attribute order and separately outputs the anomaly probability value. The request is marked as malicious if one or more features' probability exceed the defined threshold. Cho et al. [4] proposed a model which uses Bayesian parameter estimation to detect anomalous behaviors. PAYL [24] used the frequency of n-grams in the payload as features. A recent version of PAYL is proposed [23], which add some functionalities such as multiple centroids, and ingress/egress correlation, to the original version. These authors focus their efforts on solving the problem of how to build the behavior models that significantly distinguish abnormal behavior from normal behavior.

More recently, some anomaly detection methods based on feature selection are proposed [6,18,21,22,29]. In [18,22], authors combined expert knowledge with n-gram feature for reliable and efficient web attack detection and use the Generic-FeatureSelection (GeFS) measure to eliminate redundant and irrelevant features. Zhou et al. [29] proposed an ensemble learning approach to detect XSS attack. They use a set of Bayesian networks, which each Bayesian network is built with both domain knowledge and threat intelligence. All these authors defined features based on their expert knowledge. Nevertheless, the selected features are well fitting with the specific environment such as training dataset and not adaptive to various network environments.

To the best of our knowledge, there is few web instruction detection method using the semantic structure. In other research areas, researchers have taken advantage of this information. Lei et al. [17] propose a concept of pattern-tree that leverages the statistic information of the training set to learn URL patterns. This paper uses a top-down strategy to build a tree and uses statistic information to make the learning process more robust and reliable. Yang et al. [25] propose an unsupervised incremental pattern-tree algorithm to construct a pattern-tree and extract main patterns from it to classify Web page.

3 Framework of Our Approach

Without loss of generality, in this paper, we mainly focus on the HTTP request-URLs which using GET method. Although we focus on the GET requests here, our method also can be extended to all request methods, such as POST, HEAD, PUT, by converting the request data or parameters as parameter-value format.

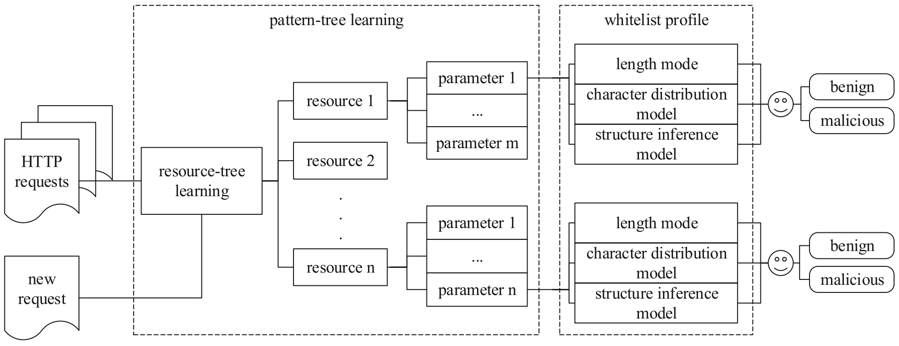

Fig. 1. The framework of our approach.

As shown in Fig. 1, our approach consists of learning phase and detection phase. In learning phase, we mainly learn semantic structure of request URL for website and build anomaly-based detection model for each trivial logical part. In detection phase, we propose an approach which based on semantic structure and anomaly detection model to classify new HTTP request as normal or abnormal.

3.1 Learn Semantic Structure of a URL

We denote the collection of URLs dataset as $U = \{u_1, u_2, \cdots, u_m\}$, in which u_i is the i-th request URL. According to HTTP protocol [10], each request URL u_i can be decomposed into several components (e.g. scheme *sch*, authority *auth*, path *path*, optional path information component *opinfo*, optional query string *query*) by delimiters like ':', '/' and '?'. As shown in Fig. 2, URLs can be decomposed into *sch*, *auth*, *path*, *query*.

Components before '?' are called static parts (i.e., *scheme*, *authority*, *path*, *pinfo*) and the rest components (i.e., *query*) are dynamic parts. *path* usually identifies the requesting resource and has a hierarchical structure. It can be further decomposed into a collection which composed of logical parts $\{(p_1, v_1), \cdots, (p_n, v_n)\}$, v_i is the i-th value in *path* split by '/' and p_i is the corresponding index of v_i. The query string $query$ always contains parameters and corresponding values submitted to server-side programs by users. As same to $path$, $query$ also can decompose into a collection $\{(p_1, v_1), \cdots, (p_n, v_n)\}$, in which p_i is the name of i-th parameter in $query$, v_i is the corresponding values of i-th parameter, n is the numbers of parameter-value pairs in $query$.

