Machine Learning to Detect Anomalies in Web Log Analysis

Qimin Cao*, Yinrong Qiao
School of Computer Science and Software Engineering
East China Normal University
Shanghai, China
e-mail: ecnu_qmcao@163.com, yihqiao@126.co

Zhong Lyu
Shanghai International Studies University
Shanghai, China
e-mail: lvzhong424@163.com

***Abstract*—As the information technology develops rapidly, Web servers are easily to be attacked because of their high value. Therefore, Web security has aroused great concern in both academia and industry. Anomaly detection plays a significant role in the field of Web security, and log messages recording detailed system runtime information has become an important data analysis object accordingly. Traditional log anomaly detection relies on programmers to manually inspect by keyword search and regular expression match. Although the programmers can use intrusion detection system to reduce their workload, yet the log system data is huge, attack types are various, and hacking skills are improving, which make the traditional detection not efficient enough. To improve the traditional detection technology, many of anomaly detection mechanisms have been proposed in recent years, especially the machine learning method. In this paper, an anomaly detection system for web log files has been proposed, which adopts a two-level machine learning algorithm. The decision tree model classifies normal and anomalous data sets. The normal data set is manually checked for the establishment of multiple HMMs. The experimental data comes from the real industrial environment where log files have been collected, which contain many true intrusion messages. After comparing with three types of machine learning algorithms used in anomaly detection, the experimental results on this data set suggest that this system achieves higher detection accuracy and can detect unknown anomaly data.**

Keywords-machine learning; anomaly detection; log files component

I. INTRODUCTION

With the development of the Information Technology industry, Web servers gradually become the targets to be attacked for their high value. Attacks including SQL injection and cross-site scripting (XSS) occur frequently during the past years, hence, Web security has caught increasing attention in both academic and industrial community. In the Web security research, the anomaly detection to the Web is essentially the analysis towards log files. As crucial recording data, log files can reflect detailed runtime information during system operation and through which most of the attacks can be traced. However, log systems tend to generate volumes of data where valuable information might be concealed. Besides that, the constantly changing attacks and improving hacking techniques have made anomaly information very difficult to be detected, which lead to the current problem that manual analysis of log files is not efficient enough to meet the log auditing demands.

On top of that, traditional intrusion detection technology requires programmers or operators to extract attack features manually, and recognize patterns of typical attacks merely based on keyword search and rule match [1]. In other words, the traditional method cannot recognize unknown attacks and leads to many false positives.

In order to compensate for the shortcomings of traditional methods, many anomaly detection methods have been proposed in the past years. Due to the development of machine learning, many machine learning algorithms are also applied in log-based anomaly detection [2]. In accordance with the data type and machine learning technology involved, the anomaly detection methods can generally be divided into two categories: supervised anomaly detection [3] and unsupervised anomaly detection [4, 5]. The supervised approach requires standard training data, which is clearly defined in both normal and anomalous situations. Nevertheless, unsupervised methods do not need labels at all. Their work is based on the observation that an anomalous instance usually behaves as an outlier that is distant from other instances.

In this paper, an anomaly detection system is proposed based on machine learning for Web log files. To offset the aforementioned cons of traditional detection, this system adopts a two level machine learning algorithm. The first step is to select normal log files using decision tree classifier, and then modeling normal data set via hidden Markov model (hereinafter HMM). This is an anomaly detection model. It conducts automated learning and training based on mass data, bringing the defense side of web confrontation to a new high.

Experiments with 4,690,000 log messages from real industry situation and volumes of actual intrusion samples demonstrate the efficacy of our anomaly detection system, with 93.54% detection accuracy and 4.09% false positive rate.

978-1-5090-6352-9/17/$31.00 ©2017 IEEE

In summary, this paper makes the following contributions:

• An anomaly detection system for Web log files has been proposed. This system adopts a two-level machine learning algorithm, the decision tree algorithm and HMM, which can detect anomalous data and spot unknown attacks.

• After comparing several machine learning algorithms and finding that this anomaly detection system has less false positive rates without severely sacrificing precision.

• Web logs are collected from real industrial confrontation situations where have many genuine attack examples, which means the data set is more typical and practical.

The rest of this paper is organized as follows. It introduces background in Section II. Then it describes system framework and design in Section III and Section IV respectively, followed by the performance evaluation in Section V. It discusses related work in Section VI and concludes the paper in Section VII.

II. BACKGROUND

In this section, this paper first introduces the Hidden Markov Model (HMM), followed by the introduction to the web logs.

