MODELING PROGRAM BEHAVIORS BY HIDDEN MARKOV MODELS FOR INTRUSION DETECTION

WEI WANG[1], XIAO-HONG GUAN[1, 2], XIANG-LIANG ZHANG[3]

[1]SKLMS (State Key Laboratory for Manufacturing Systems) and Research Center for Networked Systems and Information Security, Xi'an Jiaotong University, Xi'an 710049, China.
[2]Center for Intelligent and Networked Systems, Tsinghua University, Beijing 100084, China
[3]Department of electronic science and technology, Xi'an Jiaotong University, Xi'an 710049, China
E-MAIL: wwang@sei.xjtu.edu.cn, xhguan@tsinghua.edu.cn, zhangxl@mailei.xjtu.edu.cn

Abstract:

Intrusion detection is an important technique in the defense-in-depth network security framework and a hot topic in computer network security in recent years. In this paper, a new efficient intrusion detection method based on Hidden Markov Models (HMMs) is presented. HMMs are applied to model the normal program behaviors using traces of system calls issued by processes. The output probability of a sequence of system calls is calculated by the normal model built. If the probability of a sequence in a trace is below a certain threshold, the sequence is flagged as a mismatch. If the ratio between the mismatches and all the sequences in a trace exceeds another threshold, the trace is then considered as a possible intrusion. The method is implemented and tested on the *sendmail* system call data from the University of New Mexico. Experimental results show that the performance of the proposed method in intrusion detection is better than other methods.

Keywords:

Intrusion detection; anomaly detection; hidden markov models (HMMs); system call; program behaviors; computer security; pattern recognition

1. Introduction

Intrusion detection is an important and active area of research in computer security in recent years. In general, the techniques for intrusion detection fall into two major categories depending on the modeling methods used: misuse detection and anomaly detection. Misuse detection is usually conducted by signature matching of known attacks. Although misuse detection is very effective for detecting known attacks, it can not detect "newly invented" attacks. Anomaly detection, on the other hand, models normal behaviors and attempts to identify patterns of activities that deviate from the defined model. Anomaly detection can detect unknown intrusions and therefore is an active research area in intrusion detection. However, anomaly detection may cause a higher rate of false alarms. In practice, an Intrusion Detection System (IDS) integrate misuse detection and anomaly detection techniques to enhance the performance of the intrusion detection.

Many types of data can be used for anomaly detection, such as Unix Shell commands, audit events, keystroke, system calls, and network packages, etc. Early studies [1, 2, 3] on anomaly detection mainly focus on learning normal system or user behaviors from monitored system log or accounting log data. Examples of the information derived from these logs are: CPU usage, time of login, duration of user session, names of files accessed, etc. However, since user behaviors can change very frequently, it is difficult to model stable user behaviors and therefore the model can not be used to detect intrusions effectively.

In recent years, many research in anomaly detection focus on modeling program behaviors. Compared to user behaviors, program behaviors are more stable over time because the range of program behaviors is more limited [4]. Furthermore, it would be more difficult for attackers to break into a computer system without revealing their tracks in the execution logs [4]. Therefore, modeling program behaviors is more effective for intrusion detection and this property attracts many researchers. In 1996, Forrest et al. introduced a simple anomaly detection method based on monitoring the system calls issued by active, privileged processes. Each process is represented by the ordered list of system calls it used. In sequence time-delay embedding (STIDE), a profile of normal behaviors is built by enumerating all fixed length of unique, contiguous system calls that occur in the training data, and any violation of the defined model is considered as anomalies [5]. This work was extended by various methods. Lee et al. used data mining approach to study a sample of system call data to characterize sequences occurring in normal data by a small set of rules. During monitoring, sequences violating those rules are treated as anomalies [6]. Wespi et al. further developed Forrest's idea and proposed a variable length approach [7]. Asaka et al. proposed another approach based on the discriminant method in which an optimal classification surface is first learned from samples of the properly labeled normal and abnormal system call sequences. The surface is then used as a basis to decide the normality of a new system call sequence [8]. Rough Set Theory method is applied by our research group to learn the rule set by modeling the normal program behaviors with much smaller size of sample data set required and improved detection accuracy based on system call data [9]. Yihua et al. used kNN classifier and Wenjie et al. applied Robust support vector machines for intrusion detection [4, 10] and also get good testing results.

