

A Multi-layer Model for Anomaly Intrusion Detection Using Program Sequences of System Calls

Xuan Dau Hoang, Jiankun Hu, Peter Bertok

xhoang@cs.rmit.edu.au, jiankun@cs.rmit.edu.au, pbertok@cs.rmit.edu.au
School of Computer Science and Information Technology, RMIT University, Melbourne, Australia

Abstract- In this paper we present a new method to process sequences of system calls for anomaly intrusion detection. The key idea is to build a multi-layer model of program behaviours based on both hidden Markov models and enumerating methods for anomaly intrusion detection, which differs from the conventional single layer approach. Our experiments on Unix *sendmail* program have shown that the model is better in detecting anomalous behaviour of programs in terms of accuracy and response time. As we use the temporal characteristics in the model, it is suitable for online host-based intrusion detection systems in LAN environment.

Index Terms - Intrusion detection, anomaly detection, hidden Markov model, machine learning, system call sequence.

I. INTRODUCTION

INTRUSION detection techniques can be broadly classified into two categories: *misuse detection and anomaly detection* [8]. Misuse detection looks for signatures of known attacks, and any matched activity is considered an attack. Misuse detection can detect known attacks efficiently, but it performs poorly with unknown attacks. Anomaly detection constructs models of subject behaviour, and any significant deviation from normal behaviours is considered part of an attack. Anomaly detection has the potential to detect unknown attacks since no advance knowledge about specific intrusions is required [8]. In practice, both techniques are used in an intrusion detection system as they complement each other.

A number of techniques such as machine learning, statistics and genetic programming have been used in intrusion detection. Most of previous work on anomaly intrusion detection focused on the user behaviours. Lee and Stolfo [8] proposed a novel framework, called MADAM ID, for Mining Audit Data for Automated Models for Intrusion Detection. The framework applied data mining techniques to process the user's command traces, and then build the user profile. This approach is mainly for misuse detection. Though it can be used to detect some variations of known intrusions, it generally does not perform well against unknown attacks.

On anomaly detection at program level, Forrest, Hofmeyr, Somayaji, and Longstaff [1] introduced the concept of "Self" of Unix privileged processes such as *sendmail* and *named*

programs. They have found that short sequences of system call traces produced by the execution of these programs are a good discriminator between the normal and abnormal operating characteristics of programs. They named the short sequence of system calls as *n-gram*.

Forrest, Hofmeyr and Somayaji [2] in an extension of their earlier work [1], conducted a large number of experiments on various Unix programs in synthetic and real working environments. The experimental results have also confirmed that short sequences of system call traces are stable and consistent during program's normal activities.

There are various ways in which system call data could be used to characterize the normal behaviours of programs. One of the simplest methods used in [1], and [2] is to construct a list of unique short sequences of system call traces, which is called normal database. To detect anomalies, the short sequences from the test traces are compared to those in the normal database. If a mismatch is found, the sequence is considered as anomalous. The result is reliable if a match is found. However, this method may generate a high number of false alarms because it is extremely difficult to build a complete database for all scenarios.

Other approaches to model normal program behaviours via system call data are to construct finite state machines to recognize the patterns of the normal program activities. Among these, Micheal and Ghosh [9] presented two state-based models, in which the state machines are created automatically from the trace data. Their methods have an advantage of demanding less training data. A fast finite automaton (FA) was built in [10] by exploiting information on the execution path of the program. The major problem of this approach is the requirement of the accurate knowledge of the program's source code or the execution path, which is not always practical. Qiao, Xin, Bin and Ge [11], used Hidden Markov Model (HMM) – a powerful finite state machine to transform the input system call sequences into sequences of hidden states. Then they used enumerating method [1] on the state sequences to build the normal database for anomalous sequence detection. 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, which means a complete database of states cannot be built from an incomplete input data set.

In order to solve the problem of low reliability or incomplete database suffered by the conventional methods, we propose a new scheme to process sequences of system calls for anomaly intrusion detection in this paper. In our approach, we extend enumerating approach [1], by adding the frequency information of each unique sequence to the normal database. Furthermore, we employ hidden Markov model to develop a multi-layer intrusion detection model in order to solve the incompleteness problem of database. Using the normal database, we can classify mismatch and/or suspected rare sequences, and then pass them to the HMM for further analysis, in which we compute the probability required to produce that sequence. Based on this sequence probability we finally determine whether it is really anomalous. We have obtained data sets for *sendmail* program from [12] for our experiments.

