View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

Novel machine learning techniques for anomaly intrusion detection

Yanxin Wang
Iowa State University

Johnny Wang
Iowa State University

Andrew Miner
Iowa State University

Follow this and additional works at: http://aisel.aisnet.org/amcis2004

Recommended Citation
Wang, Yanxin; Wang, Johnny; and Miner, Andrew, "Novel machine learning techniques for anomaly intrusion detection" (2004). *AMCIS 2004 Proceedings*. 551.
http://aisel.aisnet.org/amcis2004/551

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Novel machine learning techniques for anomaly intrusion detection

Yanxin Wang
wangyx@iastate.edu

Johnny Wang
wong@cs.istate.edu

Andrew Miner
asminer@cs.istate.edu

ABSTRACT

This paper explores the methodology of using kernels and Support Vector Machine (SVM) for intrusion detection. A new insight into two well known anomaly detection algorithms - STIDE and Markov Chain anomaly detectors, is achieved using kernel theory.

We introduce two new classes of kernels used for intrusion detection – STIDE kernel and Markov Chain kernel. These kernels combined with SVM are presented to achieve improvements over STIDE and Markov Chain anomaly detectors. We provide empirical evidence that the new anomaly detectors are able to achieve better results than conventional anomaly detectors and behave robustly over noisy training data.

Keywords

Anomaly intrusion detection, Support Vector Machine, kernel method.

INTRODUCTION

The growing number of attacks on computers and networks has made the study of intrusion detection techniques a research priority. Intrusion detection aims to find the activities that compromise the integrity, confidentiality or availability of a system. There are many existing techniques for intrusion detection, which are classified into two classes: misuse detection and anomaly detection.

In misuse detection, the sequences of events that characterize an attack must be defined by experts in the intrusion detection domain. System behaviors are compared to the known attack patterns, also called intrusion signatures, to find attacks. Misuse detection can detect known attacks accurately; however, it has no ability over the novel attacks.

Anomaly detection, on the other hand, detects attacks based on normal profiles of users or systems. This approach typically consists of two phases. In the training phase, the normal profile is built based on user or system behaviors in normal situation. Statistical methods and machine learning methods are often used in building and learning the profile from normal system behaviors. In the testing phase, the system behaviors are compared to the normal profile. When a sufficient deviation from the normal profile is found, an attack alarm is raised. Unlike misuse detection, anomaly detection can catch previously unknown attacks. However, the accuracy of anomaly detection depends on the exactness of normal profiles. Because of following reasons, the normal profiles are usually not accurate and can cause high false alarm rates:

- Normal training data is not sufficient to cover all the normal behaviors.
- Normal system and user behaviors change over time.
- Attackers can insert their behaviors slowly, so the normal data becomes noisy with intrusion data. The anomaly detector trained with the noisy data will accept some attacker activities as normal.

STIDE, developed at University of New Mexico, and Markov Chain anomaly detectors, introduced by Carnegie Mellon University, are two well-known anomaly detectors for UNIX privileged system programs [1, 2]. STIDE and Markov Chain anomaly detectors work very well when a testing data set is similar to the training data set, but have the problem of high false alarm rates when only limited normal training data is provided.

A new insight is provided into the two well known anomaly detection algorithms - STIDE and Markov Chain anomaly detectors using the kernel theory. This paper starts from the analysis of STIDE and Markov Chain anomaly detectors, and answers the following questions:

- What is the theoretical basis of STIDE and Markov Chain anomaly detectors from the statistical learning point of view?
- What limitations do STIDE and Markov Chain anomaly detectors have?
- How can we improve based on STIDE and Markov Chain anomaly detectors?

This paper then presents two anomaly detectors which are based on kernel method and Support Vector Machine (SVM) learning algorithms and have a higher detection rate (rate of identified attacks) and lower false alarm rate (rate of normal instances classified as anomalies) than STIDE and Markov Chain anomaly detectors.

