Journal of Computers Vol. 31 No. 1, 2020, pp. 242-252
doi:10.3966/199115992020023101022

Webshell Detection Based on Multi-classifier Ensemble Model

Wenjuan-Lian, Qi-FAN, Dandan-Shi, Qili-An, Bin Jia*

College of Computer Science&Engineering, Shandong University of Science and Technology, Qingdao, Shandong Province, China
{Wenjuan.lian, jiab_sdust}@126.com
{1261186378, 2465704675, 410287883}@qq.com

Received 13 January 2019; Revised 4 March 2019; Accepted 12 March 2019

Abstract. Webshell is a web backdoor which can be used to remotely control web servers by hackers. Due to the continuous development of escaping technology, Webshell detection has become more and more difficult. Based on the analysis of evolving Webshell detection technology in recent years, we proposed Multi-classifier ensemble Model in this paper. Firstly, effective information of PHP files are extracted including static characters, grammar character and corresponding opcode; secondly, different base classifiers and modified classifier are trained and analyzed for further model, at last the ensemble model based on stacking is proposed and verified. Our dataset collected from multiple GitHub open source projects. The mothod proposed in this paper could ultimately achieve the accuracy of 98.447% and precision of 99.227%, which showed excellent performance compared with mainstream detection tools.

Keywords: AdaBoost, ensemble learning, machine learning, random forest, Webshell

1 Introduction

In recent years, a large number of emerging Internet industries have emerged rapidly. Various web application systems are widely used in social and banking services, which bring convenience to people's work and life. However, the security threats are not to be underestimated. And Webshell is one of them. Webshell is a command execution environment in the form of web files such as asp, php, jsp or cgi. After malicious users invades a Web site, they usually upload the Webshell file to the server and get a command execution environment to control the target Web server. Then they can prepare for subsequent authorization operations. In the 2017, 360 website guardian interception vulnerability statistics TOP 10, Webshell ranks the second place [1]. Although many security companies currently launch security products such as cloud WAF to protection. There are also many researchers use lexical analysis and other technologies to detect Webshell [2]. But as escape technology becomes more and more complex. These methods still have a lot of shortcomings.

In this paper, we study the existing technology in the field of Webshell detection, and propose a detection framework for multi-classifier fusion. By comparing various machine learning models, we finally adopted a dynamic detection method based on the Stacking model. Section 3 introduces the system framework and feature selection. And section 4 introduces the Fusion of Stacking model and multi-classifier, and introduces the principle and calculation of multi-classifier. Section 5 is the experiment and analysis, expounding the whole process from data collection, evaluation indicators to Webshell test results and analysis. Finally, we summarize the work and look forward to the future development of Webshell detection technology.

* Corresponding Author

2 Related Work

There are usually two kinds of Webshell detection method: static detection and dynamic detection. Feature based file code detection is one of the static detection. After Webshell oprates, static method detects the bytecode of Webshell and its communication is dynamic detection. Based on the detection of static features, this file does not need to be executed, detect Webshell only from the file code side. Static feature detection is the most basic and most common method. In terms of static detection, Zheng et al. analyzed the realization mechanism of Linux Webshell, described the common characteristics and the characteristic mixed method. On this basis, a detection method based on SVM classifier is put forward and realized [3]. Fei et al. analyzed the HTML feature of Webshell pages, proposed a black box detecting method based on support vector machine (SVM) classification algorithm. The method is one sort of supervised machine learning system which can detect unknown Webshell without the knowledge of source code [4]. Cui et al. proposed a Webshell detection method based on XGBoost algorithm. The proposed method improved the problem of lack of Webshell feature coverage by using the characteristics of Webshell statistic [5]. Log analysis as a means of forensics and prediction to detect Webshell. Shi et al. based on the analysis of the server log text file, and the Webshell is detected from the three angles: text feature, statistical features and correlation feature [6].

In recent years, due to the development of machine learning, it is more and more widely applied to the field of Webshell detection. Dai et al. supervised machine learning algorithm was put forward to detect Webshell intelligently. By learning the features of existing Webshell and non-existing Webshell pages, the algorithm can make prediction of the unknown pages [7]. There are still many applications for machine learning. To improve the Webshell detection feature coverage and the ability of detection algorithm, Jia et al. proposed a Webshell detection method based on random forest improved algorithm, which improved the method of random forest feature selection and analyzed features of three kinds Webshell, and built multi-dimensional features which had comprehensive coverage of static attributes and dynamic behaviors [8]. Khan MS et al. developed a new cognitive host based anomaly detection system based on supervised AdaBoost machine learning algorithm. Information fractal dimension based approach is incorporated in the original AdaBoost machine learning algorithm to assign higher weight to the classifier that estimates wrong hypothesis [9].

