Learning Classifiers for Misuse Detection Using a Bag of System Calls Representation*

Dae-Ki Kang[1], Doug Fuller[2], and Vasant Honavar[1]

[1] Artificial Intelligence Lab, Department of Computer Science, Iowa State University
{dkkang, honavar}@iastate.edu
[2] Scalable Computing Lab., Iowa State University and U.S. Department of Energy
dfuller@scl.ameslab.gov

Abstract. In this paper, we propose a "bag of system calls" representation for intrusion detection of system call sequences and describe misuse detection results with widely used machine learning techniques on University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences with the proposed representation. With the feature representation as input, we compare the performance of several machine learning techniques and show experimental results. The results show that the machine learning techniques on simple "bag of system calls" representation of system call sequences is effective and often perform better than those approaches that use foreign contiguous subsequences for detecting intrusive behaviors of compromised processes.

1 Introduction

In most intrusion detection systems (IDS) that model the behavior of processes, intrusions are detected by observing fixed-length, contiguous subsequences of system calls. For example, in anomaly detection, subsequences of input traces are matched against normal sequences in database so that foreign sequences [1, 2] are detected. One potential drawback of this approach is that the size of the database that contains fixed-length contiguous subsequences increases exponentially with the length of the subsequences. In this paper, we explore an alternative representation of system call traces for intrusion detection. We demonstrate that simple *bag of system calls* representation of system call sequences is surprisingly effective in constructing classifiers for intrusion detection of system call traces.

2 Alternative Representations of System Call Sequences

Let $\Sigma = \{s_1, s_2, s_3, \ldots, s_m\}$ be a set of system calls where $m = |\Sigma|$ is the number of system calls. Data set D can be defined as a set of labeled sequences $\{< Z_i, c_i > | Z_i \in \Sigma^*, c_i \in \{0, 1\}\}$ where Z_i is an input sequence and c_i is a

* Supported by NSF grant IIS 0219699.

P. Kantor et al. (Eds.): ISI 2005, LNCS 3495, pp. 511–516, 2005.

corresponding class label with 0 denoting a "normal" activity and 1 denoting a "intrusive" activity. Given the data set D, the goal of the learning algorithm is to find a classifier $h : \Sigma^* \rightarrow \{0, 1\}$ that maximizes given criteria. Such criteria are accuracy, detection rate and false positive rate. Each sequence $Z \in \Sigma^*$ is mapped into a finite dimensional feature vector by a feature representation $\Phi : \Sigma^* \rightarrow \mathbf{X}$. Thus, the classifier is defined as $h : \mathbf{X} \rightarrow \{0, 1\}$ for data set $\{< X_j, c_j > | X \in \mathbf{X}, c_j \in \{0, 1\}\}$. This allows us to use a broad range of machine learning algorithms to train classification for intrusion detection.

2.1 Contiguous Foreign Subsequences

In this approach, a feature is defined as $X_j = x_1 x_2 x_3 \ldots x_l$, a substring of Z_i, where $x_k \in \Sigma$ and l is a constant. The number of possible features is $\left|\Sigma^l\right| \geq j$ and each feature X_j is assigned a class label c_i according to the original sequence Z_i. *STIDE* [3] uses sliding windows with length l over an original input trace to generate fixed-length substrings as features and constructs a database of the features in the training stage, and decides a test sequence is anomalous if the number of mismatches in the user-specified locality frame (locality frame count), which is composed of adjacent features in the frame, is more than the user-specified threshold.

2.2 Bag of System Calls

"Bag of system calls" representation is inspired by "bag of words" representation that has been demonstrated to be effective in text classification problems. In our approach, a sequence is represented by an ordered list $X_i = \langle c_1, c_2, c_3, \ldots, c_m \rangle$ where $m = |\Sigma|$ and c_j is the number of occurrence of system call s_j in the input sequence Z_i. Note that this representation of system call traces does not preserve information about relative order of system calls in the sequence.

3 Data Sets

3.1 UNM System System Call Sequences

The University of New Mexico (UNM) provides a number of system call data sets. The data sets we tested are "live lpr", "live lpr MIT", "synthetic sendmail", "synthetic sendmail CERT", and "denial of service"(DoS).

