2021 IEEE International Performance, Computing, and Communications Conference (IPCCC) | 978-1-6654-4331-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/IPCCC51483.2021.9679413

ELSV: An Effective Anomaly Detection System from Web Access Logs

Wei Wan
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
wanwei@cnic.cn

Xin Shi
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
shixin@cnic.cn

Jinxia Wei
Computer Network Information Center, Chinese Academy of Sciences
Beijing, China
weijinxia@cnic.cn

Jing Zhao
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
jingzhao@cnic.cn

Chun Long *
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
anquanip@cnic.cn

***Abstract*—As network technologies have been developing rapidly, web applications have been widely used. Meanwhile, more and more web attacks have appeared. These attacks have caused a lot of losses to individuals or organizations. A large number of web access logs also bring huge challenges to the attack detection. In this paper, we propose a system ELSV(Ensemble Learning classification with Semantic Vectorization) to detect anomaly web access logs. We use the difference of word occurrence between different types of web access logs to refine the formatted log data, strengthening the distinction between them. The Word2Vec method and TextCNN model are applied to extract vectorized features, making feature extraction more automated and intelligent. We use the ensemble learning classification algorithm as the final classifier, improving the overall classification performance. ELSV is tested on the HTTP DATASET CSIC 2010. The final experimental results show that ELSV achieved 98.48% F1-score and 99.74% AUC, proving the efficacy of it.**

Keywords*—*intrusion detection, semantic refinement, vectorized features, ensemble learning

I. Introduction

With the rapid development of Internet technology and computer technology, Web application has been widely used and developed rapidly because of its advantages such as easy operation, no downloading, available at any time, and no active upgrades, and the related security issues become more and more serious. Attackers can use web attacks to steal important data such as user basic information, passwords, etc., modify the homepage information to achieve certain malicious purposes, and even obtain server permissions to control the server. Common web attacks mainly include Injection, Broken Authentication, Sensitive Data Exposure, XML External Entities (XXE), Broken Access Control, etc [1].

In network operations, a large number of network access logs are generated every day. For example, according to CSTCERT, the Emergency Response Technical Team of Chinese Academy of Sciences, the main website of Chinese Academy of Sciences (www.cas.cn) generates approximately 6M lines of web access logs per day. Traditional manual review is unrealistic. In this paper, we perform anomaly detection for web attacks by proposing ELSV. We use HTTP DATASET CSIC 2010 to verify our proposed system. We first use the semantic information in the web access logs to refine the formatted log data, then use the Word2Vec method and the TextCNN model to extract the vectorized features, and finally input the vectorized features into the ensemble learning detection algorithm to detect whether it is anomaly.

The main contributions of this paper are summarized as follows:

- To obtain more distinguishable data, we refine web access logs based on the semantic dimensions of different types of web access logs.
- To extract features without the need for expert knowledge, we use a combination of the Word2Vec method and the text-cnn model to automatically extract vectorized features.
- To improve the comprehensive classification performance, we use LightGBM, CatBoost, and XGBoost classification algorithms as the final classification.

The rest of this paper is organized as follows. In Section 2, we discuss the related work. The background technology is introduced in Section 3. The proposed system is described in Section 4. The reports of the experiments which are conducted to evaluate the proposed system are reported in Section 5. Finally, the conclusions and the future work are presented in Section 6.

II. Related Work

A. Feature Selection and Extraction

In the process of intrusion detection, the original network traffic needs to be digitally processed before it can be input into the classification algorithm. The quality of the preprocessed input data can directly affect the final classification result. Therefore, the selection and extraction of high quality features in the original network traffic data is a topic of great concern to researchers. There are many publicly available datasets, such as NSL-KDD [3], UNSW-NB15 [4], which use expert knowledge in related fields and other methods to extract data from network traffic. The process of selecting and extracting these features requires a lot of manpower and material resources. How to automate and intelligentize this process is a topic worthy of further study.