In the application of neural networks, Qi et al. proposed a Webshell detection method based on Multi-Layer Perception (MLP) neural network. In this method, they used TF-IDF to calculate the word frequency matrix, and on this basis, the feature matrix of trained sample set is selected. Finally, the detection model is obtained through multi-layer neural network training. The experimental results indicate that the detection model can handle unknown and variant samples well [10]. In multi-model analysis, Cui et al. used the common statistical features of PHP source files and extracted opcode sequence features from PHP source files, proposed a PHP Webshell detecting model, the RF-GBDT (Random Forest-Gradient Boosting Decision Tree) model, which is the combination of random forest classifier and GBDT classifier. Opcode sequence features and statistical features such as information entropy, index of coincidence and so forth, carry out TF-IDF vector and hash vector [11]. Firdaus et al. proposed to use the bio-inspired method of practical swarm optimization (PSO) and also adopted boosting (adaboost, realadaboost, logitboost, and multiboost) to enhance the machine learning prediction that detects unknown root exploit [12].

In addition, there are many other detection techniques, such as Webshell detection method based on correlation analysis [13], semantics-based Webshell detection method [14]. Tu et al. proposed a novel method based on the optimal threshold values to identify files that contain malicious codes from web applications [15], Multi-strategy based framework for Webshell detection [16] and so on.

3 System Framework

The Webshell detection model proposed in this paper is shown in Fig. 1. First, in the data preprocessing, we extracted the TF-IDF vector 1 by removing the annotations and code blocks in the data set and extracted feature based on the word bag n-gram and TF-IDF techniques. We also encode the sample to obtain opcode, and get the TF-IDF vector 2 based on N-gram participle and TF-IDF vectorization. Then,

we chose several machine learning algorithms to test the vector and integrated the better classifier. The detailed information will be discussed in Section 4.