A. Hidden Markov Model

The hidden Markov model [6, 7] is a double stochastic process developed on the Markov chain and is a set of finite states that can be converted to each other. It contains two states: the hidden state and the observed state, which is a probabilistic model that can be used to represent the statistical properties of random processes. The hidden Markov model has the advantages of short algorithm and high detection accuracy. It has been widely used in fields including signal processing, image processing and pattern recognition. The hidden Markov model can be represented by a 5-tuple $\lambda(\mathrm{N}, \mathrm{M}, \mathrm{A}, \mathrm{B}, \pi)$.

(a) N is the number of hidden status of HMM, and the hidden status can be represented by $\mathrm{S} = (S_1, S_2, \cdots, S_n)$;

(b) M is the number of observational variable which can be expressed by $\mathrm{O} = (O_1, O_2, \cdots, O_n)$;

(c) π is the initial state distribution matrix of $1 \times \mathrm{N}$, $\pi_i = P(m_t, s_i)$, where m_t represents the hidden state of the model at time t, $m_t \in S, \pi_i \geq 0, \sum_{i=1}^{n} \pi_i = 1$;

(d) $\mathrm{A}(A = a_{ij})$ is the hidden state transition probability matrix of size $\mathrm{N} \times \mathrm{N}$, $m_t = s_j$, $m_{t-1} = s_i$, $a_{ij} = P(m_t = s_j | m_{t-1} = s_i) = P\left(\frac{m_t = s_j,\ m_{t-1} = s_i}{m_{t-1} = s_i}\right)$, where $a_{ij} \geq 0, \sum_{j=1}^{N} a_{ij} = 1$ which indicates that the probability that next time (t time) is transferred to the state s_j when the model is in the s_i state at time $\mathrm{t} - 1$;

(e) $\mathrm{B}(B = b_{ij})$ is the probability distribution of all observed events in the hidden state matrix, $b_{ij} = P(O = o_i | m_t = s_i)$, $i \in [1,2, \ldots, N]$, $j \in [1,2, \ldots, M]$, indicating that when the model is in a hidden state s_i, the observable variable is o_j, and the matrix size is $\mathrm{N} \times \mathrm{M}$.

B. Web server logs

A web server log file is a simple plain text file which records information each time a user requests resources from a website [8]. Typically there are four types of server logs: access log files, error log files, agent log files, referrer log files [9]. This paper focuses on web access log files or simply web log files. A standard web server like Apache generates log messages by default in the Common Log Format (CLF) specification [10,11]. The CLF log file contains a separate line for each HTTP request. Each line is composed of several tokens separated by spaces. If a token does not have a value, then it is represented by a hyphen (-): host identity user date request status bytes Table I is an example of one entry in a log file.

III. SYSTEM OVERVIEW

Figure 1 shows the completed system framework. This paper separates the whole system into four phases: (i) Data preprocessing, which consists of feature extraction and data labeling. More specifically, it means extracting the wanted features from raw server log files and labeling it using two labels: 1 to say that a unite of data is considered as an attack and 0 for normal behaviors. (ii) Decision Tree classifier. The data we prepared in the previous section is used as training data which then will be used to train decision tree classifier. Testing data is generated with the same function to test the trained classifier. (iii) Data extractor. Using the trained decision tree classifier to make predictions with new real-world log files, that is Data set 2 in Figure 1. Then there is the normal data set and anomalous data set, and the normal data set will be fed to the data extractor. (iv) Hidden Markov Model (HMM) training, which uses the processed data from the extractor to train a model. This model is qualified to represent the all trained normal data states, and can be seen as an anomaly detector accordingly.

IV. SYSTEM DESIGN AND IMPLEMENTATION

In this section, this paper explains our system framework in detail. Firstly, it describes how to extract features and label data. Then introducing the decision tree algorithm to predict log files. Next it describes an extractor to reprocess the normal data set emanating from the previous step. At last, this paper uses HMM to build a model which has been proven to be efficient in anomaly detection.

A. Data preprocessing

1) Features extraction: HTTP Status Code, whose specific meaning can be accessed via the Internet, is a three-digit numeric code defined by RFC 2616 to signal the HTTP responsive status of a Web server. When its logs are audited, the status code can provide useful information including malicious user registration and password-guessing, all of which can be judged by the status code during a specific period of time.

URLlength is defined as the length of the fifth field in a log message. Injection means the constructed code is added or inserted into the input parameters of an application, and then operated and processed these in background. According

to the experience from security engineers, the length of injected URL tends to be longer than the normal one. Therefore, URLlength is chosen as a feature to be trained.

