An Efficient Hidden Markov Model Training Scheme for Anomaly Intrusion Detection of Server Applications Based on System Calls

X. D. Hoang and J. Hu
{xhoang, jiankun}@cs.rmit.edu.au
School of Computer Science and Information Technology
RMIT University, Melbourne, Victoria 3000, Australia

Abstract - Recently hidden Markov model (HMM) has been proved to be a good tool to model normal behaviours of privileged processes for anomaly intrusion detection based on system calls. However, one major problem with this approach is that it demands excessive computing resources in the HMM training process, which makes it inefficient for practical intrusion detection systems. In this paper a simple and efficient HMM training scheme is proposed by the innovative integration of multiple-observations training and incremental HMM training. The proposed scheme first divides the long observation sequence into multiple subsets of sequences. Next each subset of data is used to infer one sub-model, and then this sub-model is incrementally merged into the final HMM model. Our experimental results show that our HMM training scheme can reduce the training time by about 60% compared to that of the conventional batch training. The results also show that our HMM-based detection model is able to detect all denial-of-service attacks embedded in testing traces.

1. INTRODUCTION

Anomaly intrusion detection is one of two major intrusion detection techniques namely *misuse detection* and *anomaly detection* [4]. Generally, an anomaly intrusion detection scheme constructs models of subject behaviour, and considers any significant deviation from normal behaviours as part of an attack. Anomaly intrusion detection has the potential to detect unknown attacks because no prior knowledge about specific intrusions is required [4].

Intrusion detection can be considered as a data analysis process [4]. This is particularly true in anomaly intrusion detection where intrusion detection systems collect and process large volume of raw data from various sources such as network traffic and activity information of users and programs. Data processing techniques such as data mining, machine learning and statistics have been commonly used in intrusion detection systems.

In 1996, Forrest and her colleague [1] are the first researchers who proposed the idea of using programs' system calls to operating system kernel to detect the anomalous activities of programs. It is empirically proved that short sequences of system calls produced by the execution of some Unix privileged processes such as *sendmail* and *named* are a good discriminator between the normal and abnormal operating characteristics. They constructed a simple detection scheme that consists of two phases: (1) building a list of unique short sequences of a program's system calls in normal operation as *normal database* and (2) testing input short sequences against the *normal database* to find the anomalies. If a mismatch is found, the short sequence is considered to be anomalous. The idea has been extensively extended in their later work [2] with wide range of experiments on various Unix programs under synthetic and real working environment. The experimental results have also confirmed that short sequences of system calls are stable and consistent during program's normal activities. This method is simple and efficient because it requires only one pass through the training data to build up the *normal database*. However, the method may generate a high number of false alarms because it is impossible to build a complete database covering all scenarios.

Warrender *et al* [3] continued the work along this line and investigated various data processing methods through extensive experiments. They used the enumerating method [1], frequency based method, data mining, and hidden Markov model to build up the detection models from the system call traces. The experimental results have shown that hidden Markov model can generate the most accurate results on average. The major drawback of HMM approach is that HMM training demands excessive computational resources.

Several other HMM based anomaly intrusion detection approaches can be found in [5] and [6]. In [5], Qiao *et al* first built up a HMM from the input system call trace, and then used this HMM to transform the input sequences of system calls into the sequences of hidden states. Finally they applied the enumerating method [1] on the hidden state sequences to build the *normal database* to detect anomalous sequences. This transformation, however, cannot solve the problem of incomplete database since the mapping between the input system calls and output states is one to one. A complete database of hidden states cannot be built from an incomplete input data set.

In [6], Hoang *et al* have proposed a "*Multi-layer model*" to address the issue of incompleteness of *normal database*. The proposed model integrates the enumerating method [1],

0-7803-8783-X/04/$20.00 © 2004 IEEE

frequency based method, and HMM into single hybrid detection system so that it can alleviate the problem of incompleteness of *normal database* and also increase the detection accuracy. This method comprises of two phases: the training phase and testing phase. In the training phase, a *normal database* and a HMM are constructed from the system call traces of a program's normal operation. In the testing phrase, the input sequences are first compared against the normal database; if a mismatch is found the suspected sequence is then passed to the HMM for further verification. By using such two verification layers on suspicious sequences, this method is reportedly able to reduce the false alarm rate and the necessary amount of training data effectively. However it demands very high computational cost for HMM training, especially with large-size HMM and long system call traces which makes it infeasible for online intrusion detection.