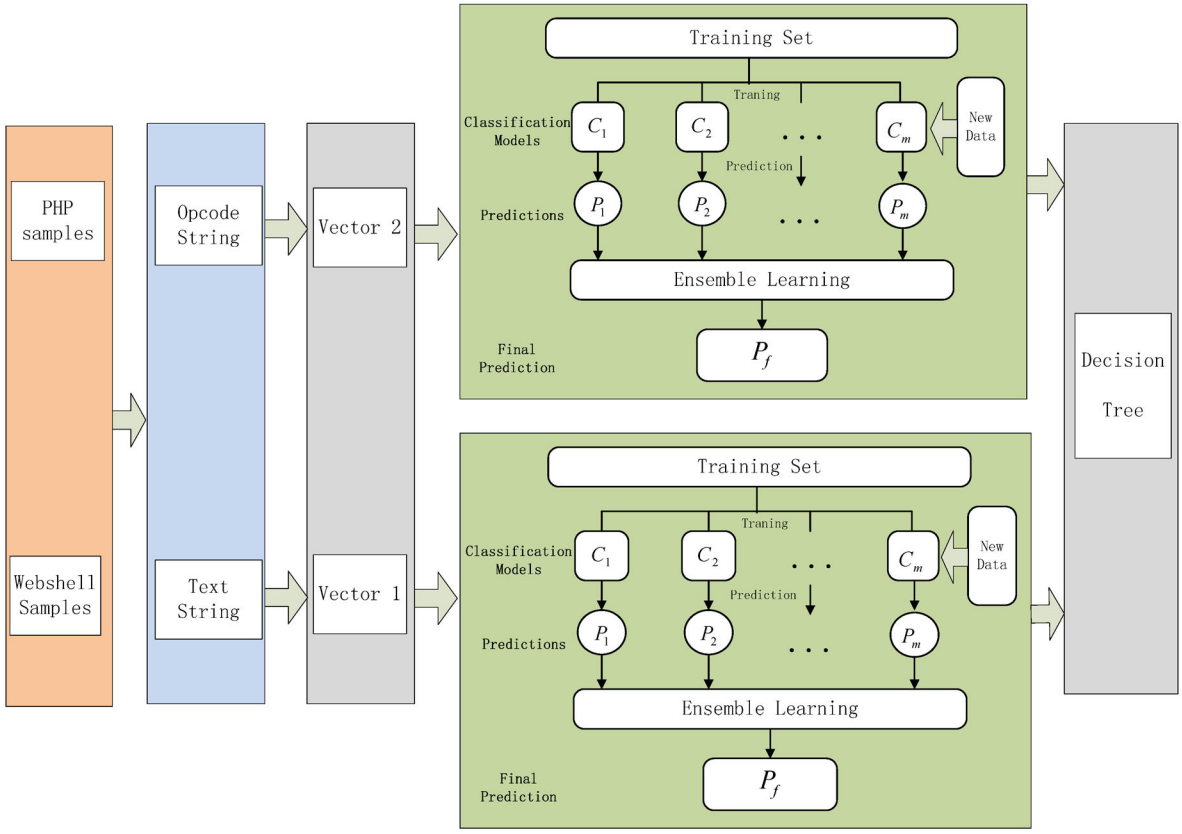

Fig. 1. Webshell detection framework

3.1 Static Characters Analysis

Webshell cannot implement certain functions without some specific keywords and sensitive functions. Static detection makes the rules of web files according to their characteristics and apply to Webshell detection. Each detection method corresponds to one function, and all functions (ie detection methods) are located in the same file, which is imported into the main file as a whole module. It can be detected by scanning a specific directory. Unencrypted Webshell can be detected based on sensitive key-words such as control-u, phpspy, Webshell, CMD. Exe, evel() and so on. The Webshell file encrypted by some encrypted websites or tools such as php shield, may contain relevant encryption information.

In addition to detecting specific keywords, there are some features of known shell files, such as " r57shell". In addition to regular expression matching for sensitive functions of eval, assert, etc., for example [\'"]e[\'"]\.[\'"]v[\'"]\.[\'"]a [\'"]\.[\'"]l[\'"]. And there will also be a combination feature search. If the keywords such as “cmd.exe” and “program files” appear simultaneously in the file content, it can be basically judged that this file is a shell file containing sensitive keywords.

Some Webshells have specific features, for example packaging all the files on the website to obtain the website source code for auditing. Webshell for packaging files usually uses specific functions, for example the gzencode, gzdeflate and gzcompress functions for compressing files in PHP. In addition to the compression function, there will be some sensitive keywords appearing in the shell file, such as “package, unix2DosTime”.

However, the attacker encrypts the sensitive features in the string, reversibly maps them to another string, and decrypts them when they are accessed. This escape technology is widely used in various kill-free webshells. Take the PHP language as an example, and built-in encryption methods such as base64 and rot13. In order to better resist detection, some malicious programs use a custom encryption function to escape. After using regular replacement and base64 encoding, the sensitive features of malicious files are well hidden.

For the encryption technology, scholars proposed to use the index of coincidence in cryptography for analysis, combined with information entropy to judge. Information entropy referring to the disorder and uncertainty of given information by measuring the average uncertainty of all possible sources of information. The entropy is higher, the information is more disordered. Generally, a Webshell is encrypted, encoded and introduced some random strings to achieve the goal of obfuscation and thus it is of high perplexity which will lead to a high information entropy. Hence, we can better identify Webshell by virtue of the value of information entropy. Index of coincidence, also called IC, which can be leveraged to evaluate the probability of finding two same letters by randomly selecting two letters from a given text. Meanwhile, Webshell tends to possess a relatively lower IC than a normal file does, in that the randomness of a text will be improved after encryption. This method can detect the effect to a certain extent, but for the purpose of protecting intellectual property rights and preventing source code leakage, many normal applications also use encryption technology to process the code. In this case, there is a possibility of false positives for the detection based on the above means.

3.2 Opcode and Other Features Extraction

Grammatical semantic analysis is based on the implementation of php language scan and compilation, stripping code, comments, analysis variables, functions, strings, language structure analysis methods to achieve the capture of key dangerous functions. This can perfectly solve the situation of underreporting. However, there are still problems in the false positives. The problem of false positives is that the detected file is a legal PHP grammar file. It is not verified whether it is a legitimate PHP grammar file. It is only scanned and analyzed.

Our approach is based first on grammar analysis, stripping tokens, comments, strings, variables, language constructs, treating PHP files as a complete string, using the most common word bag model, and extracting the word bag model using 2-gram. And processing is performed using the TF-IDF model to obtain the TF-IDF vector 1. Then PHP syntax detection, extract opcode code to solve the missing report problem.

Opcode is part of a computer instruction that specifies the operation to be performed. The format and specification of the instruction are specified by the processor's instruction specification. In addition to the operands required by the instruction itself, there may be instructions that do not require the operands to be displayed. These operands may be values in registers, values in the stack, values in a block of memory, or values in an I/O port. Usually opcode has another name---byte codes. For example, the Java Virtual Machine (JVM) and the .NET Common Intermeditate Language (CIL). The opcode in PHP belongs to the latter in the introduction.

In this paper, we read the directory where the webshell sample and the normal php file are saved, and load the corresponding opcode string, where the mark webs hell is 1, the normal php file is 0; use 2-Gram to process the opcode string, using the TF-IDF model is further processed to obtain the TF-IDF vector.