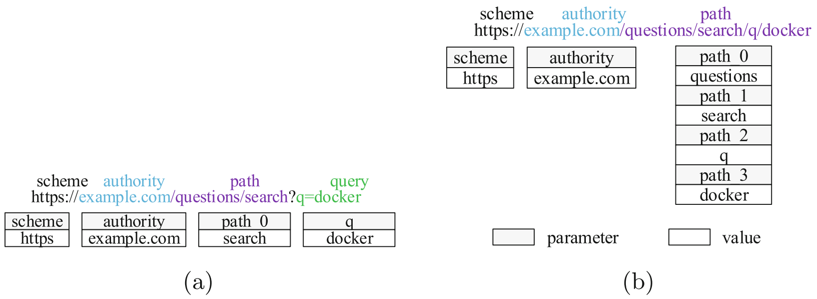

Fig. 2. The syntax structure of URL. a is dynamic URL which $path$ can present as $\{(path_0, question), (path_1, search)\}$ and $query$ can present as $\{(q, docker)\}$. b is a pseudo-static URL.

However, in the real world, $path$ not only identifies the requesting resource but also contains parameter-value pairs. Most Web-based applications use pseudo-static technique [7,20] to make the Web application more friendly to search engine. As Fig. 2b, the pseudo-static is a technique that translates the dynamic parts, like $query$, as a static format and append them as a part of the static part. The pseudo-static technology poses a challenge to learn semantic structure information. We need to learn the function or meaning of each logical part, that is, is this part whether identifying the requesting resource or just a value of parameter submitted by a user.

We use a specialized tree, named pattern-true [17], to learn the semantic structure of request URLs. We keep salient values in pattern-tree's node and generalize trivial values with regular expressions '*'. We determine whether a value is salient or trivial based on its frequency cure and entropy. As shown in Fig. 3, when values' appearance frequencies are stored in descending order, there exists a position in the frequency-curve that has the max frequency of descent. Values on the left-hand side of this position are considered to be salient, on the contrary, values on the right-hand side of this position are considered to be trivial. The position is calculated as: $pos_{dec} = max_i(\log f_i - \log f_{i-1})$, where f_i is the appearance frequency of the i-th logical part.

As shown in Algorithm 1, we use a top-down splitting strategy to divide the URLs into subgroups and build a pattern-tree. First, we determine the first logical part of all URLs as salient or trivial. Each salient value is reserved and all trivial values are generalized as '*'. According to these salient values and '*', we can split URLs into subgroups. Then, we further classify the next logical

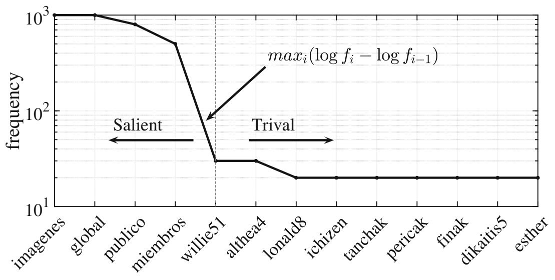

Fig. 3. A example of the values. From the frequency curve, it is clear that the maximum decline point can help distinguish salient values from trivial ones.

part as salient/trivial on each subgroup. We repeat to determine logical part as salient or trivial and divide URLs into subgroups recursively, until the subgroup is empty. Finally, we build a pattern-tree, each path of the tree is a piece of semantic structure information. Each node in the pattern-tree is a logical part in the URL and can illustrates the type of this logical part as salient or trivial.

We retrieve a path of the pattern-tree using the key-value collection kv_i. For example, for a request-URL '/question/search?q=docker', we retrieve the path according to its key-value collection, $kv = \{(p_0, question), (p_1, search), (q, docker)\}$. We examine the first key-value pair $\{p_0, questions\}$ on pattern-tree. If the key-value pair exists, the search is valid and we further examine the next key-value pair in kv on the corresponding child-tree. If the key-value pair does not exist, we replace the value of this key-value pair with '*' and re-examine it. This process is repeated until all key-value pairs in kv are examined or sub-tree is null. For this request-URL shown in Fig. 4, the retrieval path is marked with an arrow. This path shows that the semantic structure is '/question/search?q= * ', where the parameter q is trivial and the value of q can consist malicious payload to launch attacks.