In UNM system call traces, each trace is an output of one program. Most traces involve only one process and usually one sequence is created for each trace. Sometimes, one trace has multiple processes. In such cases, we have extracted one sequence per process in the original trace. Thus, each system call trace can yield multiple sequences of system calls if the trace has multiple processes. Table 1 shows the number of original traces and the number of sequences for each program.

3.2 MIT Lincoln Lab Data Sets

We used data sets provided by the MIT Lincoln Lab [4]. The fourth week (starting at 6/22/98) training data set of year 1998 is used for the experiments in this

Table 1. The number of original traces and generated sequences in UNM data sets

Program	# of original traces	# of sequences
live lpr (normal)	1232	1232
live lpr (exploit)	1001	1001
live lpr MIT (normal)	2704	2704
live lpr MIT (exploit)	1001	1001
synthetic sendmail (normal)	7	346
synthetic sendmail (exploit)	10	25
synthetic sendmail CERT (normal)	2	294
synthetic sendmail CERT (exploit)	6	34
denial of service (normal)	13726	13726
denial of service (exploit)	1	105

paper. MIT Lincoln Labs datasets include omnibus files containing all system call traces. For each omnibus file, there is a separate, network traffic analysis data file that indicates inbound network connections to the system. Attack attempts are logged with the network data, so labeling of the training data requires cross-indexing this file with the system call trace file. The system call trace file identifies the source of each call using the process ID. Therefore, cross-indexing requires tracking the argument to the 'exec' system call identifying the binary to be executed. Additionally, the timestamps from the network traffic analyzer do not exactly correspond to the execution timestamps from the operating system kernel. A tolerance of one second was chosen and seems to permit the matching of a large majority of connection attempts with their corresponding server processes run on the target system. All processes detected that do not correspond to some network connection attempt identified in the trace are removed from consideration (since they cannot be classified), as are all calls attributed to a process ID for which an 'exec' system call is not found. The resulting data are available at `http://www.cs.iastate.edu/~dkkang/IDS_Bag/`.

4 Experiments and Results

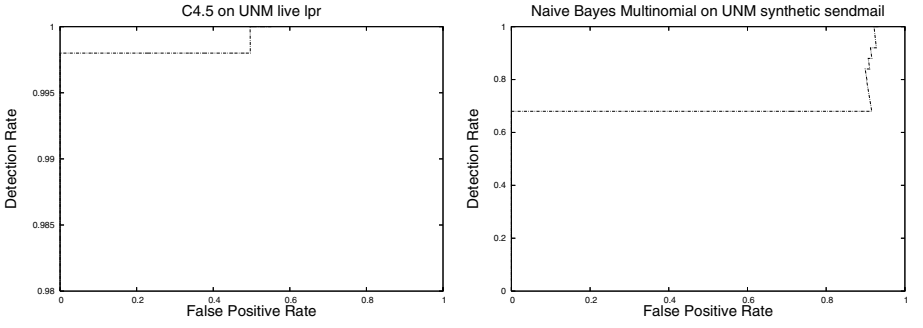
For the evaluation of classifiers generated in the experiment, ten-fold cross validation is used for rigorous statistical evaluation of the trained classifiers. Thus, in each experiment, the data set is divided into ten disjoint subsets, nine of which are used for training the classifier and the tenth part used for evaluating the classifier. The reported the results represent averages over ten such runs. Table 2 shows the accuracy, detection rate, and false positive rate [5] of the data sets we tested. The detection rate is a fraction of the intrusions identified and the false positive rate is a fraction of normal data mis-identified as intrusion.

Figure 1 shows the Receiver Operating Characteristic (ROC) Curve of "UNM live lpr" and "UNM synthetic sendmail" data sets using C4.5 and Naive Bayes Multinomial algorithms respectively.