The rest of the paper is organized as follows: part II describes the architecture of the proposed multi-layer anomaly intrusion detection model. Part III explains our profile building process including the data sets, building of the normal database and HMM model. Part IV shows the experimental results. Part V is the discussion on the experimental results, and part VI is our conclusion.

II. MULTI-LAYER MODEL FOR ANOMALY INTRUSION DETECTION

A. Preliminaries

Traditional Markov models have been used to successfully model a wide range of real world processes. However, for other processes, the strict assumption of Markov model that the next state is dependent only upon on the current state will not hold. Thus it is necessary to find more general models in order to deal with these processes while at the same time withholding the good properties of the Markov model. These needs motivate people to develop the Hidden Markov Model. Hidden Markov Model 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.

The two-hierarchy-level structure is the major advantage of HMMs. It can be used to model more complicated stochastic processes than the traditional Markov model. In practice, hidden Markov models have been widely used to model real world problems such as pattern recognition, speech recognition and modelling of biological sequences. For details of HMM, readers are referred to [5]-[7] and the references therein.

B. Multi-layer model

Our new scheme is an integrated version of the

enumerating method and HMM method as shown in Fig. 1.

The test procedures of a sequence are divided into two steps:

Step1: The sequence of system calls is compared to those in normal database to find a mismatch or a rare sequence indicated by low occurring frequency.

Step 2: If the sequence is rare and/or a mismatch, it is then input to the HMM model to compute the corresponding probability. If the probability required to produce the sequence is smaller than a pre-defined probability threshold, it is considered as an anomalous sequence. By using HMM as an additional analysis of the mismatch sequences, it can help to reduce the false alarms.

To measure the anomaly signal, we use a temporally local region of length of r consecutive sequences. Like sliding window of short sequences, the local region is moving across the test trace. However, we use separate regions, i.e. no region overlaps another. This is because overlapped regions tend to propagate one single anomaly signal to many neighbouring regions, and then this in turn, produces more

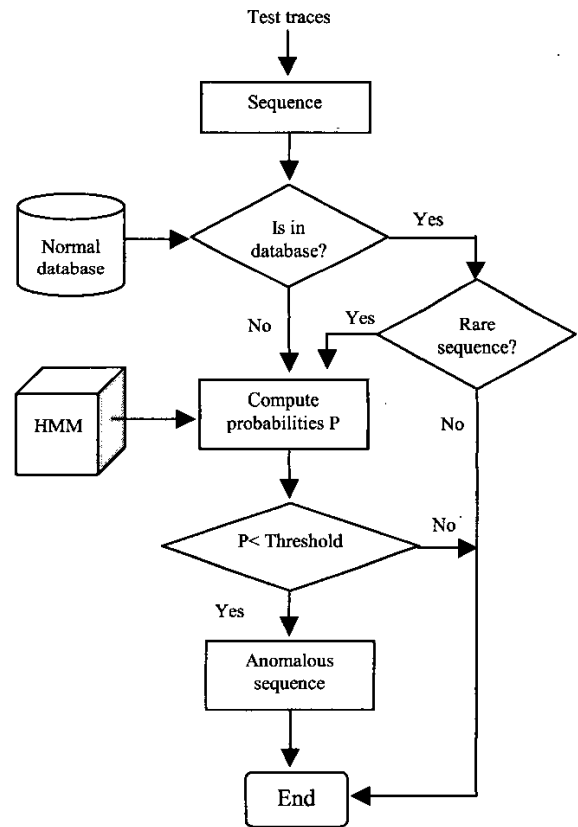


Fig. 1. Multi-layer intrusion detection scheme

false alarms. We count the number of anomalous sequences m in the region and considered the ratio of (m/r) as the value of anomaly signal. Again, a threshold of the region is needed to determine whether the region is normal or anomalous.

III. BUILDING NORMAL DATABASE AND HMM MODEL FOR EXPERIMENTS

A. Data sets

We have obtained the data sets from [12] for our experiments. The data sets consist of several system call traces of some common Unix processes such as sendmail, FTP, named and lpr programs. The procedures of generating these traces are described in [1] and [2]. Each trace is a list of system calls produced by processes of a program during its execution. Table 1 shows the sample of a trace file, in which each line contains the process identification number (PID) and the system call number as a pair. The mapping between the system call numbers and the actual system call names is provided in a separate file. For example, in the “calls.txt.sun” system call mapping file [12], number 20 is corresponding to the system call “getpid”, and number 132 represents call “mkdir”. We selected sendmail program traces to test our system, which includes:

- (1) Normal traces: traces collected during the program’s normal activities including sendmail daemon traces and sendmail’s sub-processes. These traces are used to build the normal databases.
- (2) Abnormal traces: traces come from program’s abnormal runs that generated by intrusion tools for some known intrusions. Specific anomaly traces used include 1 trace of *sm5x*, 1 trace of *sm565a*, 2 traces of *syslog-local*, and 2 traces of *syslog-remote* intrusions.