Existing efforts on intrusion detection have considered mainly 6 attributes of activities in computer systems, and these attributes can be categorized into three groups: duration property, ordering property and frequency property of events [11]. The ordering property of system calls is useful to model program behaviors for anomaly detection. Many methods have been used to consider the ordering information of the system calls for this end, such as first-order Markov Chain Models [12], high-order Markov Models [13] and HMMs [14-18], etc.

HMMs based intrusion detection method was first proposed by Warrender et al [14]. They used one system call at a time in a trace as observable and tracked what state transitions and outputs would be required of the HMMs to produce that system call. If a system calls in the trace which could only have been produced using below threshold transitions or outputs, it is flagged as a mismatch [14]. In this method, HMMs are making anomaly decisions at each system call rather than on sequences. This paper presents a new intrusion detection method using HMMs to model program behaviors. In this new method, sequences of system calls in a trace were used as observable. The probability that the HMMs produce a sequence of system calls was computed for anomaly detection. If the probability of a given sequence in a trace is below a certain threshold, the sequence is then flagged as a mismatch. If the ratio between the mismatches and all the sequences in a trace exceeds another predetermined threshold, the trace is considered as a possible intrusion. Experimental results on system call data show that the proposed method is promising in terms of accuracy and efficiency compared with other methods.

The rest of this paper is organized as follows. The next section gives a brief introduction of HMMs and shows how HMMs are applied to model the normal program behaviors. In section 3, the anomaly detection method is presented. Intrusion detection experiments are described and the testing results are summarized and discussed in section 4. Concluding remarks are given in section5.

2. Modeling Normal Program Behaviors Based on HMMs

An HMM describes a doubly stochastic process. Each HMM contains a finite number of unobservable (or hidden) states. Transitions among the states are governed by a set of probabilities called transition probabilities. An HMM defines two concurrent stochastic processes: the sequence of HMM states and a set of state output processes. As a machine learning method for constructing a finite state machine, HMMs have been widely used in knowledge discovery, pattern classification, speech recognition, DNA sequence modeling, and so on.

There are three central issues in HMMs including the evaluation problem, the decoding problem, and the learning problem. Given an input sequence of system calls, an HMM can model this sequence by three parameters—state transition probability distribution *A*, observation symbol probability distribution *B* and initial state distribution π [19]. Therefore, a sequence can be modeled as $\lambda = (A, B, \pi)$ using its characteristic parameters. Other parameters except *A*, *B*, π used in HMMs are defined as follows:

T = length of the sequence of observations (training set)

N = number of states in the model (we either know or guess this number)

M = number of possible observations (from the training set)

$S = \{s_1, s_2, \cdots, s_N\}$ finite set of possible states

$V = \{v_1, v_2, \cdots, v_M\}$ finite set of possible observations)

HMMs learning can be conducted by the *Baum-Welch* (BW) or *forward-backward algorithm*—an example of a generalized *expectation-Maximization* (EM) algorithm [19].

Standard HMMs have a fixed number of states, so we must decide the size of the model before training. Preliminary experiments showed that a good choice for the application was to choose a number of states roughly corresponding to the number of unique system calls used by the program [14]. Therefore choosing the number of the states depends on the dataset used in the experiments.

$\xi_t(i, j)$ is defined as the probability being in state s_i at time *t* and the state s_j at time *t*+1 and it can be written as following equations [19, 20, 21]:

$$\xi_t(i, j) = P(i_t = s_i, i_{t+1} = s_j \mid O, \lambda)$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (1)$$

Forward variable $\alpha_t(i)$ is defined as the probability that the model is in state s_i at time *t* and has generated the target sequence up to step *t*. It can be written as $\alpha_t(i) = P(O_1 \cdots O_t, i_t = s_i \mid \lambda)$.