Number of parameters in the query - The URL parameter is a "name = value" pair appended to the URL. The parameter starts with a question mark (?), and takes the format of name = value. If there are multiple URL parameters, the parameters are separated by an ampersand (&).

TABLE I. EXAMPLE OF ONE ENTRY IN A LOG FILE

216.35.116.91 - - [19Aug2000 14 47 37 -0400] "GET index.html HTTP1.0" 200 654		
host	216.35.116.91	The IP address of the client.
identity	-	The identity information reported by the client.
user	-	The user name of a successful HTTP authentication.
date	[19Aug2000 14 47 37 -0400]	The date and time of the request.
request	"GET index.html HTTP1.0"	The request line from the client is given in double quotes.
status	200	The three-digit HTTP status code returned to the client.
bytes	654	The number of bytes in the object returned to the client.

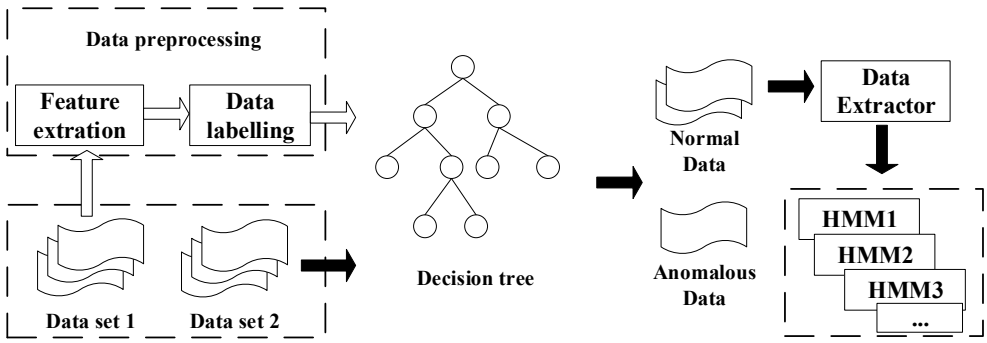

Figure 1. The framework of Anomaly Detection System

When the server receives the request and the parameter is appended to the requested URL, the server places the parameters when processing the page before the requested page is provided to the browser. As the injection attack is often injected into the parameters to achieve the purpose of attack, so this paper uses the number of parameters as a distinction feature between normal data and anomalous data.

2) Data labeling: Labeling consists of attributing a label for each unit of data, and this label will indicate if the concerned log line is considered as an attack or not. Labeling should normally be done manually by experienced security engineer. In our system, it is conducted automatically using a function that looks for specific patterns in each URL and decides if it is an attack. These patterns have been already applied to a log audit project in real situation, so the data labeling accuracy is guaranteed.

B. Decision Tree Classfier

The training data file, which is processed in the previous step, is used to train the decision tree classifier and predicts with the testing data to get the trained decision tree model. In fact, after testing a number of classifiers, the author argues that in this scenario, only with a simple rule-based classifier was able to meet demands in most of the situations. The decision tree model is then used in this paper. Due to space limitation, no further specific explanations are made.

C. Data Extractor

After training decision tree classifier, the author feeds it with new data set to make predictions, and gets the normal data set and anomalous data set. It is worth mentioning that, after getting the normal data set, the author of this paper manually confirms each piece of log message, to ensure that there is no anomalous data set. Next, the author performs another data processing on the tested data set. Specifically, extracting the URL field, then breaking down the URL by ampersand (&), and getting all the parameter names and the corresponding parameter values, as well as recording the states of the http requests. Sample form as shown in the table II.

TABLE II. PARANEBTERS FROM THE URL

Host	Key	P type	P name	P value
www.x.com	index.html	GET	id	123
www.x.com	index.html	GET	name	abc
www.x.com	test.php	POST	email	a@123.com
www.x.com	test.php	POST	name	abc
...	...	...	...	...

D. Hidden Markov Model

The author trains multiple HMMs with the data processed in sec IV-C, trains the normal models for the http request types GET, POST respectively, and also needs to train the normal models for every parameter under the GET and POST request. As illustrated in Figure 2. The reason for the above practice is
that there are possible injection attacks where there are parameters, and different parameters of the normal value is different. So through training multiple HMMs, this paper hopes to involve as many normal data states as possible.