Angiulli F et al. [5] proposed an anomaly based intrusion detection method that uses n-grams technology and a novel preprocessing step to analyze the payload of network data packets to construct relevant features without expert domain knowledge. Mimura M et al. [6] proposed a practical detection method based on linguistics, using the TF-IDF model to extract important words, and then constructing a Doc2Vec model to convert the words into feature vectors. Applying natural language processing technology to intrusion detection makes selecting and extracting features from network data packets more automated and intelligent, saving human and material resources.

B. *Web Attack Anomaly Detection*

The anomaly detection of web attacks has also become a hot topic, and there have been many anomaly detection methods for web attacks. Mac H et al. [7] used autoencoders to detect network attacks. The automatic encoder can operate on the original data, so there is no need to extract handmade features. The autoencoder receives the input tokenization request. It then classifies whether the request is malicious. Zhang M et al. [8] used Convolutional Neural Networks (CNN) when processing HTTP requests. They used a specially designed Convolutional Neural Network to automatically extract features for HTTP requests, and classified HTTP requests into normal or anomaly categories through softmax. Kuang X et al. [9] proposed a new network application firewall, DeepWAF. They systematically discussed the effective use of the currently popular CNN models, LSTM models and their combined models CNN-LSTM as well as LSTM-CNN.

Our goal is to build a model that performs well in all aspects, but it is actually not practical since only an ungenereailzed model that performs well in certain aspects (weakly supervised model) can be obtained easily. To overcome this problem, ensemble learning can combine multiple weakly supervised models to get a more comprehensive strongly supervised model. The idea of ensemble learning was first proposed by Dasarathy BV et al. [10]. It has been developed rapidly since the 1990s. Commonly used ensemble learning methods include Bagging [11] method, Boosting [12] method and Stacking [13] method. Based on the Bagging method, Breiman L et al. [14] proposed the random forest algorithm, which has been widely used in subsequent research. Using the Boosting method, Freund Y et al. [15] proposed an adaptive enhancement (Adaboost) algorithm that focused more on the error samples in the training process. Chen T et al. [16] designed a system called XGBoost, which used technologies including cache access, data compression, and realized the use of the least resources to solve real world problems. G. Ke et al. [17] proposed the LightGBM algorithm, which improved training speed while maintaining accuracy. The Catboost method [18] also had a great improvement in training speed, and solved the problem of prediction bias with the process of implementing gradient boosting.

III. Background Technique

A. *NLP Technique*

Word2Vec: In natural language, words are the most fine-grained. To deal with natural language problems, the first work is to deal with words. Words are in symbolic form (Chinese, English, Latin, etc.), so they need to be converted into numerical form, that is, embedded in a mathematical space. This embedding method is called word embedding. Word2Vec [19] is a kind of word embedding. Word embedding uses a shallow neural network that contains only one hidden layer, which takes a large text corpus as input, and finally generates a vector space in which each unique word in the corpus is assigned a corresponding vector.

TextCNN: Convolutional neural network (CNN) [20] adopts the method of local connection and sharing weights, making the network easy to optimize and reducing the risk of overfitting. Yoon Kim [21] combined CNN and NLP to design the TextCNN model, which was modified on a simple CNN model and achieved excellent results in tasks such as sentiment analysis and problem classification. The text is one-dimensional data, so one-dimensional convolution is used in TextCNN. Multiple kernels of different sizes are used to extract the key information in the sentence, so as to better capture the local correlation.

B. *Ensemble Learning*

XGBoost: XGBoost [16] is one of the boosting algorithms. The idea of Boosting algorithm is to integrate many weak classifiers to form a strong classifier. Because XGBoost is a boosted tree model, it integrates many tree models to form a strong classifier. The tree model used is the CART regression tree model. The idea of the algorithm is to continuously add trees and perform feature splitting to grow a tree. Each time a tree is added, it is actually learning a new function to fit the residual of the last prediction.