Backward variable $\beta_t(j)$ is analogously defined to be the probability that the model is in state s_i at time *t* and will generate the remainder of the give target sequence.

$\gamma_t(i)$ is the probability with which it stays at state s_i at time *t*.

$$\gamma_t(i) = P(i_t = s_i \mid O, \lambda) = \sum_{j=1}^{N} \xi_t(i, j)$$

(2)

Given the above variables calculated, a new model $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ can be re-estimated by the following equations [20]:

$$\bar{\pi}_i = \text{expected frequency(number of times) in state } s_i \text{ at time(t = 1)} = \gamma_1(i) \qquad (3)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of tranistions from state } s_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

(4)

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} = \frac{\sum_{t=1,\ s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad (5)$$

We start with rough and arbitrary estimates π, *A* and *B*, calculate improved estimates by (3), (4) and (5), and repeat until some convergence criterion is met (e.g., sufficiently small change in the estimated values of the parameters on subsequent iterations).

After the HMMs were learned on training set of system calls, normal program behaviors can be modeled by the parameters of the HMMs $\lambda = (A, B, \pi)$.

3. Anomaly Detection

Given a testing trace of system calls (length *S*) that were generated during the execution of a process, we use a sliding window of length *L* to move along the trace and get (*S*-*L*+1) short sequences of system call $X_i (1 \le i \le S - L + 1)$.

Using the normal model $\lambda = (A, B, \pi)$ which was built by the training method described in above section, the probability that a given observation sequence *X* is generated from the model can be evaluated using the *forward algorithm* by forward variable $\alpha_t(i)$ defined in the above section.

Summing up $\alpha_t(i)$ for all *i* yields the value $P(O|\lambda)$ which can be calculated by the following procedures [20, 21]:

Step 1. Initialization

$$\alpha_1(i) = \pi_i b_i(O_1) \qquad (6)$$

Step 2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \qquad (7)$$

Step 3. Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \qquad (8)$$

In the procedure of actual calculation, we use log-probabilities and to compute $\log P(O|\lambda)$ instead of $P(O|\lambda)$ for the purpose of increasing the scale of the probability.

Ideally, a well-trained HMM can give sufficiently high likelihood only for sequences that correspond to normal behaviors. Sequences correspond to abnormal behaviors, on the other hand, should give a significantly lower likelihood values. On this property the anomaly detection method in this paper is based.

Given a predetermined threshold ε_1, with which we compare the probability of a sequence *X* in a testing trace, if the probability is below the threshold, the sequence *X* is flagged as a mismatch. We sum up the mismatches and define the *anomaly index* as the ratio between the numbers of the mismatches and all the sequences in the trace. And the classification rule is assigned as follows:

$$\text{anomaly index} = \frac{\text{numbers of the mismathes}}{\text{numbers of all the sequences in a trace}} \le \varepsilon_2 \qquad (9)$$

If (9) is met, then the trace (process) embedding the testing sequences is considered as a possible intrusion.

4. Experiments

4.1. Data Set

For the purpose of comparison, the experiments are based on several sets of CERT synthetic *sendmail* system call sequences, which were collected by Forrest et al. and can be downloaded at http://www.cs.unm.edu/~immsec/. In our experiments, the sequences are first converted into segments of fixed lengths. A fraction of the normal system call segments are then grouped as the training dataset. Another fraction of normal segments and 6 abnormal traces of system calls are used as the testing dataset. The description of the datasets in the experiments is shown in Table 1.

