Intrusion Detection through Contiguous System Call Traces

Lin CHEN1,*

1Shandong University, Jinan 250100, Shandong Province, China
E-mail: * chenlin@sdu.edu.cn

Abstract. The server backend operating system for the Servers and scientific computing necessities is a big market, and the majority market share of this market is hold by the Linux based Operating System for a very long time period. However, as running in backend lack of humanity monitoring, and the default signature-based security measures are ineffectual for detecting stealth attacks, the Linux OS needs an intelligent intrusion detection solution. This paper proposed an automatic method to inspect the contiguous system call traces and identify the abnormal system call pattern only by analyzing a trace of small window size, rather than probing the full traces.

1. Introduction
The server backend operating system for the Servers and scientific computing necessities is a big market, and the majority market share of this market is hold by the Linux based Operating System for a very long time period. The Host based Anomaly Detection Systems (HADS) is a subset of the Intrusion Detection Systems (IDS) series, has the great capability to detect the operating system level stealth attacks [1], and got a great attention for a long time. The traditional anomaly detection-based IDS suffer from high false alarm rates due to the difficulty of creating a robust and pervasive baseline [2]. Computer activity is very dynamic within modern systems, and the high level of variation that results from this dynamism greatly increases the difficulty of distilling an effective and accurate baseline from which to measure anomalous divergences.

This paper proposed a method to achieve the intrusion detection by leveraging a machine learning method named XGBoost [3] to train and build a smart model for the prediction and detection of the stealth intrusion. By analyzing the prediction model trained from the classic intrusion detection samples, which collected from the Linux based systems, the patterns of the common attack activities could be revealed.

2. Related Work
The intrusion anomaly detection is very useful for detecting and identifying the system intrusions as a significant data analysis purpose. A brief analysis of two major categories of anomaly detection techniques is described in this paper, which include classification and statistical infer [4].

A Bayesian network [5], also named as Bayes network, belief network, decision network, Bayes model or probabilistic directed acyclic graphical model, which is a probabilistic graphical model (a type of statistical model). The statistical model uses a directed acyclic graph (DAG) to represent a set of variables and the conditional dependencies.

In machine learning, a supervised learning model can be associated with learning algorithms that analyzing data used for classification and regression analysis, and the support-vector machines (SVMs, also support-vector networks [6]) are one of the supervised learning models.
The Bayesian network, the support vector machine, and the neural network [7] and other rule-based methods respectively, are the four kinds of mainstream methods included in the classification-based network anomaly detection theory system. And there are three methods included in the signal processing technique, statistical anomaly detection and principal component analysis (PCA) [8].

As the intrusion detection behaviors could be described by the time series data model, the long short-term memory theory [9] has been utilized in the intrusion detection data analysis and prediction. Because of their effectiveness in broad practical applications, LSTM networks have received a wealth of coverage in scientific journals, technical blogs, and implementation guides. Attention based memory is a level of sophistication further, and current attempts to determine the applicability of learned information on the basis of semantic matching is underway in the fields of IDS.

3. Intrusion Detection

An intrusion detector learning method is presented in this paper with the purpose of detecting the system level intrusions and protect the computer system from malicious users, including the insiders, especially the malware programs from unidentified provider. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between abnormal system call, also known as intrusions or attacks, and normal system operations.

3.1. Intrusion Detection Dataset Samples

A regular system intrusion dataset sample, which includes a wide diversity of intrusions simulated in the inside system environment, is audited for the model training process to build a generic purpose machine learning model for the system attack identification or classification. A raw training dataset [10], named ADFA-LD in short, is released by the Australian Defense Force Academy (ADFA), which is dedicated to provide a modern perspective for performance evaluation. A fully patched Ubuntu Linux 11.04 installation is used as the host OS to synthesis and collect the data samples. Apache Version 2.2.17 carrying PHP Version 5.3.5 were loaded to simulate the web-based attack behaviors. To provide a web-application attack vector through the known vulnerability issues, a remote PHP code injection malware is loaded and sending attacks to a Content Management System (CMS) versioned Tiki Wiki 8.1. This configuration represents a realistic modern target with small security flaws which can be exploited incrementally to provide a full system compromise. The raw training dataset used is about 26.5 megabytes of simple text format records, which is processed into 5,951 system call trace records. There are 308 features collected for each record, including 307 system call APIs and 1 feature named “window_size”. Each record is labeled with a class or a category, “Normal” or “Anomaly”.