TABLE 1
A PAIR OF PID AND SYSTEM CALL NUMBER IN TRACES [12]

PID	1393	1393	1423
System calls	112	19	105

B. Building the normal database

In our experiment, we built the program profiles that consists of two parts: (1)- List of all unique short sequences of system call traces with frequencies, and (2)- HMM model of the program for two-layer intrusion detection.

At the first step, we built the profile of unique short sequences by enumerating all unique, contiguous sequences of a predetermined, fixed length k that occur in the training data. We ran experiments with sequence lengths of 5, 8, 11, 15, and 20. For example, with $k=5$, we slide a window of length 5 across each trace, one system call a time, adding each unique sequence to the normal database. We also count

the number of occurrences of each unique sequence in the training data set. To test the effect of the length of traces to the model, we used sub-set of 30%, 60% and 100% of the complete training data set. Building such database is efficient since it requires only a single pass through the data.

At testing time, sequences from the test trace are compared to those in the normal database. Any sequence not found in database is called a *mismatch*. Any individual mismatch could indicate anomalous behaviour, or it could be a sequence was not included in the training data. The rare sequences that have very low frequency are also taken into account when we test the trace. Mismatch and rare sequences are passed to HMM for further analysis.

C. Using HMMs to model program behaviours

To build an HMM model, the first step is to decide the number of states. A good choice of number of states for a program was roughly the number of unique system calls used by program [3]. For our specific experiments, we have 53 unique system calls in *sendmail* traces, hence we use an alphabet of 53-state HMM for *sendmail*. The HMM states are fully connected, and transitions are allowed from one state to any other state. For each state, we need to store the probabilities associated with transitions to each other state, and the probabilities associated with producing each system call. Transition and symbol probabilities are initialised randomly. The Baum-Welch algorithm [5] is used to train the model. During the training, the probabilities were iteratively adjusted to increase the likelihood that the automaton would produce the traces in the training set. Several passes through the data were required. To avoid over-fitting the training data, the likelihood of the model producing a second set of normal traces (not used in training) was periodically measured. When this second likelihood stopped improving, training was terminated [3].

Training HMMs is expensive in terms of computational power and storage. However, testing is more efficient. We use Forward algorithm [5] to calculate the probability that requires to produce the sequence and then compare this probability with the probability threshold to determine whether the sequence is normal or anomalous.

IV. EXPERIMENTAL RESULTS

Following the new scheme of intrusion detection described in section II, we have conducted several experiments on the *sendmail* program data sets obtained from [12]. Table 2 shows the number of unique sequences in normal databases with sequence length from 5 to 20 on 30%, 60% and 100% of full training data set. We use the region length r of 20 sequences in all experiments. Rare sequence threshold is 0.0001% and HMM probability threshold is 0.000001.

TABLE 2
NUMBER OF UNIQUE SEQUENCES OF SYSTEM CALLS IN NORMAL DATABASE

Trace size (% of full training data set)	k=5	k=8	k=11	k=15	k=20
30%	535	595	796	941	1112
60%	601	681	955	1151	1378
100%	748	847	1210	1462	1758

We use a normal trace that is not included into training set to test for false positive alarms at which the system reports normal activities as anomalous. Results in Table 3 shows that the false positive depends on the completeness of database, but not on the sequence length.

TABLE 3
DEPENDENCY OF FALSE POSITIVE RATES ON SEQUENCE LENGTH AND DATABASE SIZE

Trace size (% of full training data set)	k=5	k=11	k=15
30%	0.0022	0.0026	0.0026
60%	0.0011	0.0015	0.0015
100%	0.0011	0.0015	0.0015

Fig. 2 shows the strength of the anomaly signal on intrusion *sm5x* with full training data set and sequence length *k* of 5 and 20. It is clear that when the sequence length increases, the anomaly signal is locally stronger.