An overview of kernel method and SVM is given first. Then we present the analysis of STIDE and Markov Chain anomaly detector using kernel theory and introduce STIDE kernel and Markov Chain kernel based SVM for anomaly detection. After that, the experimental results is demonstrated. Then we conclude the paper with future research direction.

BACKGROUND

Kernel methods

In some optimization functions, e.g, dual presentation, the data points appear only as dot products. A kernel function is defined to return the value of the dot product between the images of the two arguments: $K(x1, x2) = \langle \Phi(x1), \Phi(x2) \rangle$, where $\Phi(x)$ is a mapping function from one feature space F to another feature space H. The kernel function can be used without knowing the mapping function or the feature space mapped to. The kernel function can facilitate the use of sparse feature space [3]. Many discriminative machine learning algorithms, such as SVM and Probably Approximately Correct (PAC) algorithm [4, 5], make use of kernel methods. Using these kernel based learning algorithms, only dot products of feature vectors need to be calculated. The high dimensional feature vectors need not to be calculated. This leads to the advantage of computational efficiency.

The most popular kernel functions include linear kernels, polynomial kernels and radial basis kernels [3]. The linear kernels $k(x,y) = \langle x, y \rangle$ are the simplest kernel function. In a linear kernel, an input vector maps to itself, i.e, $\Phi(x) = x$. The polynomial kernels $k(x,y) = \langle x, y \rangle^d$ are more complicated than linear kernel, and they usually have better generalization performance than linear kernels in classification [3]. The radial basis kernels $k(x,y)=e^{-||x-y||^2/2\delta}$ are more complicated than polynomial kernels. These kernels are called standard kernels since they are not specifically designed for one domain, instead they can be used in many domains, such as natural language processing, speech recognition and computational biology.

However, recently, there has been considerable interest in the development of application based kernels for specific applications, for instance, protein classification in computational biology [6]. A family of string kernels for protein classification are introduced, such as spectrum kernels, restricted gappy kernels, substitution kernels and wildcard kernels [7]. They are demonstrated to be very effective for matching protein types.

Support Vector Machine

SVM was first introduced by Vapnik [9] and has been used with great success for pattern recognition fields, such as handwriting recognition, speech recognition and protein homology detection [4, 6]. Given a set of labeled training vectors, SVMs learn a linear classification function. SVMs have viewed the classification problem as a quadratic optimization problem. It has exhibited excellent accuracy on test sets in practice and have strong theoretical motivation in statistical learning theory [10].

SVM method originates from the perceptron algorithm. Perceptron algorithm works by adding misclassified positive or subtracting misclassified negative examples to an arbitrary initial weight vector. In perceptron based learning, the learning problem is ill posed: finding one hyperplane that separates the input instances into one of two classes correctly in feature space. Many such hyperplanes exist. A principled way to choose the best possible hyperplane is needed. According to statistical learning theory, a hyperplane (classifier) with the best generalization ability is considered as the best possible hyperplane.

Generalization ability is the "Capacity" of a classifier - the ability to learn ANY training set without error. The bounds on error of classification suggest the possibility of improving generalization by maximizing the margin. The margin

is the distance from the closest examples to the hyperplane. The risk of overfitting, which means classifiers drawn from the training data behave considerably better on the training instances than the testing instances, can be minimized by choosing the maximal margin hyperplane in feature space [4]. SVMs improve "Capacity" by increasing the margin and lowering the classification error, and this optimization problem is solved using the Sequential Minimal Optimization algorithm. More details about SVM can be found in [11].

Some research on SVM explored how to generate new features based on training data for intrusion detection [13], and some research studied how to select important features to improve the efficiency of learning in intrusion detection [14]. Earlier research has also studied using SVM and standard kernels, such as radial basis kernel and polynomial kernel, for intrusion detection [15]. The essentials of kernel method is to map data into a linearly separable feature space. In an ideal mapping, the high dimension feature space should represent more structure, which means knowledge in this domain. In intrusion detection domain, the knowledge should include order and time information. This paper focuses on exploring the kernels specially for anomaly detection, and combine the kernels with SVM to detect anomaly.

