ISSN 2347 - 3983
Volume 8. No. 7, July 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter63872020.pdf
https://doi.org/10.30534/ijeter/2020/63872020

A Method of detecting web attacks using behavioral profile analysis technique

Do Minh Tuan[1], Tisenko Victor Nikolaevich[2], Nguyen Hoang Long[3], Nguyen Vuong Tuan Hiep[4], Nguyen Quang Dam[5]
[1,3,4,5]Information Assurance dept. FPT University, Hanoi, Vietnam, tuandmse05518@fpt.edu.vn, longdhse05220@fpt.edu.vn, hiepnvtse05065@fpt.edu.vn, damnqse05820@fpt.edu.vn,
[2]Department Quality Systems, Peter the Great St. Petersburg Polytechnic University, Russia, St.Petersburg, Polytechnicheskaya, 29, v_tisenko@mail.ru

ABSTRACT

Web attack is a common attack technique that has been causing huge consequences for organizations and businesses. Therefore, the problem of early detecting and warning about web attack techniques is very necessary today. In this paper, we propose a method of detecting web attacks using machine learning techniques to analyze user behavior profiles. The user behavior profile represents the most basic characteristics of the access that the user is performing. The difference between our approach with other approaches is that instead of using techniques that seek and extract abnormal features and behaviors of access, we use the statistical method and then select and evaluate the values as the basis for the machine learning algorithm to classify normal access and abnormal access. The experimental results shown in section 4 of the paper have proved the optimal of our approach in the problem of abnormal access detection.

Key words: web application, web attacks detection, abnormal behavior, machine learning, behavioral profile analysis technique.

1. INTRODUCTION

The document [1] listed and described 10 dangerous vulnerabilities leading to the current web attack risks. The research [1] classified web attack techniques into 10 major types of attacks. According to the report [2], web attack techniques were rated as the top of the most dangerous attack techniques in 2019. Therefore, the problem of early detecting and warning the signs of web attack campaigns is very necessary today.
The studies [3], [4] showed that there are two main methods for detecting web attacks: the signature-based method through rule set and anomaly-based method based on data analysis and statistics to seek abnormal features in access. Rule-based and Signature-based are basically the same. The signature-based method often detects intrusions by checking whether the strings or expressions in the data traffic match with patterns or not. Meanwhile, the Rule-based method allows for performing more complex logical operations, as well as checking for a specific part of the web transaction that set in the rule. The Anomaly-based method is an approach based on abnormal signs with the idea of building a protective layer and it will observe the legitimate data flow of the application and then build a statistical model to evaluate access traffic in order to against attacks.
In addition, the studies [3], [4], [5] also presented two main approaches for detecting web attacks based on signature-based and anomaly-based, including static and dynamic detection methods. Static intrusion detection [3] is based on the detailed access log of the webserver where stored information about all access requests from users. In addition, web servers can also create access logs in special formats to control the data collected. Static intrusion detection is performed only after the transaction takes place. Dynamic intrusion detection (real-time) [5] not only can detect intrusion but also can react. To quickly and effectively identify, the filtering and detecting of the intrusion detection system operate according to the following models: i) Negative (Blacklist) security model is more commonly used. The system can detect an attack based on a known pattern of malicious attacks and configure to remove it. ii) Positive (Whitelist) security model is a better way to build policies applied in firewall activity. In web application security, the Positive security model lists all the descriptions in the application. Each description needs to define the following information: the allowed request methods (for example GET/POST or POST only); the allowed Content-Type; the allowed Content-Length; the allowed parameters; required parameters and optional parameters; Data type of all parameters (for example text or integer); Additional parameters (if any).
In this paper, we use dynamic analysis methods to analyze and evaluate the access to find normal and abnormal access. To accomplish this task, we will use dynamic analysis models and machine learning algorithms to build.