3.2 Build Anomaly-Based Detection Model of a Logical Part

In building anomaly detection model phase, we first divide URLs U into several subsets $\{U_1, U_2, \cdots, U_n\}$ based on semantic structure (also is pattern-tree), where n is the number of subsets that equal to the number of semantic structure of the Web application. The subset U_i has the following characters:

1. $\forall u \in U_i$, URL u has the same semantic structure knowledge.
2. $\forall i \neq j, U_i \bigcap U_j = \varnothing$.
3. $\sum_{i=1}^{n} U_i = U$, n is the number of subsets.

According to semantic structure, we can extract the values of each trivial logical part for URL u and combine these values as a vector $pv = \{(p_1, v_1), (p_2, v_2), \cdots, (p_q, v_q)\}$,where p_i is the index for the i-th logical part and v_i the value of this logical part, q is the number of trivial logical parts in

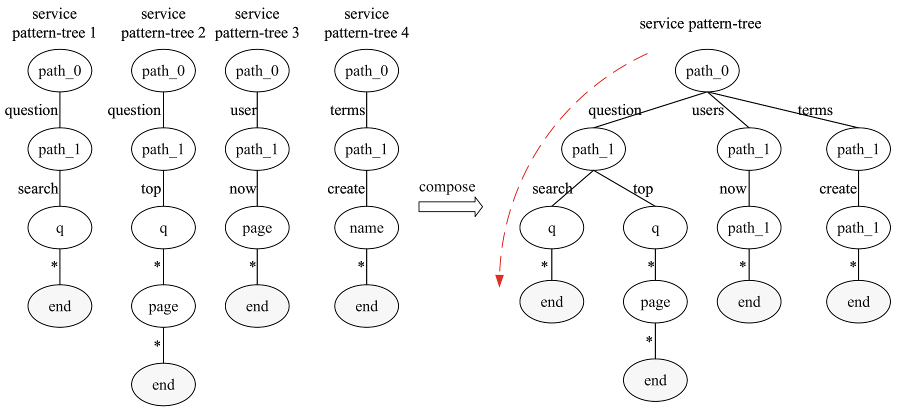

Fig. 4. A example of pattern-tree

Algorithm 1. *ConstructPatternTree* (U, j)

```
Input: Given a URL group $U$ and initialize $j$ as 0
Output: A tree node $t$ for URLs in $U$
1: create a new node $n$
2: if $j >$ the number of parameter-value pairs for URLs in $U$ then
3:     return the node $n$
4: end if
5: extract $j$-th parameter-value pair for each URL in $U$
6: calulate frequency-curve for parameter $k$
7: for URL $u \in U$ do
8:     if value $v$ of $k$ for $u$ is salient then
9:         $V_{k1} = V_{k1} \cup v$
10:    else
11:        $V_{k1} = V_{k1} \cup$ '*'
12:    end if
13: end for
14: calculate entropy $H(k)$ for this $j$-th parameter $k$
15: if $H(k) >$ threshold $t$ then
16:     $V_{k2}$ = the first $max\_number$ values of $U$
17:     $V_k = (V_{k1} \cup$ '*'$) \cap V_{k2}$
18: else
19:     $V_k = V_{k1}$
20: end if
21: split $U$ into sub-groups $\{U_1, U_2, \cdots, U_t\}$ according to $V_k$
22: for all subgroup $U_i$ do
23:     $ch = ConstructPatternTree(U_i, j + 1)$
24:     add $ch$ to $n$ as child node
25: end for
26: return the node $n$
```

u. Furthermore, we extract pv for each URL u in U_i, and combine these pv as a $m \times q$ matrix PV_i:

$$PV_i = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1q} \\ v_{21} & v_{22} & \cdots & v_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mq} \end{bmatrix}$$

where m is the number of URLs in U_i. The j-th column $[v_{1j}, v_{2j}, \cdots, v_{mj}]$ is the values for the j-th trivial logical part for all URLs in U_i.

After extract matrix PV_i for each subset U_i. We observe the values column by column of each PV_i and automatically build profile pro_j for j-th column using statistical learning techniques highlighted in [13]. Then we get q profiles for PV_i. These q profiles consist the integral anomaly detection model $model_i$ for a special piece of semantic structure, also the i-th path of pattern-tree. Each profile describes the normal behavior of request values for logical part in three aspects: length, character distributes and sequence structure of values. All these model $\{model_1, model_2, \cdots, model_n\}$ consist the entire anomaly detection model of this Web application.