The data analysis process discovered some feature patterns, which are plotted by density distributions, as shown in figure 1 and figure 2.
3.2. Model Training Process

The raw data sample is uploaded to the machine learning platform and parsed into the data frame. According to the conventional procedure, the data frame is split by the ratio of 3 to 1, that is, the 25% of the data frame becomes the validation data, and the rest of the data frame are collected into the training dataset. The XGBoost method, which would produce a reasonable model, is chosen to train the machine learning model. A XGBoost model is built with some optimized parameter, then tuning with the train data. For example, the learn rate is turned down from a default value of 0.3 to a smaller threshold of 0.1, then through the whole training process, the overfitting situation would be prevented. Considering the small amount of the sample set, the cross-validation (aka. k-fold) phase is given the 10, to make the validation result more convincing.

This paper employed a Macintosh laptop with a 3.2 GHz processor driven and 16 gigabytes memory mounted to perform the training and validating process, and the whole process takes about fifty seconds to finish.
3.3. Validation Metrics Result

One important essential part of any machine learning project would be evaluating the built machine learning algorithm. The trained model could give various results when evaluated with different measurements, for example, using a metric say accuracy_score in the evaluating process would return some satisfying results, but evaluated against other metrics such as logarithmic_loss or any other such metric would give very poor results. Most of the times classification accuracy is not enough to truly judge the model, even which is used to measure the performance of the model all the times.

To validate the model trained and built in the previous process, this paper utilized the confusion matrix to illustrate the validation metrics. A confusion matrix, also known as an error matrix is a table that is often used to describe the performance of a classification model (or “classifier”) in the field of machine learning and specifically the problem of statistical classification. The performance of the classification model is on a set of test data for which the true values are known. The confusion matrix allows the visualization of the performance of an algorithm. There were 5,469 normal records and 482 anomaly records in the sample we chosen for the model training process, which will cause the skewed class problem, which is lacking of really fix solution. In figure 3, this paper illustrated the final validation metrics by the two precision and recall metrics organized in the confusion matrix form to help validating the model. In Machine Learning, performance measurement is an essential task. When we need to check or visualize the performance of the multiclass classification problem, we can use AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) curve, which is one of the most important evaluation metrics for checking any classification model’s performance. The AUC of the model is shown in figure 4.

![Confusion Matrix](image1)

**Figure 3: Validation Metrics – Confusion Matrix**

![ROC Curve](image2)

**Figure 4: ROC Curve**
4. Conclusion
The paper described the requirement for the intrusion detection and identification in the Linux-Based operating system, and sought for finding an approach to automate the tasks by deploying a machine learning model into the system. Based on a public dataset provided by the ADFA, this paper researched the problem and designed a process to build the model for the intrusion detection and identification task, by following the XGBoost, a favourite machine learning method has been used in lots of pattern-based identification theory. The evaluation metrics showed that trained machine learning model performed very well on the given data sample, which would be a good proof for the deployment of an automated model into the operating system could solve the intrusion and identification problem.

References
[1] Hu, J. (2010) Host-based anomaly intrusion detection. In Handbook of Information and Communication Security, 235–255.
[2] Creech G, Hu J. (2014) A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns. IEEE Trans. Comput. 63:807-819.
[3] Chen T, Guestrin C. (2016) Xgboost: A scalable tree boosting system. In: the 22nd ACM SIGKDD International Conference. New York. pp 785–94.
[4] Ahmed M, Naser Mahmood A, Hu J. (2016) A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60:19–31.
[5] Friedman N, Geiger D , Goldszmidt M. (1997) Bayesian network classifiers. Machine Learning, 29:131–63.
[6] Agarwal B, Mittal N. (2012) Hybrid approach for detection of anomaly network traffic using data mining techniques. Procedia Technology, 6:996–1003.
[7] Graves A, Wayne G, Reynolds M, Harley T, Danihelka I, Grabska-Barwi 'n ska A, Colmenarejo S G, Grefenstette E, Ramalho T, Agapiou Jothers. (2016) Hybrid computing using a neural network with dynamic external memory. Nature, 538:471-476.
[8] Wold, S., Esbensen, K., Geladi, P. (1987) Principal component analysis. Chemometrics and intelligent laboratory systems, 2:37-52.
[9] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9:1735-1780.
[10] Haider W, Creech G, Xie Y, Hu J. (2016) Windows Based Data Sets for Evaluation of Robustness of Host Based Intrusion Detection Systems (IDS) to Zero-Day and Stealth Attacks. Future Internet, 8:29.