Table 1. Description of the data sets used in the experiments

Training data (Normal data)		105 traces of the latter data in *Sendmail.ini* 12 traces of the former data in *Sendmail.daemon.ini*
Testing data	Normal	42 traces of the former data in *Sendmail.ini* 5 traces of the latter data in *Sendmail.daemon.ini*
	Abnormal	4 syslog attacks and 2 unsuccessful attacks

In the training set of the normal data, there are 53 unique system calls. Therefore, we use 53 states in the experiments for modeling the program behaviors and detecting anomalies described in section 2 and section 3.

4.2. Experimental Results and Discussion

Good testing results are obtained by modeling program behaviors using the proposed method based on HMMs for anomaly detection. In the experiments, we use different window sizes as 3, 6, 7, 10, and 11 respectively for comparison of the experimental results.

The anomaly indexes obtained for each of the different 5 window sizes are shown in Table 2 together with the results obtained by Forest et al. [5] and Lee et al. [6] for comparison.

From Table 2, it is easily observed that:

(1) The anomaly indexes of the abnormal sequences are significantly higher than those of the normal testing data set (the last row). Therefore, the normal and abnormal can be easily distinguished using the proposed method based on HMMs.

(2) The anomaly indexes of the abnormal sequences are higher and the normal sequences are lower than those of obtained by Forrest et al. and Lee et al for all the window sizes. This means that the detection accuracy of the proposed method is better than the methods proposed by Forrest et al. and Lee et al.

(3) The anomaly indexes in the experiments are variable with different windows sizes. This means that the performance of the intrusion detection is related to the window size. Therefore, the study on different window sizes is meaningful.

The proposed method is effective for intrusion detection. Though training an HMM is computationally expensive in the procedure of modeling program behaviors, testing is efficient once the model was built for normal program behaviors. Therefore, the proposed method can be used for online detection in the real world.

Table 2: The anomaly indexes of the testing traces under different window sizes compared with the results of Forrest et al. and Lee et al. Note that the sequences listed in the table are all abnormal except the one in the last row and the values in the table are the percentage of the anomaly indexes.

System Call Sequences	Forrest	Lee	The proposed method based on HMMs				
	Window sizes	Window sizes	Window Sizes				
	11	7	11	10	7	6	3
syslog-remote-1	5.1	11.5	14.32	25.59	23.67	21.01	14.42
syslog-remote-2	1.7	8.4	11.73	21.70	20.43	18.28	13.41
syslog-local-1	4.0	6.1	6.31	16.72	14.97	11.85	7.46
syslog-local-2	5.3	8.0	6.39	19.23	17.22	13.96	8.46
sm565a	0.6	8.1	4.91	27.74	24.91	19.63	13.19
sm5x	2.7	8.2	2.75	28.08	24.04	19.45	10.88
Normal	0	0.6	0	0.35	0.55	0.31	0.28

5. Concluding Remarks

In this paper, a new efficient intrusion detection method based on HMMs was proposed. The method considers the transition information of system calls issued by programs. The normal program behaviors were modeled using HMMs and any significant deviation from the model is considered as possible intrusion. In stead of considering the unobservable state transition probability in other HMMs based methods in paper [14] and [16], this new method only takes into account the output observable probability. Therefore, the computation would be reduced in the procedure of anomaly detection. Another advantage of this method is it uses the percentage of the "mismatches" while the methods in paper [17] and [18] only calculate the probability of the output observable. Our method sounds more reasonable because if one system call does not appear in the training set, the probability of some sequences embedding the system call is zero. In this case, the trace may not be anomalous but method in paper [17] and [18] would classify it as intrusion. Therefore our method would reduce the false alarms and the performance of the intrusion detection would be better.

The proposed method is implemented and tested on the *sendmail* system call data from the University of New Mexico. Experiment results show that the method is more efficient in terms of detection accuracy compared to other methods.

Preliminary research shows that the number of the states is a sensitive parameter [14]. We hope to determine the physical meaning of the number by further experiments. In addition, considering the transition information of sequences is computationally expensive, another further research is in progress to mix the frequencies property with the transition information of system call sequences so that lower false alarms and missing alarms is achieved and the computation effort is reduced.

