A Matrix Decomposition based Webshell Detection Method

Xin Sun
Electric Power Research Institute of State Grid Zhejiang Electric Power Company, China
16526452@qq.com

Xindai Lu
Electric Power Research Institute of State Grid Zhejiang Electric Power Company, China
lu_xindai@zj.sgcc.com.cn

Hua Dai
Electric Power Research Institute of State Grid Zhejiang Electric Power Company, China
56633685@qq.com

ABSTRACT

WebShell is a web based network backdoor. With the help of WebShells, the hacker can take any control of the web services illegally. The current method of detecting WebShells is just matching the eigenvalues or detecting the produced flow or services, which is hard to find new kinds of WebShells. To solve these problems, this paper analyzes the different features of a page and proposes a novel matrix decomposition based WebShell detection algorithm. The algorithm is a supervised machine learning algorithm. By analyzing and learning features of known existing and non-existing WebShell pages, the algorithm can make predictions on the unknown pages. The experimental results show that, compared with traditional detection methods, this algorithm spends less time, has higher accuracy and recall rate, and can detect new kinds of WebShells with a certain probability, overcoming the shortcomings of the traditional feature matching based method, improving the accuracy and recalling rate of WebShell detection.

CCS Concepts

• Security and privacy~Malware and its mitigation

Keywords

WebShells; matrix decomposition; machine learning; detection

1. INTRODUCTION

Nowadays, computer networks play an important role in human social life, and human is becoming more and more dependent on them [1]. Once out of the computer network, the entire community would be collapsed. Thus the security and stability of computer networks is becoming increasingly important. As the core of computer networks, the protection of servers is a vital part to ensure the safety and stability of computer networks.

Around the offense and defense of server, the WebShell is popular and often used by the majority of invaders as a weapon to obtain permissions of servers. WebShell is a kind of command-execution environment, often in the form of web pages, such as asp, php, jsp or cgi pages, and we can also call it the web page backdoor [2]. After invading a website, the hacker usually mix the asp or php backdoor file with the normal web pages together, and then he can use the browser to get access of the asp or php backdoor, get a command execution environment, and finally achieve the control purpose of web server. WebShell can be used for server management, online editing web scripts, uploading and downloading files, viewing the database and executing arbitrary commands, but once they are exploited by an intruder, the server will be in great dangers. Intruders could get permission of uploading files through xss and sql injection, and then upload a WebShell contained web page to the server or create a new file directly on it. After the existence of the file on the server, the intruder could access the web page to obtain higher permissions to achieve control of the server, and then he can use the server for malicious behavior, or directly steal the confidential documents [3].

ICCSP '17, March 17-19, 2017, Wuhan, China

DOI: http://dx.doi.org/10.1145/3058060.3058083

There is no clear classification criteria for WebShells, but it still can be divided in several aspects. For example, it can be classified into asp, php, jsp, and etc according to scripting language; and can be divided into big trojan, small trojan and a word trojan by its size. The big trojan has fully functions with friendly interactive interfaces [4], through which the user can delete or modify files and databases or perform some commands, and so on. Among these three kinds of Trojans, the size of big trojan is the largest, and it usually can't be uploaded directly because of the general restriction of privileges, so it usually carried out with the use of ponies. Small trojan doesn't have full functions, and it is usually used just for uploading and database mentioning rights [4]. It's generally only 5kb in size, and can be uploaded to the server while the privileges are insufficient, which is often used as right tools to upload large horses. A word trojan refers to receiving and executing scripts dynamically with just one line of code, often used for some of the key functions [5]. A word trojan n is usually inserted into the original normal files with a very small amount of code, so it can't be easily detected.

2. RELATED WORK

In order not to be detected by administrators for a long time, the intruders use the following methods to protect their WebShells [6]:

1. Information confused. The hacker inserts a large amount of comments, unwanted variables or functions into his WebShells, to reduce the core code proportions, allowing administrators to misunderstand the page. Security tools will also be affected in the testing process.

2. Character splicing. To avoid core functions calling codes being detected, the intruder will split the function name apart, and then stitch them together when they are used. In addition, replacing the relevant characters or strings is also effective.