We parse the PHP file into opcode and get the opcode sequence. For the convenience, we intercept only a fixed length of the opcode sequence, and exceed the truncation of a fixed length. we also complement the insufficient by 0. Take a common PHP example:

```
<?php
    Echo $_GET[‘r’];
?>
```

Obtain the corresponding opcode sequence as:

(FETCH_R, FETCH_DTM_R, ECHO, ECHO, RETURN)

4 Dynamic Multiple-classifier Model

4.1 Base Classifier Selection

In this paper, we first use a variety of mainstream machine learning algorithms to detect Webshell, the results are shown in Table 5. Overall, Random Forest has the highest detection rates. Random Forest is a classifier that contains multiple decision trees. Its output results are determined by the output of multiple decision trees. Random Forest first generates training samples and test samples by resampling. Then, the training sample generates a plurality of base classifiers to form a strong classifier for detection. However,

due to the lack of abnormal samples, the traditional single classifier classification method can not achieve high classification results. Therefore, we proposed a Webshell detection method based on RF-AdaBoost multi-classifier fusion.

4.2 Improved Classifier Based on RF-AdaBoost

A single classifier is difficult to achieve better detection results, and a hybrid model that integrates multiple machine learning techniques has attracted more attention from researchers. Xue et al. [17] integrated multiple classifiers of Softmax, XGBoost and Random Forest to calculate the probability per pixel class. Finally, the fully connected conditional random fields (CRF) is used to enhance the final performance; Li et al. [18] used the gradient boosted decision tree (GBDT) algorithm to detect network attacks and used particle swarm optimization (PSO) algorithm to optimize GBDT parameters. Vajdi et al. [19] classified installed applications based on AdaBoost and J48 hybrid models to make the average true positive rate increase from 98.9% to 99.2% (with gain ratio). This paper proposes a detection method for RF-AdaBoost multi-classifier fusion.

4.2.1 Principle of RF-AdaBoost

Random forest is an extended variant of Bagging. As a parallel working method, random forest is based on decision tree-based classifier to, and further introduces random attribute selection in decision tree training process based on building Bagging integration with decision tree based classifier. AdaBoost is an adaptive Boosting algorithm that effectively copes with over-fitting, which is an algorithm suitable for various classification occasions. In this paper, M subsets were firstly randomly extracted from feature vector 1, which then were used to train RF-based classifier. Finally, the prediction results of these base classifiers were integrated by voting method, and the initial prediction result 1 was output. Then, the initial prediction result 2 was obtained by the same operation for feature vector 2. Then this paper used the AdaBoost algorithm to train the base classifiers with the obtained initial prediction result to obtain the prediction result, and weighted the classification error data to obtain a new training set to conduct a new round training for base classifiers. The step was cycled until a predetermined minimum error rate or maximum number of iterations was reached. Finally, the base classifier with good classification effect was increased in weight, and those with bad classification effect was decreased in weight, which were combined to a strong classifier.