LightGBM: The LightGBM [17] algorithm is a gradient boosting framework that uses a decision tree based on a learning algorithm. lightGBM contains two key points: (1) light is lightweight, and (2) GBM is gradient booster. LightGBM uses a decision tree algorithm based on Histogram and a leaf-wise leaf growth strategy with depth limitation, which improves the accuracy of prediction, greatly speeds up prediction, and reduces memory utilization.

CatBoost: CatBoost [18] is a GBDT framework based on a symmetric decision tree that implements fewer parameters and supports categorical variables. The main improvement is to process categorical features efficiently and reasonably. As you can see from its name, CatBoost is composed of Categorical and Boosting. In addition, CatBoost also solves the problems of gradient deviation and prediction offset, thereby reducing the occurrence of overfitting and improving the accuracy and generalization ability of the algorithm.

IV. PREPOSED SYSTEM

ELSV includes three parts: data preprocessing, extracting vectorized features, and boosting classification algorithm. The work process of proposed anomaly detection system, ELSV, is shown in Fig. 1. Next, we will introduce ELSV in detail:

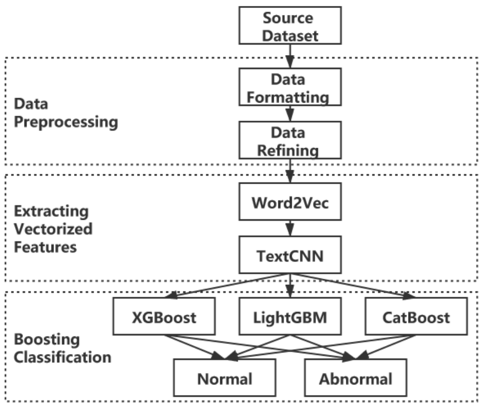


Fig. 1. The Flow Diagram of ELSV

A. *Data Preprocessing*

(I) Data Formatting:

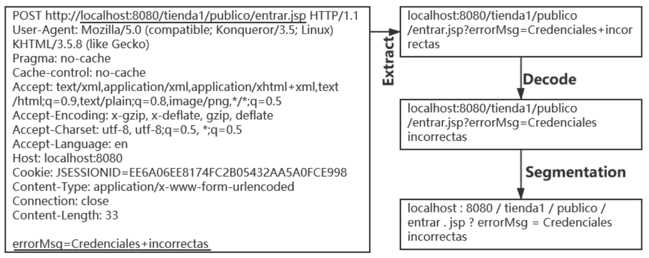
Each request in the dataset we use contains a lot of text information, which needs to be formatted so as to be used as input to the process of extracting vectorized features. The specific data formatting steps are as follows:

Step 1. Extract the host, path, and request body(the underlined part in Fig. 2) from the HTTP request. Use "?" to connect the path and the request body.

Step 2. Decode the extracted data. Since the "+" in the URL represents a space, the "+" is replaced by a space.

Step 3. Use special characters such as "/", "?", "%", "#", "&", "=" to segment the decoded data.

The above realization process is shown in Fig. 2.



```
POST http://localhost:8080/tienda1/publico/entrar.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux)
KHTML/3.5.8 (like Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml+xml,text
/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=EE6A06EE8174FC2B05432AA5A0FCE998
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 33

errorMsg=Credenciales+incorrectas
```

```
localhost:8080/tienda1/publico
/entrar.jsp?errorMsg=Credenciales+incor
rectas
```

```
localhost:8080/tienda1/publico
/entrar.jsp?errorMsg=Credenciales
incorrectas
```

```
localhost : 8080 / tienda1 / publico /
entrar . jsp ? errorMsg = Credenciales
incorrectas
```

Fig. 2. Data Formatting Process: Extract->Decode->Segmentation

(II) Data Refining:

Wu Zhendong et al. [22] proposed that normal network traffic and intrusion network traffic are significantly different in several semantic dimensions, although intrusion traffic is becoming more and more concealed. The semantics of network traffic are re-encoded through deep learning technology, which increases the ability to distinguish traffic and enhances the generalization ability of the algorithm.