3.3 Anomaly Detection

In detection phase, we determine incoming HTTP request as benign or malicious based on semantic structure and anomaly based detection model. When a new HTTP request coming, we first use pattern-tree to illustrate the request URL. If this URL is not successfully retrieved from pattern-tree, we classify this URL as malicious directly. Contrarily, this URL matches j-th path of pattern-tree. According to this path, we determine which logical part is trivial and extract the values of trivial logical parts as a collection pv. For each value v_i in pv, we use the corresponding pro_i in $model_i$ to detect it whether is benign or malicious. If any value of of pv is determined as malicious, the URL is classify as malicious. Otherwise, the URL is classify as benign.

4 Experiment

In order to evaluate the ability of our novel anomaly-detection approach, we conducted several experiments. To analyze the effect of semantic structure information on the performance of our model, we observe the change in length distribution and final classification performance when we control whether structure information is considered. To investigate the sensitivity of the model to the training data size, we tested the performance on different size training sets. And also we show the advantage of using character substitution approach. Finally, we compare our model with five existing models.

4.1 Experimental Settings

Datasets. The experiment was conducted on CSIC-2010 [11], which contains thousands of Web requests automatically generated by creating traffic to an e-commerce web application. The dataset consists of three subsets: 36,000 normal requests for training, 36,000 normal requests and 25,000 anomalous requests for the test. There are three types of anomalous request: static attacks that request for hidden(non-existent) resources, dynamic attacks that modify the valid request arguments, and unintentional illegal requests that have no malicious intention, however they do not follow the normal behavior of the web application and do not have the same structure as normal parameter values [19].

The dataset consists of HTTP requests for several resource and contains two request methods: *GET* and *POST*. According to the desired resource, dataset can divide into two types. One is requesting static resources, such as .jpg, .git, .css, .js format file stored on server. The other is requesting dynamic resources, which need to be processed by the server-side program and the response results are the execution results.

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

– **Precision** is defined as the number of true positives divided by the number of true positives plus the number of false positives.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

– **Recall** is defined as the percentage of positive cases you caught.

$$recall = \frac{true\ positives}{true\ positives + false\ negative}$$

– **F1-score** is the harmonic mean of precision and recall taking both metrics into account.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

– **Accuracy (ACC)** measures in percentage form, where instances are correctly predicted.

$$Accuracy(ACC) = \frac{TP + TN}{TP + TN + FP + FN}$$

4.2 The Comparative Performance of Using Semantic Structure or Not

To illustrate the effect of semantic structure on classification performance. We implement two systems, one using semantic structure and build profile for each type of trivial logical part, and the another is build profile by observing all values in all request-URLs. Then we compare their classification performance.

As shown in Table 1, it is the confusion matrix of these two models. When not using semantic structure information, the detection model has poor performance. For 2500 benign request-URLs, the model predicts 249 request-URLs are malicious with 9.96% false-positive rate. For 2500 malicious request-URLs, the model predicts 906 request-URLs are malicious with 36.24% recall rate and 63.76% false-negative rate. When using semantic structure information, the performance of the anomaly-based detection model has a great improvement. False-positive has reduced to 0.76% and recall has improved to 96.12%. On f1-score, we have improved it from 0.4957 to 0.9722 with 96.12% improvement rate.

Table 1. Confusion Matrix of anomaly-based detection model whether using semantic structure information

(a) The performance of anomaly-based detection model without semantic structure.				(b) The performance of anomaly-based detection model with semantic structure.			
		Actual				Actual	
		Benign	Malicious			Benign	Malicious
Predicted	Benign	2251	1594	Predicted	Benign	2481	106
	Malicious	249	906		Malicious	19	2394

This result shows that using the same methods to build normal-based detection profile, semantic structure can tremendously help us building a precise model and improve the performance. Thus, it is necessary to use semantic structure to improve the performance of the detection model in Web attack detection field.

4.3 The Performance on Different Dataset Size

This experiment intends to measure the impact of the scale of training dataset. We construct several training datasets of different sizes by randomly choosing the request-URLs for each resource. The training datasets consist of 10, 50 to 1000 with 50 steps HTTP request examples for each resource. Then, we train and evaluate our model on each training dataset.