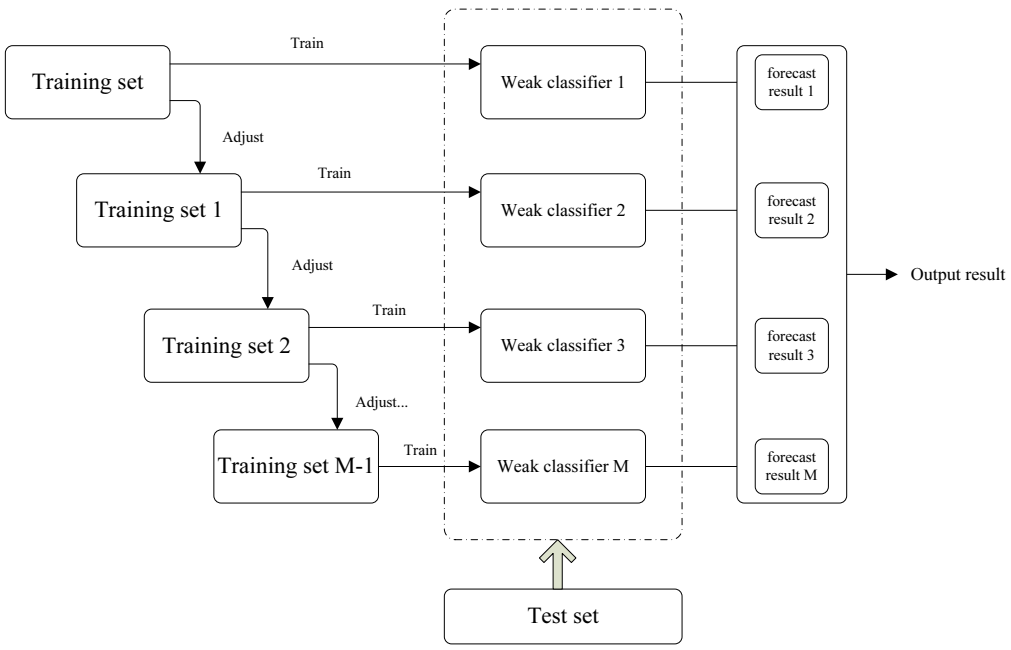

Fig. 2. AdaBoost Mechanism

4.2.2 AdaBoost Algorithm

The RF initial prediction results are combined into a training data set $(x_1, y_1),...,(x_N, y_N)$, where $y_i \in \{1,0\}$. 0 is a normal page, and 1 is WebShell. AdaBoost, which aims to learn a series of weak classifiers or basic classifiers from the training data, and then combine these weak classifiers into a strong classifier. The learning rate is 0.7. The algorithm of AdaBoost is shown in Table 1.

Table 1. AdaBoost Process

(1) Weight distributions of initial training data: of $w_i = \frac{1}{N}$,

$$D_1(i) = (w_1, w_2, ..., w_N) = \left(\frac{1}{N}, ..., \frac{1}{N}\right)$$

(2) To iterate, $t = 1, ..., T$

(a) Select a weak classifier h with the lowest error rate at present as the t-th base classifier H_t, and calculate the weak classifier h_t:

$$h_t : X \to \{1, 0\}$$

(b) The error in the distribution D_1 is:

$$e_t = P(H_t(x_t) \neq y_i) = \sum_{i=1}^{N} w_{ti} I(H_t(x_i) \neq y_i)$$

(c) The weight of the classifier in the final classifier α:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-e_t}{e_t}\right)$$

(d) Update the weight distribution of training samples:

$$D_{t+1} = \frac{D_t(i) \exp(-\alpha_t y_i H(x_i))}{Z_t}$$

(3) Strong classifier:

$$H_{final} = sign(f(x)) = sign\left(\sum_{t=1}^{T} \alpha_t H_t(x)\right)$$

4.3 Dynamic Classifier Construction

Although the RF-AdaBoost model enhances the performance of the Random Forest classifier, it improves the detection rate of Webshell. However, in practical applications, due to the continuous development of escape technology, the detection effect of the RF-AdaBoost model is not good. Webshell is becoming more and more difficult to detect, and the traditional single classifier and mixed multiple classifier methods can't achieve good detection results. Therefore, this paper proposes a detection method that dynamically integrates multiple machine learning algorithms through Stacking technology, which not only enhances the performance of the detection model, but also improves the detection efficiency of Webshell.

Stacking is usually used for the fusion of different types of model individuals, and the training results of different models are fused in some way. Because the non-linear factors of the data itself have been fully exploited, and the final prediction results are obtained by further fusion using the Decision Tree model. In this paper, we present an algorithm for Webshell detection through dynamic selection and integration of multiple machine learning algorithms. In the final stage of the model, the Stacking technique is used to fuse multiple models together, and the predicted probability values of the model are input as feature data. The labels of the initial samples are still used as sample markers, and the linear model is used to assign weights to each model reasonably. The prediction results of multiple classifiers are input into decision tree for final prediction and output.

4.4 Dynamic Classifier Selection

For different features, the detection effects of machine learning algorithms are different. However, in the actual testing process, a model is needed to make each indicator the best. The model proposed in this paper can select the top three classifiers of each indicator according to different characteristics for fusion, so as to ensure that each detection index can achieve the best effect, so as to improve the accuracy of detecting Webshell.

In the process of Webshell detection, the model firstly selects the better classifier as the primary classifier based on the input features. Then, the training data is split and the primary classifier is trained separately. And use the trained primary learning. To obtain the predicted result, the predicted result is used as the training set of the secondary learner, where in the predicted value of the first base model for the first training sample will be the first eigenvalue of the first sample in the new training set, and retained. The original target value; finally, the secondary learner is trained using the results predicted by the primary learner to obtain the final trained model. The algorithm is shown in Table 2.