Based on this idea, we refine the words in the formatted log data according to the number of times the words appear in the normal or anomaly logs. Simply comparing the number of occurrences of a word in normal or anomaly situations cannot reasonably reflect the semantic information of the word. In view of this, divide the number of times the word appears in normal or anomaly logs by the proportion of normal or anomaly logs in all logs. Take the absolute value of the difference between the two results, and set the threshold λ. When the above result is less than the threshold λ, the word is included in the stop word list, the formula is as follows:

$$\left| \frac{n[word]}{N\ /\ D} - \frac{a[word]}{A\ /\ D} \right| < \lambda \tag{1}$$

where $n[word]$ is the number of occurrences in normal requests, $a[word]$ is the number of occurrences in abnormal requests, N is the number of normal requests, A is the number of abnormal requests, and D is the number of all requests. The words in the stop word list will not be considered in the subsequent process, so as to realize the refining and purification of the data. Taking into account the obvious difference between yes and no, words that have only appeared in normal or anomaly logs will not be included in the stop word list. See Algorithm 1 for specific implementation.

Alghrithm 1 Stop Word List Creation

Input: word corpus C, formatted data set D, threshold λ
Output: stop word list SW[]

```
 1: for y in D:
 2:     N += 1 if y == 0
 3:     A += 1 if y == 1
 4: end for
 5: for sentence in D:
 6:     for word in sentence:
 7:         if word in C:
 8:             n[word] += 1 if y_sentence == 0
 9:             a[word] += 1 if y_sentence == 1
10:         end if
11:     end for
12: end for
13: for word in C:
14:     if |n[word] / (N / len(D)) - a[word] / (A / len(D))|<λ
15:         Put word in SW[] if n[word] != 0 and a[word]!=0
16:     end if
17: end for
```

B. *Extract Vectorized Features*

After the above steps, each HTTP request becomes a ‘sentence’ composed of different ‘words’. To enable it to be input into the classification algorithm, vectorization is needed. We first use Word2Vec to vectorize each ‘word’, and then use TextCNN to vectorize each ‘sentence’. The specific processing flow is shown in Fig. 3. The softmax classification in TextCNN is not used, but substituted with the ensemble learning method for classification. The specific classification method will be introduced in the next chapter.

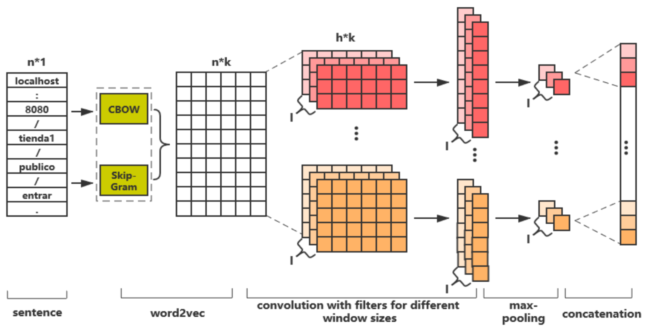


Fig. 3. The Process of Extracting Vectorized Features

C. Boosting Classification Algorithm

As introduced in related work, many excellent classification algorithms have emerged in the field of ensemble learning, as shown in Fig. 4. We use XGBoost, LightGBM, and CatBoost in boosting as classification algorithms to detect normal and anomaly logs. The specific classification process of this paper is shown in Algorithm 2.