MODEL OF STIDE ANOMALY DETECTOR

We define a kernel based on STIDE anomaly detector and also present an approach of combining the new kernel with one-class SVM for anomaly detection.

STIDE anomaly detector

In anomaly detection for system call sequences, an alphabet is used to represent system calls; a sequence of alphabet corresponds to a sequence of system calls, and sequences are collected using trace utility in UNIX. Normal sequences are obtained from normal execution of a program and abnormal sequences are generated from abnormal execution of a program.

STIDE anomaly detector uses a fixed-size sliding window of size k to generate all the subsequences with size k from the training data to form a normal k-size subsequences database [16]. Given a sequence of testing data, anomalies are detected by sliding a window of size k along the testing data: if the number of subsequences of size k not existing in the normal database is bigger than a certain predetermined anomaly threshold, δ, then the detector declares that an anomaly has occurred, and the current position in the test data can be used to provide information about where the anomaly occurred.

Earlier research results proved that this STIDE anomaly detector has limited detection coverage [2]. Following the definitions in [2], a foreign sequence is a sequence whose individual symbols appear in the training sequence, but the foreign sequence does not itself appear in the training sequence. We say a sequence is normal if it is not foreign. A minimal foreign sequence is a foreign sequence whose proper subsequences are all normal. The STIDE anomaly detector has the following detection coverage [2]:

1. If the largest minimal foreign sequence in the anomaly space has length M, then a STIDE anomaly detector with window size N >= M can correctly classify any sequence.

2. If the anomaly space contains a minimal foreign sequence of length M, then a STIDE anomaly detector with window size N < M cannot correctly classify all sequences.

STIDE kernel

Let alphabet set Z represents all the possible system calls that appear in the privileged daemon and k be the sliding window size used by the STIDE model. The input space Z^* is a space of sequences of characters from the alphabet Z. The k-STIDE kernel maps from Z^* to a $|Z|^k$ -dimensional feature vector space.

The feature map is defined as follows:

$\Phi_k(s) = (e(s_k))_{sk \in A}$

where $s \in Z^*$; $A \subseteq Z^k$; $e(s_k)$ is 1 when s_k is a subsequence of sequence s or 0 otherwise. The k-STIDE kernel is then:

$K_k(s1, s2) =< \Phi_k(s1), \Phi_k(s2) >$

According to the definition in [8], the k-STIDE kernel K is a PDS kernel since $K(x, y) =< \Phi_k(x), \Phi_k(y) >$ holds with the mapping function $\Phi_k(s)$.

From kernel method point of view, the original STIDE anomaly detector [16] can be expressed based on the kernel we defined. The STIDE anomaly detector uses the feature mapping defined above to calculate a vector $(e(s_k))_{sk \in A}$, where A is the set of all k-size subsequences from alphabet Z that exist in the privileged daemons; $e(s_k)$ is 1 when s_k is a subsequence of s or 0 otherwise.

Also, the STIDE anomaly detector uses a simple linear threshold function with the mapping method described above.

In the training phase, a vector v is calculated: $v(s) = (e(s_k))_{sk \in A}$

where $s \in Z^*$; $A \subseteq Z_k$; $e(s_k)$ is 1 when s_k is a subsequence of s or 0 otherwise.

In the testing phase, for each sequence s in testing data, a vector v' is calculated as $v' = \Phi_k(s)$.

The dot product of v' and the vector v is compared against an anomaly threshold δ. When there is one dot product bigger than or equal to the anomaly threshold, an anomaly is raised, i.e., the linear threshold function for anomaly detection is:

$\langle v', v \rangle >= \delta$, anomaly; $\langle v', v \rangle < \delta$, normal.