2. RELATED WORKS

There are two main types of web attack detection systems. The classification is mainly based on the detection mechanism of the methods.
Signature-based methods [3], [4]: this is a well-known approach and has been investigated by many researchers. So far, the research community of web attack detection has built up a complete Core Rule Set [6] to support network users. Currently, the Core Rule Set is used in most of the web firewalls [7].
Anomaly-based methods: there have been many different anomalies based approaches to network security. One of those approaches is based on the manual feature extraction techniques. Shi and et al. [8] present a list of features for queries that include URI's properties, such as length, quantity, type, and dangerous levels of each feature. After that, they applied Naïve Bayes, Decision tree, and SVM algorithm on those features to detect abnormal requests. Another approach is based on natural language processing. Zhang M and et al. [9] introduced a method that uses CNN to classify the attacks. Word2vec model is used to transform the raw request into a matrix, and then a CNN is adopted to extract the request's features. The research [10] introduces another approach using Gated recurrent unit (GRU) to analyze the contents of the requests. Every character in the request is converted into a one-hot vector with 129 dimensions, and every cell in GRU is used to analyze this request's content. Yang [11] also attempts a similar method that uses GRU to classify requests. In this research, he uses an encoding method that reconstructs a character into a 2-dimensional matrix. The authors of the research [12] use N-gram and Generic Feature Selection algorithms to extract features from DARPA and ECML/PKDD2007 datasets. In order to detect abnormal requests, they applied some clustering algorithms like C4.5, CART, Random Forest, or Random Tree. Aside from applying anomaly-based to detect abnormal requests in general, there are also some other researches focusing on detecting some common attacks on web applications [13], [14].

The study [14] presented methods of detecting abnormal behaviors of the web using the Dynamic Web Application Profiles analysis technique.

3. Web attack detection method based on behavioral profile analysis technique using machine learning

3.1 Web attack detection algorithms

3.1.1.Support vector machine

Support Vector Machine (SVM) [15] is a supervised learning algorithm that uses to classify data into separate groups. To calculate optimal using mathematics, SVM uses the term Margin. Margin is the distance from the dividing line to the nearest points of each class (in 2-dimensional space, the dividing line is a straight line; in multi-dimensional space, the dividing line is hyperplane). SVM tries to optimize by maximizing the margin value in order to find the best dividing line to classify data classes. In the 2-dimensional space problem, assuming that the dividing line has the equation:

$$w_1x_1 + w_2x_2 + b = 0$$

We can use the equation of the straight line or use equation ax + by + c = 0 to calculate. Here we use the values w_1, w_2, x_1, x_2 in order to later easily generalize to multi-dimensional space. For 2-dimensional space, the margin between 2 straight lines is calculated by the formula:

$$\text{margin} = \frac{2}{\sqrt{w_1^2+w_2^2}}$$

For multi-dimensional space, the equation of a hyperplane is:

$$w^T x + b = 0$$

Then, the margin is calculated by the formula: $\text{margin} = \frac{2}{\|w\|}$

3.1.2. Random Forest

Random Forest [16], [17] is a supervised learning algorithm. A forest includes trees. The more trees there are, the stronger the forest. Random Forest generates decision trees on randomly selected data samples. Each tree predicts results and the best solution is chosen by voting. It also provides a pretty good indicator of the importance of each feature. Random Forest has many applications such as suggestion, image categorization, and feature selection tools. It can be used to classify loyal loan applicants, identify fraudulent activity, and diagnose diseases. It is based on the Boruta algorithm which selects important features in the dataset. Technically, it is an ensemble method (based on the division and conquest approach) of decision trees generated on a randomly divided data set. The collection of these classification decision trees is also known as the forest. Individual decision trees are created by using feature selection indicators such as increasing information, increasing rate, and Gini index for each feature. Each tree depends on an independent random sample. In the classification problem, each vote and the most popular class are chosen as the final results. In the regression problem, the average of outputs of all tree is considered the final result. Random Forest is considered an accurate and powerful method because of the number of decision trees that participate in this process. It does not suffer from overfitting problems. The algorithm can be used in both classification and regression problems. Random Forest can also handle missing values. However, the Random forest is slow to make predictions because it has many decision trees. Whenever it makes a prediction, all trees in the forest must make a prediction for the same given input and then vote on it. The whole process is time-consuming.

Method of operation of the algorithm consists of 4 steps:

- Selecting random samples from a given dataset.
- Setting up the decision tree for each sample and get prediction results from each decision tree.
- Voting for each prediction result.
- Choosing the most predicted result as the final prediction

3.2. Selecting and extracting behavioral profile of web user

3.2.1. Data set description

In this paper, we extract abnormal behavior from the CSIC 2010 web attacks dataset [18]. This dataset is automatically generated and contains 36,000 normal requests and more than 25,000 abnormal requests. HTTP requests are labeled as normal or abnormal. The dataset includes attacks such as SQL, buffer overflows, crawl, file disclosure, CRLF injection, XSS, etc. The results are shown in Table 1.

.