Table 2. Percentage of misuse detection based on 10 fold cross-validation

Program	Naive Bayes Multinomial	C4.5	RIPPER	SVM	Logistic Regression
UNM live lpr					
accuracy	83.43	99.91	99.91	100.00	99.91
detection rate	100.00	99.80	99.80	100.00	100.00
false positive rate	30.03	0.00	0.00	0.00	0.16
UNM live lpr MIT					
accuracy	54.52	99.89	99.86	99.83	99.97
detection rate	100.00	99.90	99.80	99.80	99.90
false positive rate	62.31	0.11	0.11	0.14	0.00
UNM synthetic sendmail					
accuracy	20.21	94.87	94.33	95.68	95.41
detection rate	92.00	40.00	48.00	40.00	64.00
false positive rate	84.97	1.15	2.31	0.28	2.31
UNM synthetic sendmail CERT					
accuracy	24.39	96.64	95.42	96.03	96.03
detection rate	100.00	85.29	82.35	64.70	82.35
false positive rate	84.35	2.04	3.06	0.34	2.38
UNM denial of service					
accuracy	98.70	99.97	99.96	99.98	99.97
detection rate	44.76	99.04	98.09	100.00	99.04
false positive rate	0.88	0.02	0.02	0.01	0.01
MIT LL 1998 4^{th} Week					
Monday					
accuracy	100.00	100.00	100.00	100.00	100.00
detection rate	100.00	100.00	100.00	100.00	100.00
false positive rate	0.00	0.00	0.00	0.00	0.00
Tuesday					
accuracy	99.55	99.55	99.55	99.55	99.55
detection rate	98.60	98.60	98.60	98.60	98.60
false positive rate	0.00	0.00	0.00	0.00	0.00
Thursday					
accuracy	99.73	99.73	99.73	99.73	99.73
detection rate	100.00	100.00	100.00	100.00	100.00
false positive rate	0.04	0.04	0.04	0.04	0.04
Friday					
accuracy	98.80	98.80	98.80	98.80	98.80
detection rate	89.28	89.28	89.28	89.28	89.28
false positive rate	0.00	0.00	0.00	0.00	0.00

The results in table 2 show that standard machine learning techniques are surprisingly effective in misuse detection when they are used to train misuse detectors using simple bag of system calls representation. For example, with SMO (a widely used algorithm for training SVM) using a linear kernel, an SVM can perfectly detect both normal and intrusion sequences in the "UNM live lpr" data set.



(a) C4.5 decision tree induction on UNM live lpr

(b) Naive Bayes Multinomial on UNM synthetic sendmail

Fig. 1. ROC Curve of "UNM live lpr" and "UNM synthetic sendmail" data sets in misuse detection

5 Summary and Discussion

Results of our experiments using widely used benchmark data sets - the University of New Mexico (UNM) and MIT Lincoln Lab (MIT LL) system call sequences show that the performance of the proposed approach in terms of detection rate and false positive rate is comparable or superior to that of previously reported data mining approaches to misuse detection. In particular, as shown in table 2, the proposed methods achieve nearly 100% detection rate with almost 0% false positive rate on all the data sets studied with the exception of two synthetic data sets ('UNM synthetic sendmail' and 'UNM synthetic sendmail CERT').

When compared with the widely used fixed-length contiguous subsequence models, the *bag of system calls* representation explored in this paper may seem somewhat simple. It may be argued that much more sophisticated models that take into account the identity of the user or perhaps the order in which the calls were made. But our experiments show that a much simpler approach may be adequate in many scenarios. The results of experiments described in this paper show that it is possible to achieve nearly perfect detection rates and false positive rates using a data representation that discards the relationship between system call and originating process as well as the sequence structure of the calls within the traces.

Forrest et. al. [1, 3] showed that it is possible to achieve accurate anomaly detection using fixed-length contiguous subsequence representation of input data. In their approach, the detector will find anomalous subsequences right after they are executed depending on user-specified thresholds. The proposed '*bag of system calls* representation has advantage of fast learning, low memory requirement for training classifiers. A simple counter program can be used to discriminate normal sequences and abnormal sequences very quickly, before the process is terminated.

We limit our discussion for misuse detection in this paper. Additional experimental results and detailed discussions including an application to anomaly detection can be found in [5].

References

1. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for unix processes. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Society (1996) 120
2. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security **6** (1998) 151–180
3. Warrender, C., Forrest, S., Pearlmutter, B.A.: Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on Security and Privacy. (1999) 133–145
4. Lippmann, R., Cunningham, R.K., Fried, D.J., Graf, I., Kendall, K.R., Webster, S.E., Zissman, M.A.: Results of the darpa 1998 offline intrusion detection evaluation. In: Recent Advances in Intrusion Detection. (1999)
5. Kang, D.K., Fuller, D., Honavar, V.: Learning classifiers for misuse and anomaly detection using a bag of system calls representation. Technical Report 05-06, Iowa State University (2005)