3. Code encryption. As for the control code or data, they are encrypted before sending or using, and decrypted before running, like policy encryption in Vehicular ad hoc network [7]. Also, Java component can create/add a digital signature to a file. This signature ensures the integrity of the document [8]. These methods can confuse the features of pages, even machines can't recognize the encrypted code, so it's an effective method for static testing.

4. Page splitting. To avoid calling the key functions or parameters in a single file too dense so as to be detected, a full page will be split into multiple pages to upload. When used, one page could include other files and therefore call their functions or parameters. The method reduces the proportion of critical functions or parameters, and thus, can effectively reduce the probability of being detected.

5. Multiple coding. Similar to code encryption, this method does multiple coding on critical functions or parameters to change the rules of the page, and thus reduces the probability of being detected.

To solve the above problems, the administrators generally use the following methods to prevent or detect WebShells [9]:

1. Static detection. This method uses eigenvalues and hazard functions to detect WebShells. If a reasonable rule is found, this method could achieve high success rate of detection with easy operations, and quickly detect the presence of WebShells. Hujian Kang et al [10] extract different features of the target pages, use a decision tree to classify and detect WebShells; Ye Fei et al [11] analyze the structural and textual features of pages, use bag-of-words model to extract keywords, and then use the SVM method to classify and detect WebShells.

However, such static matching method only achieves high success rate with some existing WebShells, for some of the latest WebShells, leak detection rate will be relatively high, and it almost couldn't detect 0day WebShells. Thus, the static detection and manual detection generally cooperate with each other to complete WebShell defense, but for large companies with lots of host computers, there is a time-consuming issue.

2. Dynamic monitoring. For static detection, intruders often encrypt their WebShells to avoid being detected, but dynamic monitoring can solve this problem. Dynamic Monitoring could monitor the request and response flow generated in the BS activities, system commands, and status changes, to find abnormal behaviors and detect the existence of WebShells. For example, if it detects a user accesses or invokes a never used file, then the probability of containing WebShells of this file would be greatly increased. That is, the method can analyze WebShells from behavioral patterns. This method is capable to detect WebShells to a certain degree, but is difficult to detect some specific WebShells; intruders can also insert their WebShells into existing codes and increase the difficulty of being detected.

In this paper, based on matrix decomposition and machine learning methods, the algorithm could automatically learn features from the training data, and then use the learned algorithms to predict and ultimately achieve the purpose of classification. Compared with the existing general WebShell detection methods and other machine learning methods for categorizing, this method gives up the interpretability of forecasting results, and pursues a more accurate prediction; another advantage of this method is that, the prediction would be very fast after passing through the training process; in addition, the scalability of the algorithm is also very good.

3. MATRIX FACTORIZATION MODEL

In this paper, we use lowercase bold letters to represent vectors, and capital bold letters to represent matrixs, u to represent the feature of a corresponding column in a matrix, i to represent the feature of a corresponding row in a matrix. $r_{u,i} \in [0,1]$ is the score of object u corresponding to object i, if r_{ui} is closer to 1, then the feature u corresponding to feature i is more likely to contain WebShells; otherwise, if r_{ui} is closer to 0, then the feature u corresponding to feature i is more impossible to contain WebShells. $\mathbf{R}$ is the scoring matrix of size $m_u \times m_i$, which contains m_u columns and m_i rows. In addition, most of the elements in the scoring matrix are empty in most cases, so we use a triple $\mathbf{s} = (u, i, r_{ui})$ to represents a scoring message, and all the information is stored in the data set $\mathbf{S} = \{(u, i, r_{ui}) | r_{ui} \text{ is known}\}$.