Using STIDE anomaly detector, consider following example:

alphabet={a, b, c}, window size k=2

New feature space Z^k : {aa, ab, ac, ba, bb, bc, ca, cb, cc}

Training phase:

Normal processes: abbabb, bcccc

N={ab,ba,bb,bc,cc} is the normal sequences database, so v = (0, 1, 0, 1, 1, 1, 0, 0, 1)

Linear threshold function: $\langle v', v \rangle >= 2$, anomaly; otherwise, normal (anomaly threshold is 2).

Testing phase:

abcca ? v' = (1, 0, 0, 0, 0, 1, 1, 0, 1), so $\langle v', v \rangle = 1$, it is normal.

aacca ? v' = (0, 1, 1, 0, 0, 0, 1, 0, 1) , so $\langle v', v \rangle = 3$, it is abnormal.

Thus, STIDE anomaly detector compares every new instance with a standard abnormal vector, when the number of matches is bigger than a predetermined threshold, an alarm is raised.

It can be seen that the STIDE anomaly detector simply memorizes the observed patterns and all mismatches are considered as abnormalities. When the number of abnormalities are over a threshold, an anomaly is reported. Therefore, STIDE anomaly detector does not provide good generalization ability, which causes high false alarm rate since unseen normal patterns are classified as abnormalities. Thus, we consider to use Support Vector Machine to substitute STIDE's threshold approach and at the same time keep the kernel function(feature mapping) of STIDE anomaly detector. Because Support Vector Machine method aims to maximize its generalization capability, SVM can achieve less false alarm rate than STIDE, when SVM also uses STIDE kernel.

Efficient computation of STIDE kernel

Since the system call sequence length (same as the system call sequence) is k and the alphabet is Z, there can be $|Z|^k$ permutation of all possible system call sequences, which means, the vector length is $|Z|^k$. To calculate STIDE kernel, the whole vector needs to be calculated. So the algorithm complexity is $O(|Z|^k)$, which increases exponentially with sliding window size k and can be impractical to be calculated when k get higher than 4 with alphabet size above 100.

To calculate k-STIDE kernel efficiently, all the subsequences of size k that do not exist in the log file of the privileged system program can be neglected. This saves tremendous space and computation time. Thus, the complexity of the k-STIDE kernel method is reduced to O(|N|) where N is the set of all the subsequences with size k that exist in the traces of the privileged system program.

MODEL OF MARKOV CHAIN ANOMALY DETECTOR

Markov Chain models have been used for anomaly detection for system call sequences of privileged system programs in UNIX [2]. We define a Markov Chain kernel in this section.

Markov Chain anomaly detector

The Markov-based detector described in [2] uses a fixed-size detector window of size k to build a discrete-time Markov chain, as follows. Each state of the Markov chain is a sequence with size k appearing in the training data. The probability of transition from state $s = (a_1, a_2, \ldots, a_k)$ to state $s' = (a_2, \ldots, a_k, a_{k+1})$ is computed as $p(s, s') = F(s,s')/F(s)$, where F (s, s') is the number of times the sequence $(a_1, \ldots, a_{k+1})$ appears in the training data, and F (s) is the number of times the sequence $(a_1, \ldots, a_k)$ appears in the training data. Given a sequence of test data, anomalies are detected by sliding a window of size k along the test data: the current state s is given by the current k symbols, the next state s' is given by the k symbols after sliding the window by one position, and the surprise factor is calculated as $1 - p(s, s')$. If the surprise factor is above a selected anomaly threshold, then the detector declares that an anomaly has occurred, and the current position in the testing data can be used to provide information about where the anomaly occurred.

Early works have proved this Markov Chain anomaly detector can achieve better detection accuracy than the STIDE model [2]. The reason is Markov Chain model builds the normality by recording not only which subsequences are normal, but also the probability of occurrences of the subsequences. We have proved the Markov Chain anomaly detector has the following detection coverage [17]:

1. If the largest minimal foreign sequence in the anomaly space has length M, then a Markov-based detector with window size $N >= M - 1$ can correctly classify any sequence.