Table 2. Stacking Algorithm

Input: Training Set $D=\{(x_1,y_1),(x_2,y_2),...,(x_m,y_m)\}$;
Primary learning algorithm $\delta_1,\delta_2,...,\delta_T$;
Secondary learning algorithm δ;
Process:
1. for $t=1,2,...,T$ do
2. $h_t=\delta_t(D)$;
3. end for
4. $D'=\varnothing$;
5. for $i=1,2,...,m$ do
6. for $t=1,2,...,T$ do
7. $z_{it}=h_t(x_i)$;
8. end for
9. $D'=D'\cup((z_{i1},z_{i2},...,z_{iT}),y_i)$;
10. end for
11. $h'=\delta(D')$;
Output: $H(x)=h'(h_1(x),h_2(x),...,h_T(x))$

5 Experiment and Analysis

5.1 Data Sets

The Web shell is a specific use program written in the Web scripting language that provides a way to communicate with server operations. It also can perform OS operations through a web language interpreter. Webshell writing requires expertise or use of specialized tools, such as WebSHArk 1.0 [20]. The sample Webshell used in the experiment comes from several open source Webshell projects in GitHub. The normal samples are from the mainstream CMS source code, as in Table 3. And the deduplication tool is used to delete duplicate samples. After the deduplication, the number of positive samples is 9939, and the number of negative samples is 3478.

Table 3. Statistics Of Collected Samples

Sample	Source
Webshell	https://github.com/xl7dev/WebShell https://github.com/tennc/webshell
CMS	https://wordpress.org http://www.phpcms.cn/ https://www.phpMyAdmin.net/ https://github.com/smarty-php/smarty

5.2 Evaluation Indicators

Table 4 shows a confusion matrix that is used to represent the information related to the actual and predicted classifications performed by a detection model. In the experiment, 1 means Webshell, 0 means normal page.

Table 4. Classification Confusion Matrix

True Class	Predicting Results	
	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (FalsePositive)	TN (True Negative)

In Table 4, TP is the number of Webshells that are correctly detected; TN is the number of normal pages that are correctly identified; FP is the number of normal pages that are incorrectly classified as Webshells; FN is the number of Webshells that are incorrectly judged as normal pages. In this paper, the performance of Webshell detection models is evaluated by four widely used measures: Accuracy, Precision, Recall and F1-score. The calculations of these evaluation measures are defined as follows:

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

Accuracy is the proportion of the total sample that correctly predicts the number of samples.

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

Precision is the correct prediction of the number of Webshells in the proportion of all Webshell in the forecasting sample.

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

Recall is the correct prediction of the number of Webshells in all samples.

$$F1 = \frac{2 \times Precision \times Recall}{Precision \times Recall} \tag{4}$$

F1-score is the harmonic mean of the precision and the recall.

5.3 Experimental Results and Analysis

We resorted to the 10-fold cross validation which is a useful method to evaluate the effectiveness of a classifier. It divides the data set into 10 parts to make 10 rounds of evaluation. In each round, each part will be the testing set and the rest parts will be the training set. Furthermore, based on the 10 rounds of evaluation, we made assessment by drawing ROC (Receiver Operating Characteristic) Curve in our experiment.

A Receiver Operating Characteristic (ROC) curve is normally used to measure effectiveness of the detection method. It indicates how the detection rate changes, as the internal threshold is varied to generate more or fewer false alarms. It plots detection accuracy against false positive probability. Fig. 3 depicts the ROC curve for our approaches.

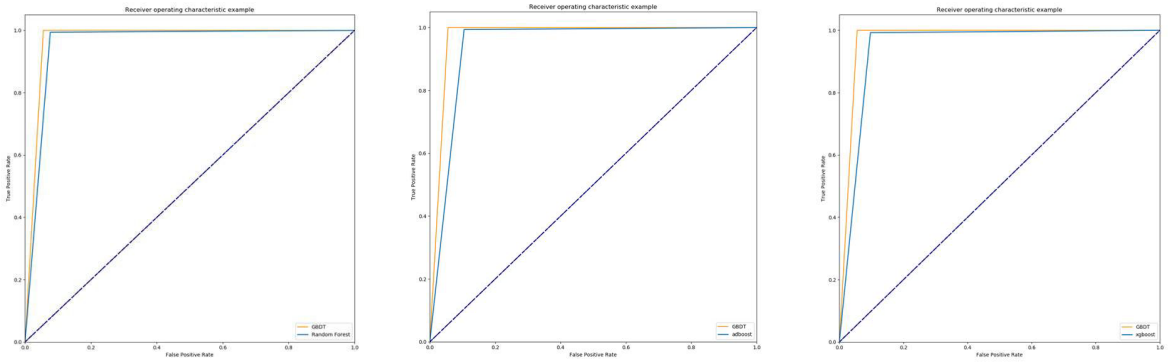

Fig. 3. The ROC curve - false positive rate vs. true positive rate

ROC curves signify the trade off between false positive rate (FPR) and true positive rate (TPR), which means that any increase in the TPR is accompanied by a decrease in the FPR. Then, as shown by the ROC curves, lines in the diagram is the closest to the left-hand border and the top border compared to other diagrams, indicated that it offers the finest result among the other methods. The experimental result in Fig. 3 shows that method based on dynamic model performed the best result.