Acknowledgements

The research in this paper was supported in part by the National Outstanding Young Investigator Grant (6970025), National Natural Science Foundation (60243001) and 863 High Tech Development Plan (2001AA140213) of China.

References

[1] S. E. Smaha, "Haystack: An intrusion detection system", Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference, pp. 37-44, December 1988.

[2] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey, "A real-time intrusion detection expert system (IDES) - final technical report", Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[3] D. Anderson, T. Frivold, and A. Valdes, "Next-generation intrusion detection expert system (NIDES): A summary", Technical Report SRI-CSL-95-07, Computer Science Laboratory, SRI International, Menlo Park, California, May 1995.

[4] Y.H. Liao, V. R. Vemuri, "Use of K-nearest Neighbor Classifier for Intrusion Detection", Computer & Security, vol. 21, No. 5, pp. 439-448, 2002.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, " A sense of self for Unix processes", Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, Los Alamos, CA, pp. 120-128, 1996.

[6] W. Lee and S. Stolfo, "Data Mining Approaches for Intrusion Detection", Proceedings of the 7th USENIX Security Symposium, Usenix Association, pp. 79-94, January 1998.

[7] Asaka M., Onabuta T., Inoue T., Okazawa S. and Goto S., "A New Intrusion Detection Method Based on Discriminant Analysis", IEICE Transactions on Information and Systems Vol. E84D, No. 5, pp. 570-577, 2001.

[8] Wespi, A., Dacier, M. and Debar, H., "Intrusion Detection Using Variable-Length Audit Trail Patterns", Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000), No. 1907 in LNCS, 2000.

[9] Z. Cai, X. Guan, P. Shao, Q. Peng, G. Sun, "A Rough Set Theory Based Method for Anomaly Intrusion Detection in Computer Networks", Expert Systems, vol. 18, No. 5, pp. 251-259, Nov 2003.

[10] Wenjie Hu, Yihua Liao and V. Rao Vemuri. "Robust Support Vector Machines for Anomaly Detection in Computer Security". The 2003 International Conference on Machine Learning and Applications (ICMLA'03). Los Angeles, California, June 2003.

[11] N. Ye, X. Li, Q. Chen S. M. Emran and M. Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data", IEEE Trans. SMC-A, Vol. 31, No. 4, pp. 266-274, 2001.

[12] N. Ye, Y. Zhang and C. M. Borror, "Robustness of the Markov chain model for cyber attack detection", IEEE Transactions on Reliability, Vol. 53, No. 1, pp. 116-121, March 2004.

[13] W.-H. Ju and Y. Vardi, "A Hybrid High-order Markov Chain Model for Computer Intrusion Detection",

Journal of Computational and Graphical Statistics, Vol. 10, No. 2, pp. 277-295, 2001.

[14] C. Warrender, S. Forrest and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models", Proceedings of 1999 IEEE Symposium on Security and Privacy, pp. 133-145, 1999.

[15] Lane T. Machine Learning Techniques for the Computer Security Domain of Anomaly Detection, PhD thesis. Purdue University, August, 2000.

[16] Yanqiao, Xie Weixin, Yangbin, Songge, "An anomaly intrusion detection method based on HMM", Eletronics Letters, Vol. 38, No. 13, pp. 663-664, 2002.

[17] D.Y. Yeung, Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models", Pattern Recognition, Vol. 36, No. 1, pp. 229-243, January 2003.

[18] Sung-Bae Cho, Hyuk-Jang Park, "Efficient anomaly detection by modeling privilege flows using hidden Markov model", Computers & Security Vol.22, No.1, pp. 45-55, 2003.

[19] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification. China Machine Press, Beijing, 2nd edition, February 2004.

[20] Rabiner, L.R., 1989. "A tutorial on hidden Markov models and selected applications in speech recognition". Proceeding of the IEEE, Vol. 77, No. 2, 1989.

[21] Rabiner, L.R. and Juang, B.H., "An introduction to hidden Markov models", IEEE ASSP Magazine, 1986.