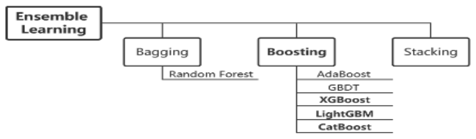


Fig. 4. Different Classification Algorithms in Ensemble Learning

Algorithm 2. Classification

Input: train data *x*, train label *y*, test data *x_text*, test label *y_test*, classification *type*

Output: *result()*, return classification performance, include accuracy, precision, recall, f1_score

```
 1: if type == 'CatBoost':
 2:     model = CatBoostClassifier()
 3:     model.fit(x, y)
 4: end if
 5: if type == 'XGBoost':
 6:     x = xgboost.DMatrix(x, y)
 7:     x_test = xgboost.DMatrix(x_test)
 8:     model = xgboost.train(x_train)
 9: end if
10: if type == 'LightBGM':
11:     t_x, v_x, t_y, v_y = train_test_split(x, y)
12:     train = lightgbm.Dataset(t_x, t_y)
13:     eval = lightgbm.Dataset(v_x, v_y)
14:     model = lightgbm.train(train, valied_sets=eval)
15: end if
16: y_pred = model.predict(x_test)
17: for i in range(0, len(y_pred)):
18:     y_pred[i] == 0 if y_pred[i] < 0.5 else 1
19: end for
20: return result(y_pred, y_test)
```

V. Evaluation

In order to evaluate the capabilities of ELSV in anomaly detection, we conducted several experiments. (1) Using different Word2Vec methods; (2) using different thresholds λ; (3) using different processing methods; (4) comparing with other methods. The experiments were conducted on a 2.3 GHz Intel Core i7 with 16GB RAM running on Windows 10 operating system. Python 3.7.3 was used as the programming language.

A. DataSet

The HTTP DATASET CSIC 2010 dataset [23] is produced by the Institute of Information Security of the Spanish Research Council (CSIC). It contains tens of thousands of automatically generated web access logs and is mainly used for testing the web attack protection system. The dataset includes various attacks, such as: SQL injection, buffer overflow, file disclosure, CRLF injection, cross-site scripting, etc. We integrate all normal requests and anomaly requests in this dataset, and select 80% of them as the training set and 20% as the testing set.

B. Evaluate Performance Indicators

In order to comprehensively evaluate this system, we use accuracy, precision, recall, and f1-score to evaluate the classification results. The specific formula is as follows:

$$accuracy(ACC) = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$precision(P) = \frac{TP}{TP + FP} \tag{3}$$

$$recall(R) = \frac{TP}{TP + FN} \tag{4}$$

$$f1 - score(F1) = \frac{2 * precision * recall}{precision + recall} \tag{5}$$

where *TP* (True Positive) is the number of anomaly logs that are predicted correctly, *TN* (True Negative) is the number of normal logs that are predicted correctly, *FP* (False Positive) is the number of normal logs that are predicted incorrectly, and *FN* (False Negative) is the number of anomaly logs that are predicted incorrectly.

C. Experimental Results

TABLE I. TEXTCNN MODEL PARAMETERS

Description	Value
Input Word Vectors	gensim.models.Word2Vec
Filter Region Size	(3,4,5)
Feature Maps for Each Filter	100
Pooling	1-Max Pooling
Batch Size	16
Epoch	10
Dropout Rate	0.5

(I) Different Word2Vec Methods:

Refer to previous work [28], we built a basic TextCNN model. The relevant parameters are shown in Table Ⅰ. We used the CBOW and Skip-gram methods in the gensim library to generate word vectors. The word vector has a dimension of 128, and each request has a fixed length of 100, padded with '0' representing null in Unicode for the portion insufficient in length. Skip-gram uses the central word to predict surrounding words. When each word is used as the central word, it has to be predicted and adjusted many times. When the amount of data is small or there are a large number of low frequency words in the corpus, the generated word vector will be better than CBOW. It can also be seen from the experimental results in Table Ⅱ that the effect of using Skip-gram is better than that of CBOW.