2. If the anomaly space contains a minimal foreign sequence of length M, then a Markov-based detector with window size $N < M - 1$ cannot correctly classify all sequences.

Markov Chain kernel

Let alphabet set Z represents all the possible system calls that appear in the privileged system program and k be the sliding window size used by the Markov Chain model. The input space Z^* is a space of sequences of characters from the alphabet Z. The k-Markov Chain kernel maps from Z^* to a $(|Z|^k)^2$ -dimensional feature vector space.

The feature map is defined as follows: $\Phi_k(s) = (p(s_k, s'_k)_{s_k, s'_k \in N})$, where $s \in Z^*$; $A \subseteq Z^k$; s_k, s'_k are two states in Markov Chain, and each of them represents a k-size sequence from alphabet Z; N are the state set of the Markov Chain; $p(s_k, s'_k)$ is the probability of transition from state s_k to state s'_k in sequence s.
The k-Markov Chain kernel is: $K_k(s1, s2) = <\Phi_k(s1), \Phi_k(s2)>$.
According to the definition in [8], a kernel K is a PDS kernel provided that there exists a mapping function Φ that $K(x, y) = <\Phi(x), \Phi(y)>$ holds. The k-Markov Chain kernel K is a PDS kernel since the above mapping function $\Phi_k(s)$ maps from a system call sequence space to another system call sequence feature space and the Markov Chain kernel $K(x, y) = <\Phi(x), \Phi(y)>$.

The original Markov Chain anomaly detector can be described using the feature mapping we defined above to calculate a probability vector $(p(s, a'))_{a' \in A}$ where s is a state that represents a k-size sequence from alphabet Z that exists in the privileged daemons and p(s, a') is the probability of transition from the state $s = (a_1, a_2, \ldots, a_k)$ to the state $s' = (a_2, \ldots, a_k, a')$.

The Markov Chain anomaly detector uses a simple linear threshold function. In the training phase, a family of probability vectors are calculated: $v_i = (p(s, a'))_{a' \in Z}$, $1 <= i <= |N| * |Z|$ where s is a state that represents a k-size sequence from alphabet Z that exists in the program execution; p(s, a') is the probability of transition from state $s = (a_1, a_2, \ldots, a_k)$ to state $s_0 = (a_2, \ldots, a_k, a')$ if the index of p(s, a') in the vector is also i, or 0 otherwise. So for this family of probability vectors, there is at most one non-zero element in each vector.

In the testing phase, for each system sequence s in the testing data, a probability vector v, which represents the features of the subsequence transition probability in s, is calculated using the feature mapping: $v = \Phi_k(s)$ as described earlier.

The dot product of v' and each vector v in the family of the probability vectors is compared against a preselected anomaly threshold δ. When there is a dot product value bigger than or equal to the anomaly threshold, an anomaly is reported, i.e., the linear threshold function for anomaly detection is: $\min\{<v', v>\} >= \delta$, anomaly;

otherwise, normal.

Efficient computation of Markov Chain kernel

For Markov Chain method, since the system call sequence length (same as the system call sequence) is k and the alphabet size is $|Z|$, there can be $|Z|^k$ permutation of all possible system call sequences. The Markov Chain method needs to calculate all the possible transitions between each system call sequence, which means, the vector length is $O((|Z|^k)^2)$, so the algorithm complexity is $O((|Z|^k)^2)$. This computation soon becomes infeasible with the increase of the size of the sliding window.