The matrix decomposition model we use combines the k dimensions factor vector $\mathbf{p}_u \in \mathbf{R}^k$ of each column u and the k dimensions factor vector $\mathbf{q}_i \in \mathbf{R}^k$ of each row i together [12]. Let $\hat{r}_{ui}$ be the prediction of r_{ui}, and we can get $\hat{r}_{ui}$ from the following formula:

$$\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i \tag{1}$$

In addition, the model adds the penalty value of the vector and reduces the parameter value to avoid over-fitting [13]. Given a training set $\mathbf{S}_t \subset \mathbf{S}$ of a scoring matrix, the parameters of the model are estimated by:

$$\underset{\Theta}{argmin} \sum_{\mathbf{s} \in \mathbf{S}_t} \left(r_{ui} - \hat{r}_{ui}\right)^2 + \lambda \left(\|\mathbf{p}_u\|_2 + \|\mathbf{q}_i\|_2\right) \tag{2}$$

The matrix decomposition based WebShell detecting algorithm is a supervised machine learning algorithm. By analyzing the data and features of known pages, we could get weights of corresponding features and build a matrix model to predict whether the unknown page exists WebShells. The flow of the algorithm is shown in Figure 1.

3.1 Feature Determination

From the above definition of matrix decomposition, we can see that the general matrix decomposition model is predicted by calculating the score of feature u to feature i, so there are only two categories of the variables to be concerned, namely the feature u and feature i. For example, we can treat feature u as a user, so that different columns of a vector represents different users; and treat i as an object, so that different lines of a vector represents different items [14].

But in the WebShell detection process, because of the large number of variable features to be detected, such as the number of words, the number of different words, the maximum word length, etc., they are all far greater than 2, where if you still use the traditional matrix decomposition model to predict the WebShell, for example, with the feature u representing the number of words, and different columns of vector representing the number of

different words, then there will be a lot of features, so it would be too difficult to decompose the matrix; moreover, the upper limit of the number of words and the number of columns of the matrix are not determined, so the use of traditional methods is not feasible, and we need to modify the features of the model.

Before we can modify the features of the model, we first need to determine how many features are there in the testing process of WebShells. In the feature selection, if the number of features we choose is insufficient, then in the score forecasting process, it may be missing the features that have greater impacts on the score, thereby affecting the accuracy of the score prediction; and if too many features are selected, the model will be over-fitting, produce a large deviation, and become more complex. Therefore, we should choose a collection of features with appropriate numbers and influence as big as possible in the prediction process. In this paper, features are divided into text features and other features, where text features are the characteristics of the object in terms of the content, including the text length, word types, and the longest word length, etc.; other features are the ones other than the text features, including the file operations, the function operations, and so on [15]. In this paper, we select the following features:

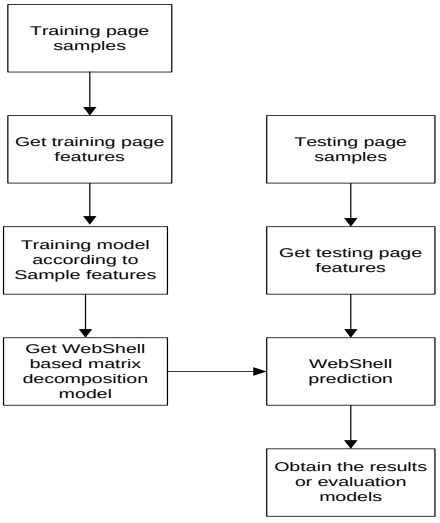

Figure 1. Flow chart of WebShell detection algorithm based on matrix decomposition.

Table 1 Feature table of WebShell detection model

Feature type	Text features	Other features
Feature description		Character manipulation function call
	Number of words	Key function call
	Number of different words	Encryption and decryption function call
	Max length of words	System function call
	Total length of text	File invocation
	Number of notes	ActiveX control call
	Number of special characters	Database call
		Number of scripts

After determining the number of features, we want to convert these features into the available form of the matrix decomposition model. Here we use a combination of classification methods.

First of all, we divide the features of the text which mentioned in Table 1 into several groups, for example, the special characters are divided by their count into three groups: 0 to 10, 10 to 50, and more than 50; the words are divided by their length into also three groups: 0 to 10,10 to 20, and more than 20; and so on. The specific boundaries and the number of groups can be adjusted according to different circumstances. After grouping the six features in the text, one is randomly selected from each feature to be combined. In this way, if each feature can be divided into three groups, we can get 3^6 features, namely a total of 729 combinations.