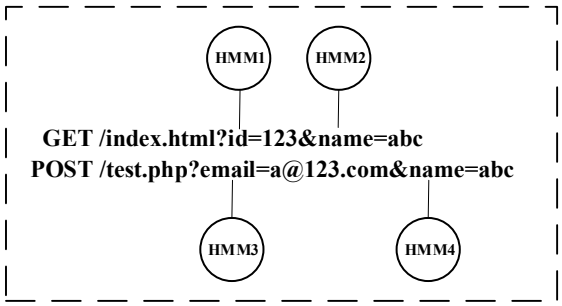

Figure 2. Hidden Markov Models

V. EXPERIMENTS

A. Configuration

Data Set. The data belongs to a log audit project of an IT security company, and it collects 4,690,000 training examples including invasions in real industry situation.

This data set consists of the data collected in four weeks. The training data is from the first two weeks including normal data and true invasions, and the rest is for testing. Due to the huge data volume, 2000 pieces of data are extracted from each weeks data set respectively for the purpose of building training data set and testing data set.

TABLE III. EVALYTION METRICS

	Predicted label		
Actual label		anomalous	normal
	anomalous	num_{aa}	num_{an}
	normal	num_{na}	num_{nn}

Evaluation Metrics. In order to investigate the effect of Anomaly Detection System, this paper first examines the performance of Anomaly Detection System comparison to three anomaly detection methods based on machine learning algorithms, that are Logistic Regression, Decision Tree, and SVM.

This paper uses the following evaluation metrics to evaluate their classification performance. The accuracy, precision, false positive rate (FPR), and true positive rate (TPR) are defined as follows while table III is the illustration of confusion matrix:

$$\mathrm{ACC} = \frac{num_{aa} + num_{nn}}{num_{aa} + num_{an} + num_{na} + num_{nn}}$$

$$\mathrm{PRE} = \frac{num_{an}}{num_{an} + num_{nn}}$$

$$\mathrm{FPR} = \frac{num_{an}}{num_{aa} + num_{an}}$$

$$\mathrm{TPR} = \frac{num_{nn}}{num_{na} + num_{nn}}$$

Experiment Environment. This paper measures the efficiency and scalability of Anomaly Detection System performance, and performs the entire process of Anomaly Detection System on a server with 8 GB memory, 8 cores at 3.30 GHz, and 1 TB hard drives.

B. Results

This paper first compares the performance of this anomaly detector with other machine learning algorithms for anomaly detection. Results taken by Anomaly Detection System (ADS), Logistic Regression (LR), Decision Tree (DT), and Support Vector Machine (SVM) algorithm are presented in Table IV.

TABLE IV. COMPARISON OF TEST PREFORMANCE ON ADS, LR, DT AND SVM

	ACC (%)	PRE (%)	TPR (%)	FPR (%)
ADS	93.54	92.07	89.90	4.09
LR	48.38	48.10	61.62	38.04
DT	74.19	86.25	55.47	6.34
SVM	90.32	91.60	97.20	14.09

For the algorithms mentioned above, this paper uses a widely-used machine learning package, scikit-learn [12] to achieve the Logistic Regression, Decision Tree, and SVM, these three learning models. There are many of parameters in SVM and Logistic Regression [13], and the author manually adjusts these parameters to achieve the best results during training. For SVM, the author tried different kernels and related parameters one by one, and found that SVM with linear kernel obtains the better anomaly detection accuracy than other kernels.

It is easy to tell from the table above that the accuracy when only adopting the decision tree algorithm is merely 74.19%, while this number can jump to 90.32% when using the HMM based on the previous result. Compared with SVM whose accuracy is 90.32%, this result is better and with lower false positive rate. To conclude, Anomaly Detection System achieved a high classification of 93.54%, with the false positive rate less than 5%. In addition, this paper analyzed the false report data. The result shows that deficient data training for some parameters made it difficult for the model to identify when part of the structure of parameters change.

VI. Related Work

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior. Dr. Denning [14] proposed a model of a real-time intrusion detection expert system which is able to acquire knowledge about behavior rules from audit records and is for detecting anomalous behavior. In 1988, a statistical anomaly detection system called Haystack was proposed [15], whose operation is based on behavioral constrains and on behavior for both group and individual users. In 2003, Kruegel et al. [16] proposed a multisensory fusion approach using Bayesian classifier for classification and suppression of false alarms In 2003, Yeung et al. [17] adopted dynamic modeling approach which is based on hidden Markov models (HMM) and the principle of maximum likelihood. In 2004, Decision Tree was first applied to failure diagnosis for web request log system in [18]. In [4], Liang et al. employ SVM to detect failures and compared it with other methods.

One of the first works on unlabeled network anomaly detection was presented in [19], where a clustering algorithm is used to discover anomaly in the training data set, and the clusters of normal data are used to construct a supervised detection model. Inspired by this, this paper, however, adopted a machine learning model on the basis of text analysis. This paper is able to build a model for massive normal log files and then treat it as the detector. For the data that is different from the detector will be tagged as anomalous instead. This can prevent unknown attacks effectively.

