Efficient Algorithms for Intrusion detection

Niranjan K. Boora[1], Chiranjib Bhattacharyya[2], and K. Gopinath[2]

[1] Dept. of Electrical Engineering,
[2] Dept. of Computer Science & Automation,
Indian Institute of Science, Bangalore, India.

Abstract. Detecting *user to root* attacks is an important intrusion detection task. This paper uses a mix of spectrum kernels and probabilistic suffix trees, as a possible solution for detecting such intrusions efficiently. Experimental results on two real world datasets show that the proposed approach outperforms the state of the art Fisher Kernel based methods in terms of speed with no loss of accuracy.

1 Introduction

Intrusion[1] can be defined as an attempt to either (i) access unauthorized information, or (ii) manipulate information, or (iii) render a system unreliable or unusable. Such Intrusions can be further categorized into Denial of Service, User to Root Attacks, Remote to User Attacks, and Probing[2]. In this paper we address the problem of detecting User to Root attacks. This attack happens when an unauthorized user gets root privileges. To detect such attacks, it maybe useful to study system audit data. System audit data refers to an ordered sequence of system calls, also known as system call traces, made by a privileged program accessible only by the root. The underlying assumption in studying audit data to detect user to root attacks is that privileged programs would behave differently when the system is compromised. The intrusion detection problem can be posed as that of deciding whether a given system call trace is due to normal mode of operation or that of a compromised system.

Given historical data, one can use various pattern recognition techniques to design classifiers to solve such problems. In the context of this paper, such classifiers will be called Intrusion detection systems (IDS) ([3, 4]). Designing such IDS for system audit data can be posed as the problem of classifying sequences, which has a rich literature. Most methods for sequence classification are based on fitting probabilistic models, like Markov Chains, Hidden Markov models (HMMs) etc, to model the class-conditional densities, and then the likelihood is used to compute the class label of a given system trace. However such methods do not yield good discriminators for intrusion detection datasets. Recently [5] (see references therein for other approaches) proposed to use support vector machines(SVMs) along with HMMs to classify system call traces. This proposal uses a scheme suggested by [6] for classifying protein sequences. This method is accurate but extremely slow in deciding the class of a given system trace.

The main contribution of the paper is to examine two methods, namely Spectrum Kernels with SVMs [7] and Probabilistic Suffix Trees (PSTs)[8] proposed

in the context of computational biology. These schemes do not attempt to model the class conditional densities very accurately yet they have good discriminative power; they are also fast. Our experiments on two real world datasets show that a hybrid of these two approaches is faster and more accurate.

The paper is organized as follows. In the next section, we discuss Fisher kernels. In Section 3, we discuss Spectrum Kernels. In Section 4, PSTs and the hybrid method is outlined. In Section 5, we discuss experimental results.

2 Fisher Kernels

2.1 Hidden Markov Models

Hidden Markov Model (HMM) ([9–11]) is a generative model to handle variable length strings, i.e., the traces of system calls. HMM is characterized by the transition, emission, and initial probabilities. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$ be the set of distinct system calls and $S = \{s_1, s_2, \ldots, s_N\}$ be the set of states. The Hidden Markov Model is represented as $\lambda = \{A, B, \pi\}$, where $A = \{a_{ij}\}$, $B = \{b_j(k)\}$, and $\pi = \{\pi_i\}$. Let a trace be represented as $O = \{O_1, \ldots, O_T\}$.
$P(X(t+1) = s_j | X(t) = s_i) = a_{ij}$
$P(O(t) = v_k | X(t) = s_j) = b_j(k)$
$P(X(1) = s_i) = \pi_i$
where $X(t) = s_i$ represents that the model is in state s_i at time t and $O(t) = v_l$ represents, at time t the system call observed is v_l. HMM is a powerful probabilistic model and is well suited to fit the variable length system call traces. Using Baum-Welch algorithm, one can efficiently learn HMMs from a given set of sequences.