To calculate k-Markov Chain kernel efficiently, all the subsequences of size k that do not exist in the traces of privileged system program in training and testing data sets can be neglected as with the STIDE kernel. For state $s_k = (a_1, a_2, \ldots, a_k)$, only states $s' = (a_2, \ldots, a_k, a')$ where $a' \in Z$ can be its subsequent state. So $p(s_k, s'_k)$ can be represented by $p(s_k, a')$, which can be computed as $F(s_k, a')/F(s_k)$, where $F(s_k, a')$ is the number of times the sequence $(a_1, a_2, \ldots, a_k, a')$ appears in the training data, and $F(s_k)$ is the number of times the sequence $(a_1, a_2, \ldots, a_k)$ appears in the training data. Thus, the complexity of the k-Markov Chain kernel method is reduced to $O(|A|*|Z|)$ where A is the set of all the subsequences with size-k that exist in the privileged system program.

DISADVANTAGES OF STIDE AND MARKOV CHAIN ANOMALY DETECTORS

The disadvantage of STIDE and Markov Chain anomaly detectors is their over-simplicity. They completely depend on the training data thus have the over-fitting problem. By substituting the simple linear threshold function with SVM, which is proved to be very effective in statistical learning, improvement can be achieved over the STIDE and Markov Chain anomaly detectors. Using SVM method, the generalization capability is maximized, so the overfitting problem is overcomed greatly.

Another disadvantage of STIDE and Markov Chain anomaly detectors is the selection of a threshold. The threshold is critical in these two approaches, however, there is no efficient and effective method for selection of a proper threshold. Using SVM method combined with STIDE and Markov Chain kernel, the threshold is no longer needed.

EXPERIMENT ON SVM BASED ANOMALY DETECTORS

STIDE and Markov Chain kernel are both PDS kernels, or they satisfy Mercer's condition, which guarantees the convergence of a SVM algorithm. We use a publicly available SVM software SVMlight, which is an implementation of Vanik's Support Vector Machine. The optimization algorithm used in SVMlight are described in [18]. The algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently.

We test our approaches and compare them with the STIDE and Markov Chain anomaly detectors on the synthetic and live sendmail data set of UNM [19]. UNM sendmail data includes hundreds of normal execution of sendmail daemons and several traces of sendmail daemons that are attacked. In each experiment, 10 normal traces and 5 abnormal traces are randomly drawn from this UNM data set. We can employ more traces in training but that only 10 normal traces and 5 abnormal traces since more training data We compare the performance of the four anomaly detectors - STIDE, Markov Chain detector, STIDE kernel based SVM and Markov kernel based SVM - on UNM synthetic and live sendmail data. Table 1 shows the comparison results using the UNM data set. The experiment demonstrates STIDE and Markov kernel based SVM has better performance than STIDE and Markov Chain anomaly detectors.

We also test the combination of SVM with polynomial kernel and radial basis kernel. For the SVM with polynomial and radial basis kernel, we simply use system call as feature space. The input vector is $v(s) = (e(a))_{a \in Z}$ where Z represents the system call alphabet, *e(a)* is 1 when subsequence *a* exists in the normal training data and is 0 otherwise. Table 2 shows the comparison between STIDE kernel based SVM, Markov kernel based SVM, polynomial kernel based SVM and radial basis kernel based SVM using the UNM dataset.

This paper focuses on exploring meaningful kernels for anomaly detection. Obviously, STIDE kernel and Markov Chain kernel capture more knowledge, such as time and order information, than standard kernels, which explains why experiment 2 show our STIDE and Markov Chain kernel achieve better accuracy than the SVM anomaly detector based on polynomial kernels and radial basis kernels.