VII. Conclusion

This paper described an anomaly detection system based on machine learning for Web log files. The system will be applied to analysis log files for the main purpose. This paper first preprocessed the log files by applying decision tree algorithms based on patterns by rule set, then modelled the normal data set by using HMM, to build a model which can represent normal data status as the abnormity detector. The data belongs to a log audit project in real industry situation. Compared with other single-level machine learning algorithm, this system achieved the classify accuracy of 93.54%, and false positive rate of 4.09%.

In the future, the author of this paper will add training data with more parameters to build a stronger model. In anomaly detection field, any detection model can only be effective and responsible for the past situations. As existent attacks mutate, and many new attacks people never know occur, and the feasibility and effectiveness of this system will reduce for sure. Therefore, the author wants to introduce a retraining module. This module is supposed to identify useless and outmoded data, and the author will replace them with new one. The author hopes to iterate this retraining module through this method when necessary to improve its adaptability and scalability.

References

[1] A. J. Oliner, A. Ganapathi, and W. Xu, “Advances and challenges in log analysis,” *Commun. ACM*, vol. 55, pp. 55–61, 2012.

[2] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 2124, 2010, Haifa, Israel*, 2010, pp. 37–46.

[3] Waseem Ahmed, Yong Wei Wu, "Reliability Prediction Model for SOA Using Hidden Markov Model", ChinaGrid Annual Conference (ChinaGrid) 2013 8th, pp. 40-45, 2013.

[4] Y. Liang, Y. Zhang, H. Xiong, and R. K. Sahoo, “Failure prediction in IBM bluegene/l event logs,” in Proceedings of the 7th IEEE International Conference on Data Mining

[5] J. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, “Mining invariants from console logs for system problem detection,” in USENIX Annual Technical Conference, 23-25, 2010.

[6] W. Khreich, E. Granger, A. Miri, and R. Sabourin, “Adaptive roc-based ensembles of hmms applied to anomaly detection,” vol. 45, no. 1, 2012, pp. 208–230.

[7] J. J. Flores, F. Calderon, A. Antolino, and J. M. Gar-´ cia, “Network anomaly detection by continuous hidden markov models: An evolutionary programming approach,” vol. 19, no. 2, 2015, pp. 391–412.

[8] S. E. Salama, M. I. Marie, L. M. E. Fangary, and Y. K. Helmy, “Web server logs preprocessing for web intrusion detection,” vol. 4, no. 4, 2011, pp. 123–133.

[9] L. K. J. Grace, V. Maheswari, and D. Nagamalai, “Analysis of web logs and web user in web mining,” vol. abs/1101.5668, 2011.

[10] M. A. T. G. Castellano, A. M. Fanelli, “Log data preparation for mining web usage patterns,” vol. abs/1101.5668, 2007, pp. 371–378.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel.

[11] Zolotukhin M, Hamalainen T, Kokkonen T, et al. Analysis of HTTP Requests for Anomaly Detection of Web Attacks[C]// IEEE, International Conference on Dependable, Autonomic and Secure Computing. IEEE Computer Society, 2014:406-411.

[12] B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in python,” 2012. [12] P. Bod´ık, M. Goldszmidt, A. Fox, D. B. Woodard, and

[13] H. Andersen, “Fingerprinting the datacenter: automated classification of performance crises,” in European Con- ference on Computer Systems, Proceedings of the 5th European conference on Computer systems, 2010, pp. 111–124.

[14] D. E. Denning, “An intrusion-detection model,” in Proceedings of the 1986 IEEE Symposium on Security and Privacy, 1986, pp. 118–133.

[15] S. F. Owens and R. R. Levary, “An adaptive expert system approach for intrusion detection,” vol. 1, 2006, pp. 206–217.

[16] C. Krugel, D. Mutz, W. K. Robertson, and F. Valeur,¨ “Bayesian event classification for intrusion detection,” in 19th Annual Computer Security Applications Conference (ACSAC), 2003, pp. 14–23.

[17] D. Yeung and Y. Ding, “Host-based intrusion detection using dynamic and static behavioral models,” vol. 36, no. 1, 2003, pp. 229–243.

[18] M. Y. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. A. Brewer, “Failure diagnosis using decision trees,” in 1st International Conference on Autonomic Computing), 2004, pp. 36–43.

[19] R. Vaarandi, “A data clustering algorithm for mining patterns from event logs,” in Ip Operations Management, 2003, pp. 119–126.