The HMM training using multiple-observation sequences algorithm (HMMMOSA) [9] has been successfully applied in speech recognition. The idea of using multiple-observation sequences is useful for reducing demands on training time and memory resources. In order to solve the aforementioned HMM training problem for the anomaly intrusion detection, we introduce and modify the HMMMOSA scheme in this paper. Our new HMM training scheme first divides the long training sequence into a number of subsets of sequences. Next, each subset of data is used to train one sub-model and then the sub-model is incrementally merged into the final model. This step is different from the HMMMOSA scheme where the training of all sub-models is completed first and then they are merged into final model. With this modification, the HMM training scheme has become incremental. Therefore it can further reduce the resource demands of the HMMMOSA scheme, which makes it suitable for online intrusion detection. Our experiments on the *sendmail* and *inetd* data [12] show that our HMM training scheme can reduce the training time and memory requirements significantly.

The rest of the paper is organized as follows: Part 2 first provides a brief introduction to the hidden Markov model. Then we describe HMM modelling of normal behaviour of programs. Part 3 illustrates the proposed HMM training scheme for anomaly intrusion detection. Part 4 presents experimental results and analysis and part 5 is our conclusion.

2. PRELIMINARY

2.1 Hidden Markov Model

Hidden Markov Model (HMM) is a double embedded stochastic process with two hierarchy levels. The upper level is a Markov process, in which the states are not observable. Observation is a probabilistic function of the upper level Markov states. Different Markov states will have different observation functions.

HMMs are very powerful modelling tools although they are computationally expensive [7]. HMMs have been widely used in DNA sequence modelling, speech recognition and pattern recognition. For convenience, we use the same HMM notations as that in [8]. A HMM has the following elements:

- N: number of states in the model
- M: number of distinct observation symbols per states
- T: length of observation sequence, i.e. the number of symbols observed
- i_t : state in which we are in at time t
- $V = \{v_1, v_2, \ldots, v_M\}$: the discrete set of possible observation symbols
- $\pi = \{\pi_i\}$, $\pi_i = P(i_1 = i)$: the probability of being in state i at $t = 1$
- $A = \{a_{ij}\}$, $a_{ij} = P(i_{t+1} = j,\ i_t = i)$: the probability of being in state j at time t+1 given that we were in state i at time t.
- $B = \{b_j(k)\}$, $b_j(k) = P(v_k \text{ at } t \mid i_t = j)$: the probability of being observing symbol v_k given that we are state j.
- $O = \{O_1, O_2, \ldots, O_t, \ldots, O_T\}$: observation sequence; O_t denotes observation symbol observed at time t.

And $\lambda = \{A, B, \pi\}$ will be used as compact notation to denote an HMM.

2.2 Building HMM to Model Program's Normal Behaviour

We consider the system call trace of a program produced in its execution as the observation sequence and each system call in the trace as one observation symbol in terms of HMM notations. The first step in building HMM is the selection of the HMM size. The number of unique system calls used in the program's trace is defined as the number of distinct observation symbols M. The selection of number of states is based on the number of unique system calls used by program in the trace [3]. In our specific HMM models we choose the number of states N equal to the number of distinct observation symbols M. Specifically, we select N=M=23 for *sendmail* program (*stide* data set [12]) and N=M=35 for *inetd* program. We choose our HMM as ergodic model [7] in which states are fully connected, and transitions are allowed from one state to any other states.

Our HMM based intrusion detection scheme consists of two phases and is presented in Fig 1. In the training phase, the system call data are first transformed to HMM observation sequence. Next the HMM is inferred from the observation sequence. In the testing phase, the system call data are first transformed to HMM short sequences, and then the HMM is used to compute the probability of each test short sequence in order to determine if it is normal or anomalous. A pre-defined probability threshold is used in the testing process.

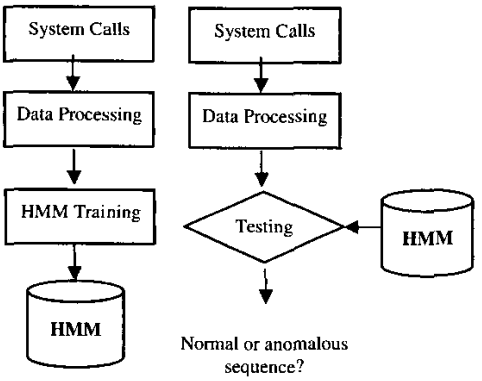


Figure 1. HMM training and testing for anomaly intrusion detection based on system calls