The result is shown in Fig. 5, as the size of training dataset increases, *precision*, *f1-score*, *acc* of this model also increases, although *recall* decreased. The precision score increase from 0.8568 to 0.9483 as the data size increase from 10 to 100. F1-score and accuracy are also increased from 0.9202 to 0.9687 and from 0.9138 to 0.963 separately. Only recall decreased from 0.9936 to 0.99. When feeding more training examples to model, the anomaly-based detection model can learn more precise normal behavior(length, character distributions, and structure). Thus, the classification performance is being better. When the data size is greater than 900, the impact of increasing size of training dataset on classification performance is not obvious. Especially, on the very small training dataset that each resource only has 10 request-URLs examples, our model also achieves 96.2% accuracy.

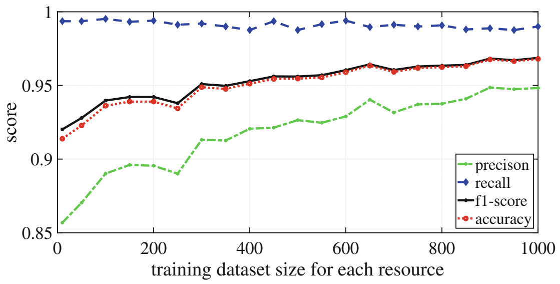

Fig. 5. The performances of our approach to different scales of training datasets.

This result shows that our approach based on semantic structure not require large scale dataset in training phase. On the tiny training dataset, our method also achieves good performance. Compare with other anomaly-based detection methods, especially of deep learning based method need 5000 sample for each class to get an ideal classification performance [16], we can learn enough knowledge to achieve a acceptable performance from limited dataset. Our model solves the knotty problem that there not exits enough scale training dataset to train an anomaly-based detection model in real life.

4.4 Compare with Other Approach

We compared our model with other anomaly-based detection approaches on CSIC-2010 Dataset. The results are described in Table 2 which includes classification performances of SOM, C4.5, Naive Bayes, X-means and EM approaches evaluated in [15]. Compared with all other models on the CSIC-2010 dataset, our model achieves the best performance in Precision, F1-score, ACC, False-Positive rate. Even through, X-means reported the highest recall, it does not perform well in precision, F1-score and accuracy. Our model is very sensitive to detect malicious request and also maintaining a low false-positive rate of 0.88%.

This comparison result shows that our novel detection method based on semantic structure achieves better performance than those methods not using semantic structure.

Table 2. Classification performance of our approach and other approaches

	Precision	Recall	F1-Score	Acc	FP
SOM	0.6980	0.9497	0.8046	0.9282	0.0503
C4.5	0.9654	0.8697	0.9150	0.9650	0.1303
Native Bayes	0.6696	0.5235	0.5876	0.8408	0.4765
X-means	0.4631	**0.9865**	0.6303	0.7493	0.0135
EM	0.4851	0.7516	0.6167	0.786	0.2484
Our approach	**0.9929**	0.9552	**0.9737**	**0.9742**	**0.0088**

5 Conclusion and Future Work

In this work, we proposed a novel anomaly detection approach for web applications that leveraging semantic structure knowledge. We proposed approach to learn semantic structure information and built an anomaly detection profile for each type of trivial parameter in three aspects: length model, character distribution model, and structure model. Then we used the detection results from each trivial parameter in URL to classify whether the incoming URL is malicious.

The proposed approach was tested on the CSIC-2010 dataset. Using semantic structure, we achieved 97.42% accuracy and 99.29% precision. And F1-score and recall increased 196.12% and 162.8% than without using semantic structure. Even on the small dataset that only contains 10 records for each type of URL, our approach also archives 88.94% accuracy.

In the future, we intend to research how to learn the changing of semantic structure information with an increment learning mechanism. To provide better services for users, Web-application is constantly evolved, such as adding new or removing old resources and changing the parameters of resources. Thus, the semantic structure information of the Web-application is changing frequently.

References

1. Adhyaru, R.P.: Techniques for attacking web application security. Int. J. Inf. **6**(1/2), (2016)
2. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web services. In: Web Services, pp. 123–149. Springer (2004). https://doi.org/10.1007/978-3-662-10876-5_5
3. Ariu, D., Tronci, R., Giacinto, G.: HMMPayl an intrusion detection system based on hidden Markov models. Comput. Secur. **30**(4), 221–241 (2011)
4. Cho, S., Cha, S.: Sad: web session anomaly detection based on parameter estimation. Comput. Secur. **23**(4), 312–319 (2004)
5. Corona, I., Ariu, D., Giacinto, G.: Hmm-web: a framework for the detection of attacks against web applications. In: 2009 IEEE International Conference on Communications, pp. 1–6. IEEE (2009)
6. Cui, B., He, S., Yao, X., Shi, P.: Malicious URL detection with feature extraction based on machine learning. Int. J. High Perform. Comput. Netw. **12**(2), 166–178 (2018)