	Precision rate	Detection rate	False alarm
STIDE	83.9%	74%	6.2%
Markov Chain	86.8%	78%	4.5%
STIDE kernel SVM	87.7%	77.8%	2.4%
Markov kernel SVM	94%	90%	2%

Table 1. Comparison of STIDE, Markov Chain, STIDE kernel based, Markov Chain kernel based anomaly detector(using a threshold 0.0001 for one-class SVM)

	Precision rate	Detection rate	False alarm
Polynomial kernel SVM	86.7%	77%	3.6%
Radial basis kernel SVM	89.3%	81.1%	2.6%
STIDE kernel SVM	87.7%	77.8%	2.4%
Markov kernel SVM	94%	90%	2%

Table 2. Comparison of polynomial kernel, radial basis kernel, STIDE kernel and Markov Chain kernel based anomaly detector (using a threshold 0.0001 for one-class SVM)

CONCLUSION AND FUTURE WORK

This paper presents the existing methods for anomaly intrusion detection, namely, STIDE anomaly detector and Markov Chain anomaly detector, points out the weakness of the two methods, and presents our novel method using kernel method and SVM. This research investigates meaningful kernels for intrusion detection. We identify two kernels that can represent the similarity of system call sequences. We also experiment on combining STIDE kernel and Markov Chain kernel with SVM to improve the classification result. We believe our results provide strong evidence that STIDE kernels and Markov Chain kernels, in conjunction with SVMs, could offer a more accurate and effective alternative to conventional anomaly detection algorithm (STIDE and Markov Chain method) for detecting anomalies in system call sequences. The threshold is still critical for one-class SVM for the result. The automatic generation of threshold is an interesting research direction in the future.

REFERENCES

1. S. Forrest, S. A. Hofmeyer, and A. Somayaji. Computer immunology. Comm. ACM, 40(10):88-96, Oct. 1997.
2. Roy A. Maxion and Kymie M. C. Tan. Anomaly detection in embedded systems. IEEE Trans. Computers,51(2):108-120, Feb. 2002.
3. Bernhard Scholkopf and Alex Smola. Learning with Kernels. MIT Press, Cambridge, MA, 2002.

4. J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge iscovery, 2(2):121-167, 1998.
5. W. Maass. Efficient agnostic PAC learning with simple hypotheses. In Proceedings of the Seventh Annual Conference on Computational Learning Theory, pages 67-75, 1994.
6. E. Eskin C. Leslie and W.S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In the Pacific Symposium on Biocomputing, pages 564-575, 2002.
7. S.V.N. Vishwanathan and A.J. Smola. Fast kernels for strings and trees. In Neural Information Processing Systems, 2002.
8. J.P. R. Christensen C. Berg and P. Ressel. Harmonic analysis on semigroups. Springer-Verlag:Berlin-New York, 1984.
9. B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, 1992.
10. Vladimir N. Vapnik. Statistical Learning Theory. Wiley, Chichester, GB, 1998.
11. J. Platt. Fast training of support vector machines using sequential minimal optimization, pages 185-208. In B. Schlkopf, C. Burges and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, MA, 1999.
12. J. Shawe-Taylor A. J. Smola B. Sch¡§olkopf, J. Platt and R. C. Williamson. Estimating the support of a high-dimensional distribution. Neural Computation, 13:1443-1471, 2001.
13. Wenke Lee and Sal Stolfo. A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security, 3(4), Nov 2000.
14. S. Mukkamala A. H. Sung. Identifying important features for intrusion detection using support vector machines and neural networks. In Symposium on Applications and the Internet, 2003.
15. S Mukkamala and A H. Sung. Learning machines for intrusion detection: Support vector machines and neural networks. In International Conference on Security and Management, pages 525-531, 93.
16. S. Forrest, S. A. Hofmeyer, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. Proceedings 1996 IEEE symposium on Security and Privacy, pages 120-128, May 1996. Oakland, CA.
17. Y. Wang, A. S. Miner, and J. Wong. Comments on "Anomaly detection in embedded systems". submitted to IEEE Transactions on Computer, 2003. http://latte.cs.iastate.edu/Research/Intrusion/papers/Comm03.ps.
18. T. Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning. MIT-Press, 1999.
19. University of New Mexico. Compuer immune systems – data sets. online, 2003. http://www.cs.unm.edu/˜immsec/systemcalls.htm.