TABLE II. THE PREFORMANCE OF DIFFERENT WORD2VEC METHODS

Classifier	Word2Vec Method	accuracy	precision	recall	f1-score
XGBoost	CBOW	0.9867	0.9908	0.9576	0.9739
	Skip-gram	**0.9933**	**0.9986**	**0.9756**	**0.9870**
LightGBM	CBOW	0.9863	0.9946	0.9525	0.9731
	Skip-gram	**0.9930**	**0.9992**	**0.9740**	**0.9865**
CatBoost	CBOW	0.9852	0.9925	0.9501	0.9709
	Skip-gram	**0.9918**	**0.9992**	**0.9693**	**0.9840**

(II) Different Threshold λ:

We input the word vector generated by the Skip-Gram method into the TextCNN model, adjust the value of the threshold λ, and compare the impact of different threshold λ on the classification performance. We select 0、2^0、2^1、2^2、2^3、2^4、2^5、2^6、2^7、2^8 as the threshold λ, and use XGBoost, LightGBM or CatBoost as the classifier. It can be seen from Table Ⅲ that when λ=128, the accuracy and f1-score using XGBoost or LightGBM as the classifier have reached the optimal values, where the accuracy is 0.9933 and 0.9930, and f1-score is 0.9870 and 0.9865; when CatBoost is used as the classifier, the accuracy of 0.9924 and the f1-score of 0.9851 are obtained when λ=32. Through the comparison of the results, the best comprehensive performance is achieved when λ=128 and XGBoost is used as the classifier. Fig. 5 shows the change trend of accuracy and f1-score of the above three classifiers when λ takes different values.

TABLE III. THE ACCURACY AND F1-SCORE OF DIFFERENT THRESHOLD Λ WITH XGBOOST, LIGHTGBM AND CATBOOST

λ	XGBoost		LightGBM		CatBoost	
	accuracy	*f1-score*	*accuracy*	*f1-score*	*accuracy*	*f1-score*
0	0.9908	0.9820	0.9910	0.9824	0.9907	0.9818
2^0	0.9912	0.9829	0.9912	0.9828	0.9910	0.9824
2^1	0.9913	0.9830	0.9914	0.9832	0.9912	0.9829
2^2	0.9922	0.9848	0.9922	0.9847	0.9923	0.9849
2^3	0.9919	0.9841	0.9922	0.9849	0.9921	0.9845
2^4	0.9923	0.9850	0.9926	0.9855	0.9915	0.9843
2^5	0.9924	0.9852	0.9924	0.9852	**0.9924**	**0.9851**
2^6	0.9925	0.9854	0.9926	0.9855	0.9916	0.9836
2^7	**0.9933**	**0.9870**	**0.9930**	**0.9865**	0.9918	0.9840
2^8	0.9920	0.9844	0.9921	0.9846	0.9916	0.9836

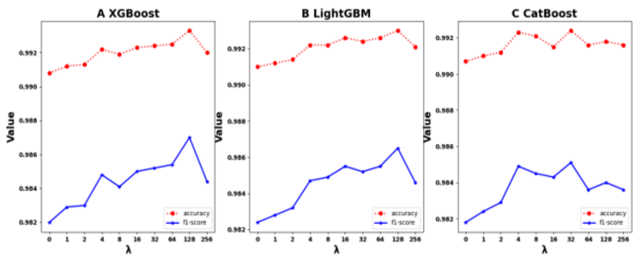


Fig. 5. The Changing Trend of Different Threshold λ with XGBoost, LightGBM and CatBoost

(III) Comparison of Different Processing Methods:

We use λ=0 (without using the threshold λ) and directly use Softmax to classify in the TextCNN model as the baseline, and compare with the processing method of λ=128. When λ=128, GBDT, LightGBM, CatBoost, and XGBoost are respectively used to replace Softmax to classify normal and anomaly logs (directly use the vectorized features extracted from the TextCNN model as the input of the classification algorithm). As shown in Table Ⅳ, compared with Softmax and GBDT, the use of LightGBM, CatBoost, XGBoost has greatly improved the comprehensive classification performance, and the use of XGBoost as the classifier has achieved the best comprehensive classification performance. Its f1-score is 0.9870, which is an increase of 0.0150 compared to 0.9720 using Softmax, and an increase of 0.0182 compared to 0.9688 using Softmax when λ=0 (Baseline).