2.3 HMM Batch Training Algorithm

We use the well-known Baum-Welch algorithm [7] to train our HMM sub-models. The HMM training using Baum-Welch algorithm is considered as batch training because it allows only one observation sequence. Given the observation sequence
$O = \{O_1, O_2, ..., O_T\}$, the algorithm estimates HMM model's parameters $\lambda = \{A, B, \pi\}$, to maximize $P(O \mid \lambda)$. The Baum-Welch algorithm can be described in brief as follows [7]:

1. Let initial model be λ_0
2. Compute new model λ based on λ_0 and observation sequence O
3. If $\log(P(O\mid \lambda)) - \log(P(O\mid \lambda_0)) < DELTA$ go to step 5
4. Else set $\lambda_0 \leftarrow \lambda$, and go to step 2
5. Stop

where *DELTA* is a pre-defined threshold value of the natural logarithm of the probability.

3. PROPOSED HMM INCREMENTAL TRAINING SCHEME

3.1 Data Sets

We use the "*inetd*" and "*stide*" data sets given in [12] for our experiments. The procedures of generating and collecting these traces are described in [1], [2] and [3]. Each trace is a list of system calls produced by processes of a program during its execution. We select the live *sendmail* and *inetd* data collected at the University of New Mexico (UNM) to use in our experiments. The data include:

- *Normal traces*: traces collected during the program's normal activities including sendmail daemon traces and *sendmail*'s sub-processes. The sendmail data consist of 13,276 traces with the total of 15,631,952 system calls. The inetd data include a single trace with 541 system calls. These traces are used to build the hidden Markov models.
- *Abnormal traces*: traces come from a program's abnormal runs that generated by intrusion tools for some known intrusions. Specific anomaly traces used include 105 traces of a denial-of-service attack on *sendmail* and a single trace of a denial-of-service attack on *inetd*.

3.2 HMM Incremental Training from Multiple Observation Sequences

Gotoh *et all* [10] proposed efficient HMM training schemes using incremental *ML* and *MAP* estimation algorithms for speech recognition. The HMM incremental training has the advantage of faster convergence than that of traditional batch training. However, their algorithms only hold when the subsets of training data are independent. In our training data set, the system calls are in interactive relations and condition of independent subsets does not hold. Davis and other authors [9] proposed a simple method to learn HMM from multiple observation sequences (HMMMOSA). In their approach, first the set of sub-sequences are used to learn a set of sub-models independently. Next, when the learning of all sub-models is complete, the sub-models are merged together based on weights to produce the final HMM. No constraint of independent subsets is required in this method.

In our approach, we modify the HMMMOSA scheme [9] to make it become incremental. Our new HMM training scheme first divides the long training sequence into a number of subsets of sequences. Next, each subset of data is used to train one sub-model and then the sub-model is incrementally merged into the final model. The training scheme can be described in the following steps:

1. Divide single observation sequence O into K sub-sequences $\{O_{(1)}, O_{(2)}, ..., O_{(K)}\}$.
2. Initialise the HMM final model $\lambda \leftarrow \varnothing$ (empty model).
3. Take a sub-sequence $O_{(k)}$ to train sub-model $\lambda_{(k)}$ using HMM batch training algorithm described in section 2, sub-section 2.3.
4. Incrementally merge $\lambda_{(k)}$ into final model λ.
5. Repeat steps 3 and 4 for all sub-sequences.

A graphical presentation of our scheme is shown in Fig 2 in which $\lambda_{(k)}^{(all)}$ is the final HMM model at k observation sequences.

The HMM parameters in the incremental merging step of the sub-model $\lambda_{(k)}$ and the final HMM λ are calculated as follows:

$$\bar{a}_{ij} = w_k * a_{ij}^{(k)} + w * a_{ij}$$

$$\bar{b}_{ij} = w_k * b_{ij}^{(k)} + w * b_{ij}$$

$$\bar{\pi}_i = w_k * \pi_i^{(k)} + w * \pi_i$$

where $w_k = 1 / P(O_{(k)}|\lambda_{(k)})$, $P(O_{(k)}|\lambda_{(k)})$ is the probability to generate sub-sequence O_k from model λ_k, and $w = 1 / P(O_{(1)}, O_{(2)}, O_{(k-1)}|\lambda)$, $P(O_{(1)}, O_{(2)}, O_{(k-1)}|\lambda)$ is the probability to generate sub-sequences $\{O_{(1)}, O_{(2)}, O_{(k-1)}\}$ from model λ.