Table 1: Data fields in the CSIC dataset

Data Field	Description
index	The ordinal number
method	Method for HTTP/1.1 such as GET, HEAD, POST, PUT, etc.
url	Links or addresses used to reference resources on the Internet
userAgent	An identifier string of web browser when sending a request to the web server
cacheControl	Optimize page load speed, increase security
accept	A type of data that will be received from the response (if the response returns a different type, it will be immediately banned). The most common types are text/html, application/xhtml+xm
acceptEncoding	Declare the content encoding type that the request accepts
AcceptCharset	Use to indicate which character set is accepted
acceptLanguage	Use to indicate which languages are accepted
host	The IP address of the server
contentLength	The number of octets that the message body represents
contentType	The type of information that the server returns to the client, it must match the accept that the client request
cookie	Contain encrypted information used to send to the server to help identify sessions between client-servers
PAYLOAD	CONTAIN DATA AND PARAMETERS SUBMITTED BY THE USER

In the problem of analyzing user behavior to identify anomalies, the focus will usually be on the data fields that the user enters. For the CSIC dataset, the paper will focus on the payload, URL, and cookie fields to build the feature set.

3.2.2. *Extracting features using TF-IDF and N-Gram techniques*

❖ **Applicating N-Gram in feature extraction**

The statistical language model allows to assign (estimate) the probability of a sequence of m elements (usually words) $P(w_1 w_2 ... w_m)$. In other words,this model allows to predict the likelihood of a string of words appearing in that language. According to the Bayes formula:

$$\mathbf{P(AB) = P(B|A) * P(A)}$$

Where:

- P(A) is the probabilities of observing A
- P(B) is the probabilities of observing B
- P(B|A) is a conditional probability: the likelihood of event B occurring given that A is true.

From that, having the following formula:

$$P(w_1 w_2 ... w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) * ... * P(w_m|w_1 w_2 ... w_{m-1})$$

According to this formula, the problem of calculating the probability of each word sequence becomes the problem of calculating the probability of a word with the condition is knowing the previous words.P(w_1 = $P(w_1|start)$ is the probability to w_1 is at the beginning of the string, or in other words, one can put a start symbol in each string.

In fact, based on the Markov model, one only calculates the probability of a word based on max n words preceding it (usually n = 0,1,2,3). Therefore, the statistical language model is often called the N-gram model where N is the number of words (including the word which should be counted and the previous context words). An n-gram model with n=1 is referred to as a "unigram"; with n=2 is a "bigram", with n=3 is a "trigram".

However, the bigger the n, the bigger the number of cases, so often using only *n* equal to 1, 2, or sometimes 3. To calculate the probability of the text with an acceptable amount of memory, we use approximating n-order Markov:

$P(w_1 w_2 \ldots w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) * \ldots * P(w_{m-1} \mid w_{m-n-1} w_{m-n} \ldots w_{m-2}) * P(w_m \mid w_{m-n} w_{m-n+1} \ldots w_{m-1})$

With this formula, we can build a language model based on the statistics of phrases with less than n + 1 words.

❖ **TF-IDF**

Term Frequency - Inverse Document Frequency (TF-IDF) is a technique of calculating the weight of a word, thereby evaluating the importance of that word in the text. Where:

- **TF**is calculated by the formula:

$$\textbf{tf}(t,d) = \frac{f(t,d)}{\max\{f(w,d) : w \epsilon d}$$

Where:

- tf(*t*, *d*): Term frequency of word *t* in document *d*
- f(*t*, *d*): the number of times that term *t* occurs in document *d*
- max({*f(w, d) : w* ∈ *d*}): the frequency of the most occurring term in document d

- **IDF**is calculated by the formula:

$$\textbf{idf}(t,D) = \textbf{log} \frac{|D|}{|\{d \epsilon D : t \epsilon d\}|}$$

Where:

- idf(*t*, *D*): the IDF value of word *t* in the corpus *D*.
- |*D*|*:* total number of documents in the corpus
- |*{d* ∈ *D : t* ∈ *d}*|*:* the number of documents where the term *t* appears

- **TF-IDF** is calculated by the formula:

tf-idf(t, d, D) = tf(t, d) x idf(t, D)

For example, extracting features using a combination of N-Gram and TF-IDF for user requests http://localhost:8080?id=abc';+drop+table+usuarios;. The results are shown in Table 2.