The left picture in Fig. 6 is the ROC curve of the above case, and the right one is the curve after narrowing the y-axis range to show a more obvious distinction. It can be seen that the best classification performance is achieved when λ=128 and XGBoost is used as the classifier. We calculated the AUC values of these processing methods and compared them in Fig. 7. It can be seen that when λ=128 and XGBoost is used as the classifier, the highest AUC value 0.9973 is achieved. which is 0.0024 higher than 0.9949 of the base line (not using the threshold λ and directly using Softmax for classification).

TABLE IV. THE PERFORMANCE OF DIFFERENT PROCESSING METHODS

Processing Method	accuracy	precision	recall	f1-score	AUC
Softmax(Base)	0.9843	1.0	0.9396	0.9688	0.9949
λ-Softmax	0.9858	**1.0**	0.9455	0.9720	0.9955
λ-GBDT	0.9852	0.9967	0.9461	0.9707	0.9960
λ-LightGBM	0.9930	0.9992	0.9740	0.9865	0.9962
λ-CatBoost	0.9918	0.9992	0.9693	0.9840	0.9971
λ-XGBoost	**0.9933**	0.9986	**0.9756**	**0.9870**	**0.9973**

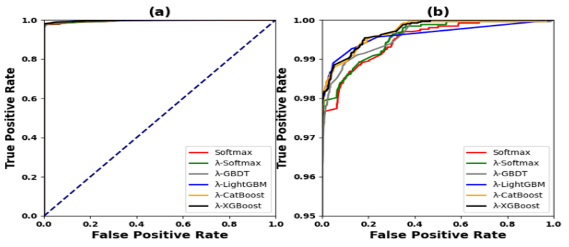


Fig. 6. (a) is the ROC Curve of Different Processing Methods. (b) is the ROC Curve after Narrowing the Y-axis Range

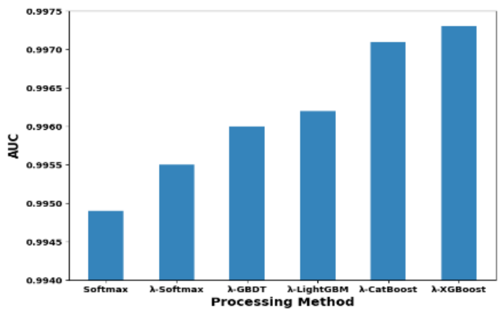


Fig. 7. The AUC Values of Different Processing Methods

(IV) Comparison with Other Methods:

We compared ELSV with other methods. As shown in Table Ⅴ, Mac H et al. [7] achieved 0.9464 on precision, 0.9462 on recall, 0.9463 on f1-score; J. Zhang M et al. [8] achieved 0.9649 on accuracy, 0.9335 on recall; and Kuang X et al. [9] achieved 0.9726 on accuracy, 0.9771 on precision, 0.9549 on recall and 0.9659 on f1-score. It can be clearly seen that ELSV has achieved the best results in all aspects.

TABLE V. COMPARISON OF CLASSIFICATION PERFORMANCE OF DIFFERENT RESEARCH

Method	accuracy	precision	recall	f1-score
Mac H et al. [7]	\	0.9464	0.9462	0.9463
Zhang M et al. [8]	0.9649	\	0.9335	\
Kuang X et al. [9]	0.9726	0.9771	0.9549	0.9659
ELSV	**0.9933**	**0.9986**	**0.9756**	**0.9870**

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed ELSV, an effective anomaly detection system for detecting web attacks over web access logs. The formatted log data is refined according to the difference of word occurrence between normal logs and anomaly logs, which not only simplifies the formatted data, but also improves various performance indicators for downstream classification task. The Word2Vec method and TextCNN model are used in conjunction to extract the vectorized features of web access logs. It does not require expert knowledge for manually extracting the features so that it is more automated and intelligent. The use of ensemble learning classifier for anomaly detection further improves the comprehensive performance indicators of the classification results.