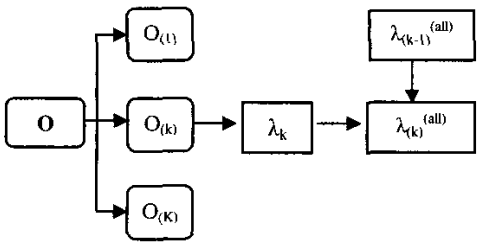


Figure 2. Incremental HMM Training from multiple sub-sequences

4. EXPERIMENTAL RESULTS AND DISCUSSION

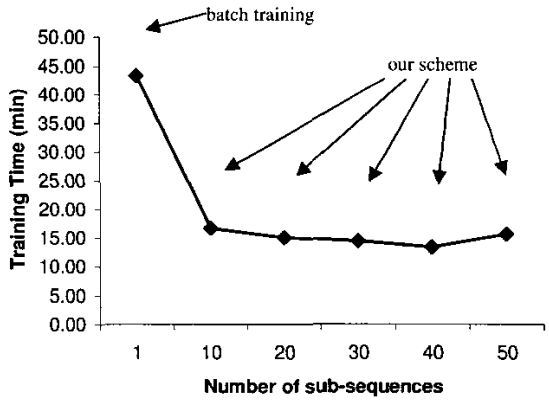
4.1 Experiment Design and Results

We have conducted several experiments on the *sendmail* and *inetd* data sets described in section 3, sub-section 3.1. The HMM training *DELTA* threshold is 0.1. All HMM sub-models have the same size as the size of the final HMM. First, we measure the efficiency of our HMM incremental training scheme against the traditional HMM batch training in terms of training time and memory requirements. Next, we test our HMM based detection models on two *denial-of-service* (DOS) attacks on *sendmail* and *inetd*. In these experiments we use HMM models to compute the logarithm of the probabilities (*log(P)*) of test sequences, and we consider these values as anomaly signals. The length of short sequences used is 20 system calls.

Fig. 3 shows the relation between the HMM training time and the number of sub-sequences in our scheme. The *sendmail* data set is used in this experiment with the total length of 1,000,000 system calls. The traditional HMM batch training is shown in the figure with sub-sequence number of 1.

Fig. 4 shows log(P) of length-20 sequences in trace produced by a denial-of-service attack on the *inetd* program. The DOS intrusion embedded in *inetd* trace is clearly detected by the anomalous signal of the sequences between 13 and 30.

Fig. 5 shows log(P) of length-20 sequences in trace produced by a denial-of-service attack on the *sendmail* program. The *sendmail* DOS intrusion is also clearly detected by the anomalous signal of the sequences between the start of the trace and sequence 2500, and between sequence 9000 and the end of the trace.



multiple sequences. The total length of all sub-sequences is 1,000,000 system calls. When the number of sub-sequences is 1 it is the traditional HMM batch training.

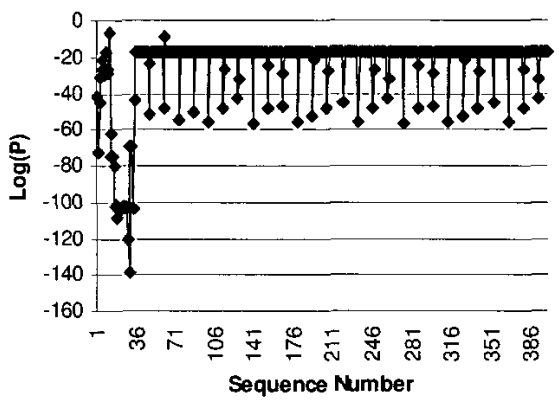


Figure 4. Log(P) of sequences of trace produced by a

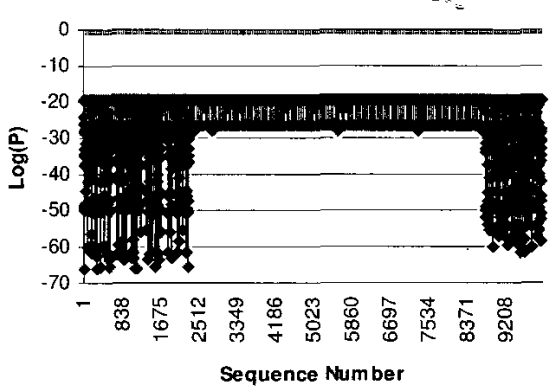


Figure 5. Log(P) of sequences of trace produced by a denial-of-service attack on *sendmail*.