7. Cui, M., Hu, S.: Search engine optimization research for website promotion. In: 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, vol. 4, pp. 100–103. IEEE (2011)
8. Denning, D.E.: An intrusion-detection model. IEEE Trans. Software Eng. **2**, 222–232 (1987)
9. Fan, W.K.G.: An adaptive anomaly detection of web-based attacks. In: 2012 7th International Conference on Computer Science & Education (ICCSE), pp. 690–694. IEEE (2012)
10. Fielding, R., et al.: Hypertext transfer protocol-HTTP/1.1. Technical report (1999)
11. Giménez, C.T., Villegas, A.P., Marañón, G.Á.: HTTP data set CSIC 2010. Inf. Secur. Inst. CSIC (Span. Res. Nat. Coun.) (2010)
12. Hawkins, D.M.: Identification of Outliers. Springer, Dordrecht (1980). https://doi.org/10.1007/978-94-015-3994-4
13. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 251–261. ACM (2003)
14. Kruegel, C., Vigna, G., Robertson, W.: A multi-model approach to the detection of web-based attacks. Comput. Netw. **48**(5), 717–738 (2005)
15. Le Jr, D.: An unsupervised learning approach for network and system analysis (2017)
16. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
17. Lei, T., Cai, R., Yang, J.M., Ke, Y., Fan, X., Zhang, L.: A pattern tree-based approach to learning URL normalization rules. In: Proceedings of the 19th International Conference on World Wide Web, pp. 611–620. ACM (2010)
18. Nguyen, H.T., Torrano-Gimenez, C., Alvarez, G., Petrović, S., Franke, K.: Application of the generic feature selection measure in detection of web attacks. In: Herrero, Á., Corchado, E. (eds.) CISIS 2011. LNCS, vol. 6694, pp. 25–32. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21323-6_4
19. Qin, Z.Q., Ma, X.K., Wang, Y.J.: Attentional payload anomaly detector for web applications. In: Cheng, L., Leung, A.C.S., Ozawa, S. (eds.) ICONIP 2018. LNCS, vol. 11304, pp. 588–599. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04212-7_52
20. Shi, J., Cao, Y., Zhao, X.J.: Research on SEO strategies of university journal websites. In: The 2nd International Conference on Information Science and Engineering, pp. 3060–3063. IEEE (2010)
21. Tang, P., Qiu, W., Huang, Z., Lian, H., Liu, G.: SQL injection behavior mining based deep learning. In: Gan, G., Li, B., Li, X., Wang, S. (eds.) ADMA 2018. LNCS (LNAI), vol. 11323, pp. 445–454. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05090-0_38
22. Torrano-Gimenez, C., Nguyen, H.T., Alvarez, G., Franke, K.: Combining expert knowledge with automatic feature extraction for reliable web attack detection. Secur. Commun. Netw. **8**(16), 2750–2767 (2015)
23. Wang, K., Cretu, G., Stolfo, S.J.: Anomalous payload-based worm detection and signature generation. In: Valdes, A., Zamboni, D. (eds.) RAID 2005. LNCS, vol. 3858, pp. 227–246. Springer, Heidelberg (2006). https://doi.org/10.1007/11663812_12
24. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30143-1_11

25. Yang, Y., Zhang, L., Liu, G., Chen, E.: UPCA: an efficient URL-pattern based algorithm for accurate web page classification. In: 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 1475–1480. IEEE (2015)
26. Yu, Y., Yan, H., Guan, H., Zhou, H.: DeepHTTP: semantics-structure model with attention for anomalous HTTP traffic detection and pattern mining. arXiv preprint arXiv:1810.12751 (2018)
27. Yurcik, W., Barlow, J., Rosendale, J.: Maintaining perspective on who is the enemy in the security systems administration of computer networks. In: In ACM CHI Workshop on System Administrators Are Users, p. 345. ACM Press, November 2003
28. Zhang, J., Zulkernine, M.: Anomaly based network intrusion detection with unsupervised outlier detection. In: 2006 IEEE International Conference on Communications, vol. 5, pp. 2388–2393. IEEE (2006)
29. Zhou, Y., Wang, P.: An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence. Comput. Secur. **82**, 261–269 (2019)