Our results are based on the HTTP DATASET CSIC 2010 dataset, which is only experimental data and relatively old. In future work, we will deploy ELSV in CSTCERT, mainly responsible for the network system operation and maintenance of China Science and Technology Network, to realize the detection of web attacks in the actual environment.

ACKNOWLEDGMENT

The authors would like to thank the editorial board and reviewers. This work was supported by the "14th Five-Year Plan" Internet Information Special Pre-construction Project of the Chinese Academy of Sciences [WX145XQ10], [WX145XQ11].

REFERENCES

[1] OWASP Top 10-2017, [online] Available: https://www.owasp.org/www-pdf-archive/OWASP_Top_10-2017_(en).pdf.pdf.

[2] Danielsson H , Rönnberg, Jerker, Levén, Anna, et al. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content[J]. Faculty of Arts & Sciences, 2014, 30(4):595-599.

[3] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6, IEEE, 2009.

[4] Moustafa N , Slay J . UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)[C]// Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.

[5] Angiulli F , Argento L , Furfaro A . Exploiting n-gram location for intrusion detection[J]. Computer Science, 2014:1093-1098.

[6] Mimura M , Tanaka H . A Linguistic Approach Towards Intrusion Detection in Actual Proxy Logs: 20th International Conference, ICICS 2018, Lille, France, October 29-31, 2018, Proceedings[C]// 2018.

[7] Mac H , Tran D Q , Tran H A . Detecting Attacks on Web Applications using Autoencoder[C]// The Ninth International Symposium on Information and Communication Technology (SoICT 2018). 2018.

[8] Zhang M , Xu B , Bai S , et al. A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN[C]// International Conference on Neural Information Processing. Springer, Cham, 2017.

[9] Kuang X , Zhang M , Li H , et al. DeepWAF: Detecting Web Attacks Based on CNN and LSTM Models[M]// Cyberspace Safety and Security, 11th International Symposium, CSS 2019, Guangzhou, China, December 1–3, 2019, Proceedings, Part II. 2020.

[10] Dasarathy B V , Sheela B V . A composite classifier system design: Concepts and methodology[J]. Proceedings of the IEEE, 1979, 67(5):708-713.

[11] Breiman L . Bagging Predictors[J]. Machine Learning, 1996.

[12] Freund Y , Schapire R E . A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting - ScienceDirect[J]. Journal of Computer and System ences, 1997, 55(1):119-139.

[13] Wolpert D H . Stacked generalization[J]. Neural Networks, 1992, 5(2):241-259.

[14] Breiman L . Random Forests[J]. Machine Learning, 2001, 45(1):5-32.

[15] Freund Y , Schapire R E . A desicion-theoretic generalization of on-line learning and an application to boosting[J]. Journal of Computer and System ences, 1995, 55:119-139.

[16] Chen T , Guestrin C . XGBoost: A Scalable Tree Boosting System[J]. 2016.

[17] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 3146-3154.

[18] orogush A V , Ershov V , Gulin A . CatBoost: gradient boosting with categorical features support[J]. 2018.

[19] Mikolov T , Chen K , Corrado G , et al. Efficient Estimation of Word Representations in Vector Space[J]. Computer Science, 2013.

[20] Lecun Y , Bottou L . Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[21] Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[22] Wu Zhendong, Wang Jingjing, Hu Liqing, et al. A network intrusion detection method based on semantic Re-encoding and deep learning. 2020.

[23] Torrano-Giménez C, Pérez-Villegas A, Álvarez G. HTTP DATASET CSIC 2010; 2010.