Table 2: Example of feature extraction using a combination of N-Gram and TF-IDF

	tfidf		tfidf		tfidf
tab	0.23375	**rop**	0.138057	**=ab**	0.138057
ble	0.23375	**st:**	0.138057	**?id**	0.138057
abl	0.23375	**t:8**	0.138057	**alh**	0.138057
abc	0.23375	**tp:**	0.138057	**bc'**	0.138057
';+	0.138057	**dro**	0.138057	**bct**	0.138057
lho	0.138057	**cal**	0.138057	**c';**	0.138057
e+a	0.138057	**cta**	0.138057	**ttp**	0.138057
hos	0.138057	**+ab**	0.138057		
htt	0.138057	**+dr**	0.138057		
id=	0.138057	**+ta**	0.138057		
le+	0.138057	**//l**	0.138057		
le;	0.138057	**/lo**	0.138057		
loc	0.138057	**80**	0.138057		
d=a	0.138057	**0?i**	0.138057		
oca	0.138057	**808**	0.138057		
op+	0.138057	**80?**	0.138057		
ost	0.138057	**://**	0.138057		
p+t	0.138057	**:80**	0.138057		
p:/	0.138057	**;+d**	0.138057		

4. EXPRIMENTS AND EVALUATE

4.1. Installation requirements

- *Hardware:* 32-bit (x86) or 64-bit (x64) processor with 2 gigahertz (GHz) or faster; 4GB RAM or higher; Hard disk with at least 10 GB of free space.
- *Software:* Installing on Windows/Linux systems (Centos 7.2); Programming tool: Python 2.7 or higher or Pycharm Professional 2020.1 IDE.

4.2. Experimental scenarios

The input CSIC data set will be divided into several different sets to examine the model. The model building process consists of two main stages: 1) Training Phase, 2) Testing phase.

❖ **Model Training Phase includes 3 main steps:**

- Step 1: Calculating the occurrence of important new key characters and save them in the database.
- Step 2: The vector space module is used to convert string data into vectors. Using the technique of extracting data that was introduced and described the calculation in section 2 by the author.
- Step 3: The data processing module using machine learning algorithms. Replacing the different algorithms in turn in order to determine the most optimal model for the problem. The algorithms used in this paper consist of KNN, SVM, and Random Forest.

❖ **Model Testing Phase includes 3 main steps:**

- Step 1: Removing labels on the test data.
- Step 2: Extracting features similar to step 2 in phase 1.
- Step 3: Experiment with models corresponding to machine learning algorithms built in stage 1. The author chooses the method of evaluating accuracy by using the confusion matrix describe as follows:

Confusion Matrix is a method of evaluating the results of classification problems with consideration of both the accuracy and completeness of the predictions for each class. A confusion matrix consists of the following 4 metrics for each classification class:

- TP (True Positive) is the number of abnormal samples that are correctly classified.
- FP (False Positive) is the number of normal samples that are incorrectly classified as abnormal sample.
- TN (True Negative) is the number of normal samples that are correctly classified.
- FN (False Negative) is the number of abnormal samples that are incorrectly classified as normal sample.

$$\text{precision}=\frac{TP}{TP+FP}$$

$$\text{recall}=\frac{TP}{TP+FN}$$

We see that the higher the Precision and the Recall, the better. But in reality, these two values cannot reach the maximum at the same time and often have to look for a balance. The $F1_{score}$ measure is the weighted average of Precision and Recall. It tends to be zero if these two values tend to be zero.

$$F1_{score} = 2*\frac{precision \times recall}{precision + recall}$$

4.3. Experimental results

The experimental dataset consists of 36,000 normal requests and 25,065 abnormal requests. Randomly dividing this data set at the ratio of 80% for training and 20% for testing. The number of classes is 2. From the implementation of dividing the input data with the above ratio, we get the results table 3.

Table 3: Experimental results for detecting web attacks based on user behavior

	SVM				**Random Forest**			
	F1_Score	**Confusion Matrix**	**Precision**	**Recall**	**F1_Score**	**Confusion Matrix**	**Precision**	**Recall**
N=2	0.983698	[7090 99] [136 4888]	0.986229	0.9811	**0.985959**	[7092 97] [105 4919]	**0.986507**	**0.98541**
N=3	**0.997219**	[7171 18] [22 5002]	**0.997496**	**0.9969**	0.984335	[7069 120] [105 4919]	0.983308	0.98536
N=4	0.994289	[7138 51] [31 4993]	0.992906	0.9956	0.978658	[7061 128] [178 4846]	0.982083	0.97525

Experimental results in Table 3 show that most of the results on the measurements of the SVM algorithm are better than the Random Forests algorithm. For the SVM algorithm, the anomaly detection result is highest with n = 3. For the Random Forest algorithm, the anomaly detection result is highest with n = 2.

Besides, from the experimental results, can see that extracting features based on N-Gram and TF-IDF gives the best results with N = 3. These results are completely consistent with reality because when increasing the values of N, the calculation and extraction will take more time but the results are not as good as smaller N values. Also, if N is too small, it is unreasonable because feature values will not be fully extracted.