In this paper, we use different machine learning algorithms to compare different features. They have achieved good accuracy and have very low false positive rates, as in Table 5. This method has some advantages in all aspects of Webshell detection. Our method uses the best performing machine learning algorithm of the two features for integrated learning. The accuracy of ten-fold cross-validation reaches 98.447% with precision of 99.227%, far exceeding most machine learning.

Table 5. Comparison of different machine learning algorithms for different features

Feature	Algorithm	Accuracy	Precision	recall	F1-score	ten-fold Cross validation
TF-IDF vector2	Random forest	0.95816	0.89104	0.95367	0.92130	0.93419
	GDBT	0.95816	0.89458	0.94888	0.92093	0.92960
	Xgboost	0.95324	0.88323	0.94249	0.91190	0.93239
	Adaboost	0.95119	0.87334	0.94728	0.90881	0.94454
TF-IDF vector1	Random forest	0.95309	0.89367	0.96107	0.92614	0.92716
	GDBT	0.94267	0.87126	0.94750	0.90778	0.90526
TF-IDF vector1 & TF-IDF vector2	This Method	0.98447	0.99227	0.98972	0.99099	-

In order to compare our model with other systems, we downloaded several popular mainstream webshell detectors from the Internet, including 360, Avast, Webshellkiller, D shield, FindWebshell and so on. The results showed that some detectors had fine performance on the rate of recall, such as D Shield and AVAST, but most of them could not reach high accuracy. The method we proposed in this paper could achieve the optimal effect in each indicator. (compare results in Fig. 4 were scanned on November 12, 2018)

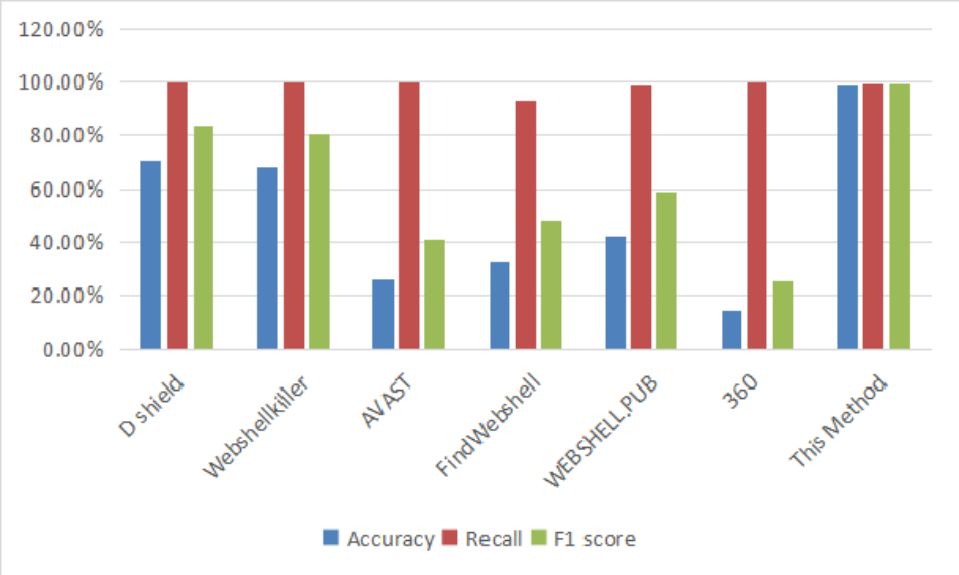

Fig. 4. Results of our method and current detection tools

From the comparison results of Fig. 4 and other detection software, it can be seen that most tools can not detect Webshell well according to the information of the PHP document, and the method proposed in this paper can achieve the accuracy of 98.447%. Therefore, the method proposed in this paper could detect unknown Webshell without the script source code (ie black box detection). The method can achieve the effect of white detection of other detection software.

6 Conclusion

This paper proposes a dynamic detection method based on Stacking model to detect Webshell. This method not only has high detection precision and accuracy, but also has certain independence. This paper uses a combination of multiple detection methods to detect Webshell, which overcomes the shortcomings of traditional detection methods. The experimental results demonstrated that the proposed method has great advantages compared with existing detection products. This method reduces the false positive rate and false negative rate, and greatly improves the current situation of false positive rate, high false negative rate and low detection precision. With the escape of Webshell technology has become increasingly complex, and the transmission of data flow is more complicated which is quite different from the complexity of normal programs. This provides new ideas for detection technology. At the same time, we can use the technology of honeypot technology, social engineering, fingerprint tracking and other technologies to achieve Webshell trace-ability in passive detection. And the machine learning based method will be more applied to the detection field.