By analogy, we also group each of the other features. If each feature can be divided into two groups, we can derive 2^8 features from other features, a total of 64 combinations.

After the above combinations, we think of these 729 kinds of different combinations of text features as the characteristics of u, and these 64 kinds of different combinations of other features as the characteristics of i, thus we have determined the number of column and row vectors and their meanings of the scoring matrix $\mathbf{R}$, which is of size 729×64.

3.2 Matrix Factorization

After defining the scoring matrix and its corresponding column and row vectors, we can use the matrix decomposition model as described earlier for training and prediction.

During the WebShell training process, for a given file, $\hat{r}_{ui}$ is a dependent variable and denotes the existence of WebShells in the file. u represents the combination of text features of the file, and i represents the combination of other text features of the file. So each given file corresponds to one element in the matrix. Then we get the following equation:

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{p}_u^T \mathbf{q}_i \tag{3}$$

In the formula, μ represents the average of all the predictions of the scoring matrix, the parameters b_u and b_i represent the combination of text features u and other features i respectively, and correspond to the average of the possibility predictions of WebShell. The k dimensions factor vector $\mathbf{p}_u \in \mathbf{R}^k$ of the combination u and the k dimensions factor vector $\mathbf{q}_i \in \mathbf{R}^k$ of the combination i are the parameters corresponding to the combination of text features or other features, and constantly changes along the training process. Here, we use a gradient descent method to train $\mathbf{p}_u$ and $\mathbf{q}_i$.

In order to minimize the loss function after training [16], and obtain a more accurate prediction result, that is,

$$\underset{\Theta}{argmin} \sum_{\mathbf{s} \in \mathbf{S}_t} \left(r_{ui} - \hat{r}_{ui} \right)^2 + \lambda \left(b_u^2 + b_i^2 + \left\| \mathbf{p}_u \right\|_2 + \left\| \mathbf{q}_i \right\|_2 \right) \tag{4}$$

The training process is as follows:

1. For each given result r_{ui} in the training set, $\hat{r}_{ui}$ can be predicted by the formula (3)

2. The prediction error is : $\grave{o}_{ui} = r_{ui} - \hat{r}_{ui}$

3. Update $\mathbf{p}_u$ and $\mathbf{q}_i$ according to the gradient descent formula:

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \gamma \cdot \left(\grave{o}_{ui} \mathbf{q}_i - \lambda \cdot \mathbf{p}_u \right) \quad (5)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \gamma \cdot \left(\grave{o}_{ui} \mathbf{p}_u - \lambda \cdot \mathbf{q}_i \right) \quad (6)$$

Where γ is the step size of the gradient descent algorithm and λ is the learning rate.

After obtaining the trained model, we can use it to predict WebShells and in turn judge it by comparing the prediction result with the real result. For a given file, we need to predict the existence of WebShells. We first analyze the properties of the file mentioned in Table 1 and determine the corresponding rows and columns of the file and their corresponding $\mathbf{p}$ and $\mathbf{q}$ by their attributes, where $\mathbf{p}$ and $\mathbf{q}$ are the corresponding trained parameters. After that, we can use formula (3) to predict the file. If the result is greater than 0.5, the file is more likely to have WebShells, and the greater the value is, the greater possibility of existence will be; on the contrary, if the result is less than 0.5, the file is more impossible to contain WebShells, and the smaller the value is, the smaller possibility of existence will be.

4. EXPERIMENT

4.1 Experimental Data Source

As for the different languages, such as php, asp, jsp, etc., their WebShell scripts are similar, and our matrix decomposition based WebShell detection algorithm is generic to all of these languages, so we have collected the WebShells written in php language as experimental data. We use a total of 1002 php pages not contain WebShell and 347 php pages contain WebShells. Before the training and testing process, we divide the pages into five groups for different experiments, four of those are used as the training data, and another as the testing data to evaluate the detection capacity of our model comprehensively.

