Dynamic Tracing Buffer for Intrusion Detection by XGboost

Lin CHEN1,*
1Shandong University, Jinan 250100, Shandong Province, China
E-mail: * chenlin@sdu.edu.cn

Abstract. Dynamic tracing buffer for the intrusion detection program or service is a novel method to design the system guard process. However, due to lack of human monitoring and its default signature-based security measures are ineffectual for detecting stealth attacks, the server distributed system needs an intelligent intrusion detection solution. This paper proposed a novel method to achieve the intrusion detection task through a dynamic contiguous system call tracing buffer.

1. Introduction
Intrusion Detection Systems (IDS), specifically Host based Anomaly Detection Systems (HADS), have received a great deal of attention over the past decade due to their capability to detect stealth attacks at the OS level [1]. The traditional anomaly detection-based IDS suffer from high false alarm rates due to the difficulty of creating a robust and pervasive baseline [2].

This paper proposed a method to achieve the intrusion detection by leveraging the dynamic tracing buffer method and the XGBoost [3] machine learning method to train a prediction model. The size of the tracing buffer is related to the precision and effectiveness of the intrusion detection task.

2. Related Work
The intrusion anomaly detection is an important data analysis task which is useful for detecting and identifying the system intrusions. The traditional method to design the intrusion detection program is based on the theory of statistical infer [4]. The classification-based network anomaly detection includes four kinds of mainstream methods, support vector machine [5], Bayesian network [6], neural network [7] and other ruled-based methods respectively.

3. Dynamic Tracing Buffer Method
This paper proposed a novel method to achieve the intrusion detection task through a dynamic contiguous system call tracing buffer. The size of the tracing buffer is related to the precision and effectiveness of the intrusion detection task. A small tracing buffer will perform effectively, but with a consequence of a low precision, such as a lack of regulation or costing a collateral damage. Failed in the anomaly process detection is one common example of the lack of regulation. In contrast to the lack of regulation, been convicted of misjudgment for killing a normal process is another common case of the collateral damage. Oppositely, a big tracing buffer will give a high precision, but accompanying a heavy overhead, especially under a heavy system load situation, because more bigger tracing buffer costs more memory footprint used or referenced by the intrusion detection diagnosis program while running in backend service model.

The intrusion detection diagnosis program or service is aimed to detect the system level intrusions and protect the computer system from unauthorized users, including perhaps insiders, especially the...
malware programs from unidentified provider. The main core of the intrusion detection diagnosis process is a predictive model, or more specifically a classifier, which was built by an intrusion detector learning task in advance. The classifier possesses the capability of distinguishing between abnormal system call, also known as intrusions or attacks, and normal system operations.

3.1. Intrusion Detection Dataset
To build the described classifier, a general machine learning model, for the system attack identification or classification, requiring a standard system intrusion data sample. The raw training dataset [8] audited for the model training process, 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, and includes a wide variety of intrusions simulated in an inside system environment.

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”.

3.2. Dataset Analysis
Data analysis, or feature engineering, the corresponding academic term, is a precondition of the model training process. The ADFA-LD dataset contains 5,951 records, each of which is labeled with a class or a category, “Normal” or “Anomaly”. The number of the records labeled “Normal” is 5469, and the rest 482 records are labeled “Anomaly”. In other words, the ADFA-LD dataset included 5,469 normal samples and 482 anomaly samples, would cause the skewed class problem to the model built on the dataset, which there is not really a solution to the problem. In this case, the precision and recall measures are leveraged to evaluating the final model. The precision metric describes how many of the data samples, which got classified as positives, actually are illustrating positive. While, the recall metric refers to the percentage of correctly classified positive based on the overall number of positives of the data set.

The dynamic tracing buffer is related to the feature named “window_size” in the ADFA-LD records. To discover the trends of the tracing buffer size in the samples, the numeric distribution and the density distribution of the window_size feature are analyzed. The density distribution of the feature is shown in figure 1, while the numeric distribution result is shown in figure 2.

The density distribution graph figure 1a and figure 1b are produced by the same dataset samples, with the only difference of window size value range in the x axis. The figure 1a shows the domain between 0 and 4500, while the figure 1b shows the domain between 0 and 2000. As shown in the figure 1a and 1b, the statistic of the feature is gathered in the table 1. The numeric distribution of the feature could be more comprehensive to recognize the trend and the pattern, which is shown in figure 2a and figure 2b.

The maximum value of the window size in the samples is 4494, while in contrast, the minimum value is 75. According to the anomaly samples, the maximum value of the window size is 2620.
### Figure 1: Feature Density Distribution

![Density plot of Window Size of samples](a)

![Density plot of Window Size of samples](b)

### Figure 2: Feature Numeric Distribution

![Histogram of data_size](a)

![Histogram of data_size](b)

### Table 1. Statistic of feature `window_size`.

| `window_size` | Samples | Anomaly Samples | Normal Samples |
|---------------|---------|-----------------|----------------|
| > 0           | 5951    | 482             | 5469           |
| > 1000        | 687     | 30              | 685            |
| > 2000        | 153     | 2               | 151            |
| > 3000        | 44      | 0               | 44             |
| > 4000        | 2       | 0               | 2              |

3.3. **Model Training and Validation Process**

According to the density and numeric distribution of the feature “`window_size`”, the raw data sample is resampled by the value of the feature “`window_size`”, more accurately, the sample of the “`window_size`” feature value between 100 and 1000 is extracted into the new dataset. 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 75% of the data frame combines the training data, and the rest of the data sample becomes the validation data. The XGBoost method, which would produce a reasonable model, is chosen to train the machine learning model.
Evaluating machine learning algorithm is an essential part of any project. The final validation metrics is presented by the confusion matrix measured by precision and recall, as shown in figure 3, while the AUC of the model is shown in figure 4.

|       | Anomaly | Normal | Error | Rate | Precision |
|-------|---------|--------|-------|------|-----------|
| Anomaly | 80      | 36     | 0.2903| 36 / 124 | 0.79     |
| Normal | 24      | 1339   | 0.0176| 24 / 1,363| 0.07     |
| Total  | 112     | 1375   | 0.0403| 68 / 1,487| 0.71     |
| Recall | 0.71    | 0.98   |       |       |           |

Figure 3: Validation Metrics – Confusion Matrix

Figure 4: ROC Curve

4. Conclusion
In this paper, the dynamic tracing buffer method and the XGBoost machine learning method are leveraged to build a model for the intrusion detection and identification based on the public ADFA-LD data sample. The model trained on the buffer size between 100 and 1000 performs very well on the contiguous system call traces data.

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, Guerstin 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] Agarwal B, Mittal N. (2012) Hybrid approach for detection of anomaly network traffic using data mining techniques. Procedia Technology, 6:996–1003.
[6] Friedman N, Geiger D, Goldszmidt M. (1997) Bayesian network classifiers. Machine Learning, 29:131–63.
[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] 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.