4.2 Discussion

Table I shows the HMM training time of batch training mode (training with single sequence) and our scheme. It is clear that our training scheme is much more efficient than the batch mode in all number of sub-sequences tested. On average, our scheme can reduce about 60% training time compare to that of batch training. The training time of HMMs in our scheme depends on the number of sub-sequences. Generally, the training time decreases when the number of sub-sequences increases. However, the length of sub-sequences should not be too small because it may lead to the over-fitting of HMM parameters. In our experiment, we define the minimum length of the sub-sequence based on the size of HMM model.

TABLE I. RELATION BETWEEN HMM INCREMENTAL TRAINING TIME AND THE NUMBER OF SUB- SEQUENCES

Number of sub-sequences	Training time (min)	In comparison to batch mode (%)
1 (batch mode)	43.33	100.00%
10	16.67	38.46%
20	14.98	34.58%
30	14.43	33.31%
40	13.32	30.73%
50	15.57	35.92%

As the memory requirements for HMM training are in the complexity of $O(2NT)$ [3], where N is number of hidden states and T is the length of the training sequence, our HMM training scheme uses shorter training sequence and hence, demands less system memory than that of batch training.

In addition, a significant reduction of the training time and the incremental estimation of HMM parameters make it feasible for our scheme to be used in online HMM training for real-time intrusion detection in which the working HMM model is updated dynamically from on-going data.

Fig. 4 and 5 clearly show that our HMM based detection model is able to detect all denial-of-service attacks on server applications embedded in testing data such as *sendmail* and *inetd*. The *log(P)* values of anomalous sequences are significantly smaller than that of normal sequences.

5. CONCLUSION

In this paper, a HMM incremental training scheme from multiple observation sequences for anomaly intrusion detection has been presented. Our experimental results show that our HMM training scheme can save up to 60% training time compared to batch training. The scheme is very promising for use in the online HMM training for real-time intrusion detection.

Open Issues

The initial values of HMM parameters in the training are sensitive factors to the convergence rate. In this paper we simply adopted the popular approach of using random values for HMM initial model. Searching for the optimal HMM initial values will be an interesting research question.

6. REFERENCES

[1] Forrest S., Hofmeyr S. A., Somayaji A., and Longstaff T. A., "A sense of self for Unix processes". *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996.

[2] Forrest S., Hofmeyr S. A., and Somayaji A., "Intrusion detection using sequences of system calls". *Journal of Computer Security* Vol. 6, pp. 151-180, 1998.

[3] Warrender C., Forrest S., and Perlmutter B., "Detecting intrusions using system calls: Alternative data models". *Proceedings of the 1999 IEEE Computer Society Sympo-sium on Research in Security and Privacy (Berkeley, CA, May).* IEEE Computer Society Press, Los Alamitos, CA, pages 133-145, 1999.

[4] Lee W., and Stolfo S. J., "A Framework for Constructing Features and Models for Intrusion Detection Systems". *ACM Transactions on Information and System Security*, Vol. 3, No. 4, pages 227-261, November 2000.

[5] Qiao Y., Xin X. W., Bin Y., and Ge S., "Anomaly intrusion detection method based on HMM". *IEEE Electronic Letters* Online No: 20020467, 2002.

[6] Hoang X. D., Hu J. and Bertok P., "A multi-layer model for anomaly intrusion detection using program sequences of system calls". *International Conference on Network - ICON2003*, pages 531-536, September 2003, Sydney, Australia.

[7] Rabiner L. R., "A tutorial on hidden Markov model and selected applications in speech recognition", in *Proceedings of IEEE*, Vol. 77, No. 2, February 1989.

[8] Dugad R., and U. B. Desai, "A tutorial on Hidden Markov Models," *Technical report No.: SPANN-96.1*, May 1996.

[9] Davis R. I. A. and Lovell B. C., "Improved Estimation of Hidden Markov Model Parameters from Multiple Observation Sequences", in *International Conference on Pattern Recognition*, pages 168-171, Quebec City, Canada, August 2002.

[10] Gotoh Y., Hochberg M. M., and Silverman H. F., "Efficient Training Algorithms for HMMs Using Incremental Estimation", *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*, VOL. 6, NO. 6, NOVEMBER 1998, pages 539-548.

[11] The General Hidden Markov Model library (GHMM) web page: http://www.ghmm.org/.

[12] University of New Mexico's Computer Immune Systems Project web page: http://www.cs.unm.edu/~immsec/systemcalls.htm.