When classifying the page features, we directly classify them according to the result if the features are two types, and for those whose results are numbers, we classify them according to the mode of feature results, that is, if the result of a feature has a mode of p and maximum value of q, then it will be divided into three categories: $\left[0, \frac{p}{2}\right]$, $\left(\frac{p}{2}, \frac{n-p}{2}\right]$, $\left(\frac{n-p}{2}, \infty\right)$, then we group the training and prediction of data according to the classification criteria. As for the parameters, we have done different testing work and choose a step size of 0.1, learning rate selection of 0.05.

4.2 Criteria for Judgement

In order to evaluate the matrix decomposition prediction model introduced in this paper, two methods are adopted:

1. Root mean square error [17], also known as the standard error, is the square root of a ratio, which is the square of the deviation between the observed value and the true value to the number of observations n:

$$\mathrm{RMSE}\left(\mathbf{S}_t\right) = \sqrt{\frac{1}{|\mathbf{S}_t|} \sum_{r_{ui} \in \mathbf{S}_t} \left(r_{ui} - \hat{r}_{ui}\right)^2} \quad (7)$$

The smaller the RMSE, the lower the prediction error, the better the result.

2. Accuracy, recall rate, and F value evaluation [18]. Accuracy refers to the proportion between correct predictions and all predictions, recall rate is the proportion between the correct predictions and the total number of WebShells. Table 2 shows the classification of the predicted results, according to the table, the accuracy rate is $\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$, the recall rate is $\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$, and then F value can be calculated as $\frac{2 \cdot \mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$. The higher the accuracy and the recall rate, the higher the F, the better the prediction.

Table 2. Prediction classification table

Quantity		Prediction	
		0	1
Real	0	True Negative (TN)	False Positive (FP)
	1	False Negative (FN)	True Positive (TP)

4.3 Experimental Result

In order to evaluate the performance of the matrix decomposition based WebShell detection method, and show that the machine learning algorithm adopted in this paper has better performance than other algorithms in WebShell detection, we use the Logistic algorithm for comparison. At the same time, a common security product is used as the basic result in the detection process.

Logistic algorithm is a generalized linear regression algorithm [19]. Assuming that there are m independent variables $\mathbf{x}=\left(x_1, x_2, ..., x_m\right)$, and the conditional probability $\mathrm{P}\left(y=1|x\right)=p$ denotes the probability of the occurrence of an event P, then the algorithm can be expressed as:

$$\mathrm{P}\left(\mathrm{Y}=1|x\right) = \frac{1}{1+\mathrm{e}^{-g(x)}} \quad (8)$$

where:

$$g\left(x\right) = \theta^T \mathbf{x} \quad (9)$$

In the formula (9), $\mathbf{x}=\left(x_1, x_2, ..., x_m\right)$ corresponds to the attribute values of table 1, θ is the parameter of the attribute, then Logistic algorithm maps the inner product of the document attribute and the value to $[0,1]$ interval, which is used to predict whether WebShell exists or not. As with the matrix decomposition based WebShell detection algorithm introduced earlier in this paper, if the probability P is greater than 0.5, then WebShell is predicted to exist in the file, and if P is less than 0.5, then it predicts the absence of WebShell.

The experimental results are shown in Table 3 and Table 4:

Table 3. Prediction result of RMSE for WebShell

RMSE	Matrix decomposition	Logistic	Some security product
Experiment result	0.211	0.238	0.520

Table 4. Prediction result of accuracy, recall rate and F value for WebShell detection

Result	Matrix decomposition	Logistic	Some security product
Accuracy	0.982	0.880	0.497
Recall rate	0.813	0.601	0.411
F value	0.890	0.714	0.450

From the experimental results, it can be seen that the RMSE and accuracy rate of the matrix decomposition and Logistic algorithms are both better than the security product, which shows that it is effective to use the machine learning method to detect WebShells.

In addition, we compare the matrix decomposition method with the Logistic algorithm and find that the former is better. We assume that there are two reasons: First, compared with Logistic algorithm, the matrix decomposition method is more advanced, and is better in mining WebShell features from the page and then do more accurate predictions. Secondly, in the aspect of data source, the features of matrix decomposition based WebShell detection method have been classified, while the Logistic algorithm uses the accurate features, which may be the reason accounting for the WebShell detection difference, because the accurate features the logistic algorithm use may result in the emergence of over-fitting, but the classified features within a certain range will be regarded as the same data, which slow down over-fitting to a certain extent, and thus makes WebShell detection results more accurate.