2.2 Support Vector Machines

The data for 2 class classification problem is often specified by a dataset $D = \{(x_i, y_i) | x_i \in R^n, y_i \in \{1, -1\}\ i = \{1, 2, \ldots, l\}\}$ which consist of observation vectors x_i, and their class labels y_i. Given D, the learning problem is to find a decision function which will predict y given a novel observation x.

Support Vector Machines (SVMs) ([12]) solves the learning problem by posing it as the following convex quadratic optimization problem:

$$\max_{\mu_i} \sum_{i=1}^{l} \mu_i - \frac{1}{2} \sum_{i,j=1}^{l} \mu_i \mu_j y_i y_j K(x_i, x_j)$$
$$\text{subject to} \qquad 0 \le \mu_i \le C, \forall i$$
$$\sum_{i=1}^{l} \mu_i y_i = 0 \qquad (1)$$

The function $K : R^n \times R^n \to R$ is called the Kernel function and should be positive definite[12]. Let $\{\mu_i^*\}$ be the optimal values for the problem (1). The decision function is constructed from $\{\mu_i^*\}$ as follows $f(x) = sign(\sum_{x_i \in S} \mu_i^* y_i K(x_i, x) + b^*)$ where $S = \{x_i : 0 < \mu_i^* < C\}$ is the set of support vectors. This powerful classification tool is applicable only to real valued data. However this methodology can still be used if a kernel function can be defined over any two pair of sequences. Recently some progress in this direction has been made in the area of computational biology [13, 7, 14] which can be applied for designing IDS.

2.3 Fisher Score

The Fisher Score is a vector of parameter derivatives of log-likelihood of a probabilistic model. The *Fisher Score* is given by

$$U_X = \nabla_\Theta \log P(X|\lambda) \qquad (2)$$

If we consider the probabilistic model to be HMM (λ), the parameters or sufficient statistics Θ becomes the set of transition and emission probabilities and each component of U_O is a derivative of the log-likelihood probability of the sequence O, given the parameters of the HMM. The vector U can be used to represent each sequence O by a vector. Using this vectorial representation, various kernels can be defined, e.g. $K(O^i, O^j) = U(O^i)^T U(O^j)$.

The components of this vector associated with the emission probabilities can be computed as [13] $U_{kj} = \frac{E_j(k)}{b_j(k)} - \sum_{m=1}^{M} E_j(m)$, where $E_j(k)$ is the number of times symbol k is observed in state j. Likewise, for transition probabilities, the components are calculated as $V_{ij} = \frac{T_{ij}}{a_{ij}} - \sum_{k=1}^{N} T_{ik}$ where T_{ij} is the number of times transition to state j is taken from state i.

3 Spectrum Kernel

Spectrum kernel is a kernel function [7] defined on the input space $\mathcal{O}$ of all finite length sequences of characters from an alphabet $\mathcal{A}$ of size $|\mathcal{A}|$. Given a number $k \geq 1$, the k-spectrum of a sequence is the set of all k-length (contiguous) subsequences that it contains. The k-spectrum kernel is given as $K_k(O^i, O^j) = \Phi_k(O^i)^T \Phi_k(O^j)$ The feature map is defined from $\mathcal{O}$ to $R^{|\mathcal{A}|^k}$ as $\Phi_k(O^i) = (\Phi_a(O^i))_{a \in A^k}$ where $\Phi_a(O^i)$=number of times a occurs in O^i and $\mathcal{A}^k$ is the set of all possible sequences of length k of an alphabet of size $|\mathcal{A}|$. In this paper, the normalized kernel K_k is used

$$K_k^{Norm}(O^i, O^j) = \frac{K_k(O^i, O^j)}{\sqrt{K_k(O^i, O^i)}\sqrt{K_k(O^j, O^j)}}. \qquad (3)$$

For computing the kernel values, one needs to build a suffix tree for the collection of k-length subsequences of O^i and O^j, obtained by moving a k-length sliding window across each of O^i and O^j. At each depth-k leaf node of the suffix tree, store two counts, one representing the number of times k-length subsequence of O^i end at the leaf and the other count represents k-length subsequence of O^j end at the same leaf. Once the suffix tree is constructed, the kernel values are obtained by traversing the suffix tree and computing the sum of the products of the counts stored at the depth-k leaf nodes.