Acknowledgements

This work was supported in part by NSFC (61472229 and 61702306), Sci. & Tech. Development Fund of Shandong Province of China (2016ZDJS02A11, ZR2017BF015 and ZR2017MF027), the Humanities and Social Science Research Project of the Ministry of Education (18YJAZH017), the Taishan Scholar Climbing Program of Shandong Province, and SDUST Research Fund (2015TDJH102).

References

[1] 360 Internet Security Center, 2017 China website security situation analysis report. <http://zt.360.cn/1101061855.php?dtid=1101062368&did=490995546>, 2018 (accessed 18.06.23).

[2] L.-Y. Deng, D.-L. Lee, Y.-H. Chen, L.-X. Yann, Lexical analysis for the Webshell attacks, in: Proc. 2016 International Conference on International Symposium on Computer, 2016.

[3] M. Zheng, M. Rui, Z. Tao, W. Wen, Research of Linux WebShell detection based on SVM classfier, Netinfo Security, 5(2014) 5-9.

[4] Y. Fei, G. Jian, Y. Wang, Black box detection of WebShell based on support vector machine, Journal of Nanjing University of Aeronautics & Astronautics 47(6)(2015) 924-930.

[5] Y.-P. Cui, K.-X. Shi, J.-W. Hu, Research of WebShell detection method based on XGBoost algorithm, Computer Science 45(S1)(2018) 375-379.

[6] L.-Y. Shi, F. Yong, WebShell detection method research based on web log, Journal of Information Security Research 2(1)(2016) 66-73.

[7] D. Hua, L. Jing, X.-D. Lu, S. Xin, Machine learning algorithm for intelligent detection of WebShell, Chinese Journal of Network and Information Security 3(4)(2017) 51-57.

[8] W.-C. Jia, L.-L. Qi, S. Fan, R.-G Hu, WebShell detection method based on random forest improved algorithm, Application Research of Computers 35(5)(2018) 1558-1561.

[9] M.S. Khan, S. Siddiqui, R.D. Mcleod, K. Ferens, W. Kinsner, Fractal based adaptive boosting algorithm for cognitive detection of computer malware, in: Proc.2017 International Conference on Cognitive Informatics & Cognitive Computing, 2017.

[10] X.-B. Xu, X.-M. Nie, A method of detecting WebShell based on multi-layer perception, Communications Technology 51(4)(2018) 895-900.

[11] H. Cui, D. Huang, Y. Fang, L. Liu, C. Huang, WebShell detection based on random forest-gradient boosting decision tree algorithm, in: Proc. 2018 Third International Conference on Data Science in Cyberspace (DSC), 2018.

[12] A. Firdaus, N.B. Anuar, M.F.A. Razak, I. A. T. Hashem, A.K. Sangaiah, Root exploit detection and features optimization: mobile device and blockchain based medical data management, Journal of Medical Systems 42(6)(2018) 112.

[13] Z. Ying, H. Yong, WebShell detection method based on correlation analysis, Journal of Information Security Research 4(3)(2018) 251-255.

[14] Y, Nan, F. Yong, H. Cheng, L. Liang, Semantics-based WebShell detection method research, Journal of Information Security Research 3(2)(2017) 145-150.

[15] T.D. Tu, C. Guang, X.-J. Guo, W.-B. Pan, Webshell detection techniques in web applications, in: Proc. 2014 5th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2014.

[16] W.-Q. Wang, G.-J. Peng, Z.-H. Chen, A.-Q. Hu, Ulti-strategy based framework for WebShell detection, Computer Engineering and Design 39(4)(2018) 907-911+917.

[17] Y. Xue, H. Geng, H. Zhang, Z. Xue, G. Xu, Semantic segmentation based on fusion of features and classifiers, Multimedia Tools & Applications 77(2018) 22199-22211.

[18] L.-J. Li, Y. Yu, S.-S. Bai, J.-J. Cheng, Towards effective network intrusion detection: a hybrid model integrating Gini index and GBDT with PSO, Journal of Sensors 6(2018) 1-9.

[19] M. Vajdi, A. Torkaman, M. Bahrololum, M.H. Tadayon, A. Salajegheh, Proposed new features to improve Android malware detection, in: Proc. 2017 International Conference on International Symposium on Telecommunications, 2017.

[20] J. Kim, D.H. Yoo, H. Jang, K. Jeong, WebSHArk 1.0: A benchmark collection for malicious web shell detection, Journal of Information Processing Systems 11(2)(2015) 229-238.