5. Conclusion and Outlook

This paper analyzes the feature confusion method of Webshell and corresponding countermeasures. Based on the problem of current WebShell detection method, a matrix decomposition based WebShell detection method is proposed. This method is based on machine learning algorithm and can learn the features of WebShell pages quickly and accurately. The method overcomes the shortcomings of the traditional feature matching based method, and improves the accuracy and recall rate of WebShell detection. However, the rationality and validity of the classification method are not confirmed when classifying the page features. It is hoped that an accurate feature classification method can be obtained in the later research to improve the success rate of WebShell detection.

6. References

[1] Xia, Y., Ren, X., Peng, Z., Zhang, J., & She, L. (2014). Effectively identifying the influential spreaders in large-scale social networks. Multimedia Tools and Applications, 1-13.

[2] Tu, T. D., Guang, C., Xiaojun, G., & Wubin, P. (2014, July). Webshell detection techniques in web applications. In 2014 International Conference on Computing, Communication and Networking Technologies (ICCCNT), (pp. 1-7). IEEE.

[3] Stranieri, A., & Zeleznikow, J. (2002). WebShell: The development of web based expert systems. In Research and Development in Intelligent Systems XVIII (pp.245-258). Springer London.

[4] Zhao, X., Liu, H., Xue, G., & Cao, W. (2014, December). Analysis of trojan horse events by query of vulnerability information in searching engines. In 2014 Seventh International Symposium on Computational Intelligence and Design (ISCID), (Vol. 2, pp. 268-271). IEEE.

[5] Attack and Precaution of Webpage Trojan[J]. Computer Knowledge and Technology, 2014 (2X): 932-934.

[6] Mingkun, X., Xi, C., & Yan, H. (2012). Design of software to search ASP web shell. Procedia Engineering, 29, 123-127.

[7] Liu, X., Xia, Y., Chen, W., Xiang, Y., Hassan, M. M., & Alelaiwi, A. (2016). SEMD: Secure and Efficient Message Dissemination with Policy Enforcement in VANET. Journal of Computer and System Sciences. 1316-1328.

[8] Xia, Y., Liu, X., Xia, F., & Wang, G. (2016). A reduction of security notions in designated confirmer signatures. Theoretical Computer Science, 618, 1-20.

[9] Le, V. G., Nguyen, H. T., Lu, D. N., & Nguyen, N. H. (2016, September). A Solution for Automatically Malicious Web Shell and Web Application Vulnerability Detection. In International Conference on Computational Collective Intelligence (pp. 367-378). Springer International Publishing.

[10] HU, J., XU, Z., MA, D., & Yang, J. (2012). Research of Webshell Detection Based on Decision Tree [J]. Journal of Network New Media, 6, 005.

[11] Ye F, Gong J, Yang W. (2015). Black Box Detection of Webshell Based on Support Vector Machine[J]. Journal of Nanjing University of Aeronautics and Astronautics, 2015, 47(6): 924-930.

[12] Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

[13] Bishop, C. M. (2006). Pattern recognition. Machine Learning, 128.

[14] Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In Advances in neural information processing systems (pp. 556-562).

[15] Langley, P. (1994, November). Selection of relevant features in machine learning. In Proceedings of the AAAI Fall symposium on relevance (Vol. 184, pp. 245-271).

[16] Schorfheide, F. (2000). Loss function ‐ based evaluation of DSGE models. Journal of Applied Econometrics, 15(6), 645-670.

[17] Cohen, F. E., & Sternberg, M. J. (1980). On the prediction of protein structure: the significance of the root-mean-square deviation. Journal of molecular biology, 138(2), 321-333.

[18] Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006, December). Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In Australasian Joint Conference on Artificial Intelligence (pp. 1015-1021). Springer Berlin Heidelberg.

[19] Hosmer, D. W., & Lemeshow, S. (2000). Introduction to the logistic regression model. Applied Logistic Regression, Second Edition, 1-30.