4 Probabilistic Suffix Trees

A PST is an n-ary tree whose root node gives the probability of each symbol of the alphabet while the nodes at subsequent levels give next-symbol probabilities conditioned on a combination of one or more symbols having been seen earlier.

The probabilities are estimated by relative frequency-counts of symbol occurrences in the training sequences. The tree depth is kept to a minimum by excluding nodes that do not provide stochastic information not already given by existing nodes. Each edge in the tree is represented by a symbol of the alphabet. No two edges emanate from any node having the same symbol, which bounds the degree by the alphabet size. Each node is assigned a string which can be generated by traversing up the tree from that node to the root. Thus PSTs are characterized by the parameter L, the maximal length of a possible string in the PST; in other words, the memory length of the PST. For a more detailed review on PSTs see [14].

4.1 SVM with Spectrum Kernel and PST

We propose a hybrid method which is faster than both Spectrum kernel and PSTs. As a first step, we train SVM with Spectrum Kernel to obtain the support vectors. To test a new trace O (to check whether it is normal or abnormal), n kernel values are to be computed, if there are n support vectors. The computation of each kernel value, $K(O, O^i)$, where O^i is the ith support vector, proceeds by first constructing a suffix tree, then traversing the tree to evaluate the kernel value. The time required in building the suffix tree dominates the prediction time for each sequence. It would be thus interesting to explore the idea of building 2 PSTs rather than n suffix trees and use these 2 PSTs to make predictions. The underlying intuition is that the support vectors have enough discriminatory information which could be efficiently represented by suffix trees. Two PSTs are trained on the support vectors. A normal PST model is trained using support vectors that are from normal class and an abnormal PST model is trained using support vectors that are from the other class. The classification of test traces is carried out by calculating the log-likelihood with respect to the two models and deciding accordingly.

5 Experimental Results

A good IDS should be able to classify all intrusions correctly. This is characterized by Hit Rate (HR) defined as probability of IDS correctly classifying a trace that belongs to the abnormal class. It can be measured on a test set by c_a/l_a where l_a is the number of abnormal traces in the test set and c_a is the number of such traces correctly classified. Having a high HR means having a low false negative error. Another requirement of a IDS is it should have a low False Alarm (FA) rate, defined as the probability of misclassifying a normal trace. It is measured on a test data by m_n/l_n where l_n is the total number of normal traces while m_n is the total number of such traces which are misclassified. A good IDS should be able to quickly decide the class of a trace. We propose to measure this by the average time t taken to classify a trace over all the traces in a test set.

For our experiments, we have chosen two system call datasets, namely the MIT Live lpr and the UNM Live lpr that are publicly available [15]. In the experiments, 0.5 fraction of dataset (0.5 fraction of normal traces and 0.5 fraction of abnormal traces) was randomly selected for training and the remaining fraction was used as a test set.

The experimental results for SVM with spectrum kernel and PST in terms of the three parameters HR, FA, t are presented in Table 1. For comparison, experimental results are also given for HMM + Fisher Score + SVM in Table 2 and PSTs in Table 3. The parameter C (see equation 1) in SVMlight software is set to 100 throughout this work. All the experiments have been done on a computer system powered by a Intel 2.4GHz processor with 1GB RAM. The code for Spectrum Kernel and PST have been written in C language. SVMlight software [16] has been used for support vector machines and GHMM software [17] for HMM modeling.