5. CONCLUSION

Detecting anomalies from web user behavior is a difficult problem in preventing web application attacks. In fact, there are lots of access that slightly unusual but not really abnormal. This leads to noise when classifying, making the real abnormal created by the attack very difficult to identify. In this paper, based on N-Gram and TF-IDF techniques, we have tried to analyze and extract abnormal features from user's access as a basis so that machine learning algorithms can classify normal and abnormal access. The application of supervised machine learning algorithms in the early detecting and warning abnormal behavior of users from access is important in detecting and warning web attacks. This approach will quickly and accurately identify normal and abnormal access to help web systems minimize the risk of data loss. In the future, we will combine the dynamic web analytical method with advanced deep learning algorithms to improve the ability to detect and alert the abnormal behavior of web users

REFERENCES

[1] **OWASP Top Ten**. Available https://owasp.org/www-project-top-ten/ [access date 3/1/2020]

[2] **Symantec**. 2019 Internet Security Threat Report," vol. 24, Feb. 2019, [Online], Available: https://www.symantec.com/content/dam/symantec/docs /reports/istr-24-2019-en.pdf.

[3] Nancy Agarwal, Syed Zeeshan Hussain. **A Closer Look at Intrusion Detection System for Web Applications**. *Security and Communication Networks*, vol. 2018, no. 2, pp. 1- 27, 2018. https://doi.org/10.1155/2018/9601357

[4] M. H. Kamarudin, C. Maple, T. Watson, and N. S. Safa. **A New Unified Intrusion Anomaly Detection in Identifying Unseen Web Attacks**, *Security and Communication Networks*, vol. 2017, no. 1, pp. 1-18, 2017. https://doi.org/10.1155/2017/2539034

[5] **What are Web Application Vulnerabilities?** Available**:** https://www.rapid7.com/fundamentals/web-application-vulnerabilities/. [access date 15/2/2020]

[6] **Trustware SpiderLabs**, "ModSecurity: Open Source Web Application Firewall," *ModSecurity*, [Online], Available: https://www.modsecurity.org/.

[7] **ModSecurity Core Rule Set Project**, "OWASP ModSecurity Core Rule Set," 2016, [Online], Available:https://coreruleset.org/.

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

[9] M. Zhang, B. Xu, S. Bai, S. Lu, and Z. Lin. **A Deep Learning Method to Detect Web Attacks Using a Specially Designed CNN**. *Proceedings of the International Conference on Neural Information Processing*, pp. 828-836, 2017. https://doi.org/10.1007/978-3-319-70139-4_84

[10] J. Zhao, N. Wang, Q. Ma, and Z. Cheng. *Classifying Malicious URLs Using Gated Recurrent Neural Networks*. *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 385-394, 2019.

[11] W. C. Yang, W.Zuo, and B. Cui,. **Detecting Malicious URLs via a Keyword-based Convolutional Gated-recurrent-unit Neural Network**, *IEEE Access*,vol. 7, no. 2019, pp. 29891-29900, 2019.

[12] N. v. Ginkel, W. De Groef, F. Massacci, and F. Piessens. **A Server-Side JavaScript Security Architecture for Secure Integration of Third-Party Libraries**. *Security and Communication Networks*, vol. 2019, no. 6, pp. 1-21, 2019. https://doi.org/10.1155/2019/9629034

[13] I. Ro, J. S. Han, and E. G.Im. **Detection Method for Distributed Web-Crawlers: A Long-Tail Threshold Model**. *Security and Communication Networks*, vol. 2018, no. 4. pp 1-7, 2018.

[14] Cho Do Xuan, Nam Nguyen, Hoa Nguyen Dinh. **An adaptive anomaly request detection framework based on dynamic web application profiles**. *International Journal of Electrical and Computer Engineering (IJECE)*. Vol 10, No 5. pp. 5335-5346. 2020.

[15] Smola, A.; Vishwanathan, S.V.N. **Introduction to Machine Learning**; *Cambridge University Press*: Cambridge, UK, 2008.

[16] Leo Breiman. **Random Forests**. **Machine Learning**, vol. 45, no. 1, pp. 5- 32, 2001. https://doi.org/10.1023/A:1010933404324

[17] Thomas G. Dietterich. **Ensemble Methods in Machine Learning**. **Proceedings of the International Workshop on Multiple Classifier Systems**, (MCS 2000), pp 1-15, 2000. https://doi.org/10.1007/3-540-45014-9_1

[18] **CSIC-dataset**, "HTTP DATASET CSIC 2010,"2010, [Online], Available: http://www.isi.csic.es/dataset [access date 3/1/2020].