	MIT	UNM
HR 1	SKL={10,15,20,25}	All values of SKLs
FA 0	All values of SKLs	All values of SKLs
t	0.5 -1 milliseconds	0.3 - 1 milliseconds

Table 1. *SVM with Spectrum Kernel and PST* results as spectrum kernel length SKL is varied from 5 to 30 in steps of 5

The parameter L associated with PST was varied from 5 to 30 in steps of 5. The best choice was $L = 10$ and SKL= 10. In this case, $HR = 1, FA = 0$. As the

	MIT	UNM
HR 1	{5,10,15,20,25,35}	{10,15,20,25,30,35}
HR 0.998004	30	5
FA 0	{10,20,25,30}	-
FA	5: 0.003698, 15: 0.009615 35: 0.002219	{5,10,15,20,30}: 0.000465 {25,30}: 0.002327
t	8 -140 milliseconds	9 -135 milliseconds to

Table 2. Anomaly Detection using HMM + Fisher Score + SVM. The testing times are reported for all values of hidden states (HS) from 5 to 30 in steps of 5. The row corresponding to FA, under MIT or UNM column with the pattern x: y means FA achieved for HS equal to x. HS varies from 5 to 35 in steps of 5.

tables show, one can obtain the same accuracy for other methods, but the hybrid of Spectrum kernel and PST has far smaller value of t. This demonstrates that the proposed method is quicker than the other state of the art methods.

	MIT	UNM
HR 0.998004	{5,10,15}	All values of L
HR 1	{20,25,30}	-
FA	For all L, 0.002959	For all L, 0.002792
t	$0.5 - 2$ milliseconds	$0.3 - 3$ milliseconds

Table 3. PST results as L varied from 5 to 30 in steps of 5

6 Conclusions

A hybrid of spectrum kernel and probabilistic suffix trees outperforms state of the art Fisher kernel methods. The accuracy of the proposed classifier is same as that of the Fisher kernel. The utility of probabilistic suffix trees and spectrum kernels as possible solutions to intrusion detection tasks is also demonstrated.

References

1. J. P. Anderson, "Computer security threat monitoring and surveillance," tech. rep., James P Anderson Co., Fort Washington, Pennsylvania, April 1980.
2. K. Kendall, "A database of computer attacks for the evalutation of intrusion detection," Master's thesis, MIT, June 1999.
3. S. Axelsson, "Intrusion detection systems: A survey and taxonomy," tech. rep., Department of Computer Engineering, Chalmers University of Technology, 2000.
4. A. Sundaram, "An introduction to intrusion detection," *ACM Crossroads Student Magazine*, 1996.
5. J. Baras and M. Rabi, "Intrusion detection with support vector machines and generative models," tech. rep., Institute for Systems Research, University of Maryland, 2002.
6. T. Jaakkola and D. Haussler, "Using the Fisher kernel methods to detect remote protein homologies," in *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pp. 149–58, 1999.
7. C. Leslie, E. Eskin, and W. Stafford, "The spectrum kernel: A string kernel for SVM protein classification," in *Proceedings of the Pacific Symposium on Biocomputing*, pp. 564–575, Jan 2002.
8. D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25(2-3), pp. 117–149, 1996.
9. L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
10. L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4–15, January 1986.
11. R. Duggad and U. B. Desai, "A tutorial on hidden Markov models," tech. rep., Electrical Department, Indian Institute of Technology, Bombay, 1996.
12. C. J. C. Burges, "A tutorial on support vector machine for pattern recognition," in *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.
13. P. Pavlidis, T. S. Furey, M. Liberto, D. Haussler, and W. N. Grundy, "Promoter region-based classification of genes," in *Proceedings of the Pacific Symposium on Biocomputing*, pp. 151–163, January 2001.
14. G. Bejerano and G. Yona, "Variations on probabilistic suffix trees: statistical modeling and prediction of protein families," *Bioinformatics*, vol. 17, no. 1, pp. 23–43, 2001.
15. UNM, Department of Computer Science, "Computer immune systems homepage," http://www.cs.unm.edu/immsec/systemcalls.htm.
16. T. Joachims, "Making large-scale SVM learning practical," *Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.)*, 1999.
17. M. P. I. f. M. G. Algorithmics group, "General hidden Markov model library (ghmm)," http://sourceforge.net/projects/ghmm/.