We have also conducted experiments on some known intrusion traces such as *sm5x*, *sm565a*, and *syslog* with the sequence length of *k=11*. Fig. 3, 4, 5, 6, 7 and 8 clearly show that these intrusions can be detected with the region threshold of 20%.

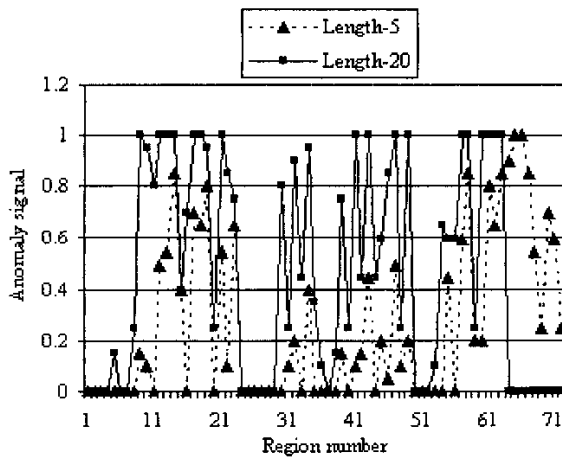


Fig. 2. Anomaly signal on *k=5* and *k=20* of *sm5x* intrusion

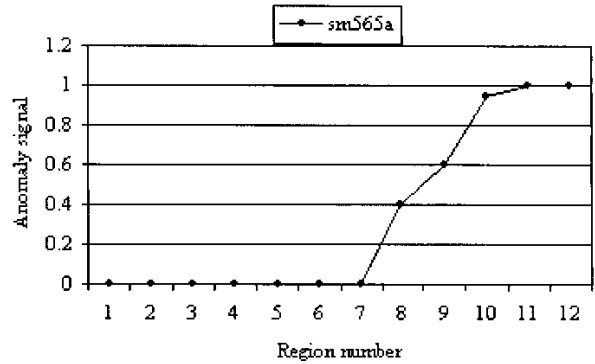


Fig. 3. Anomaly signal on *k=11* of *sm565a* intrusion

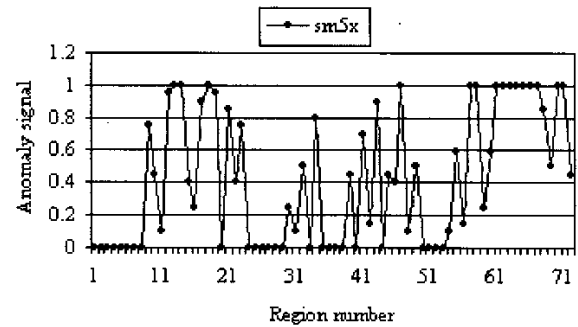


Fig. 4. Anomaly signal on *k=11* of *sm5x* intrusion

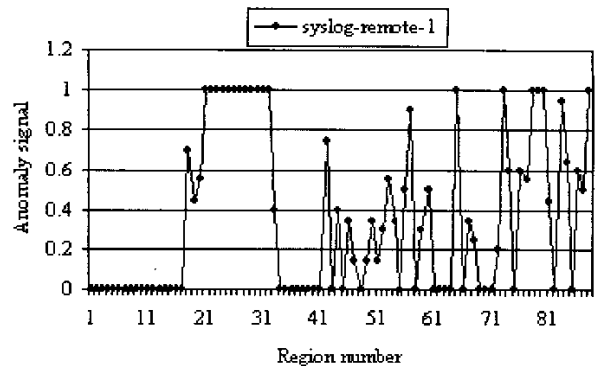


Fig. 5. Anomaly signal on *k=11* of *syslog-remote 1* intrusion

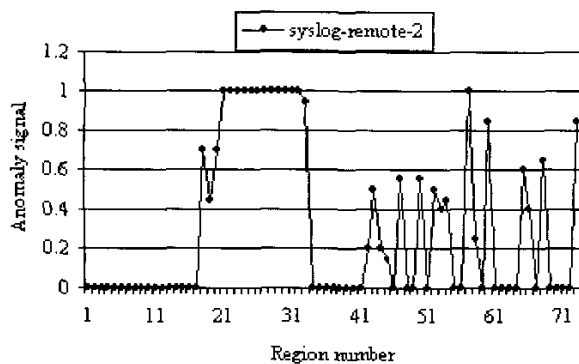


Fig. 6. Anomaly signal on $k=11$ of *syslog-remote 2* intrusion

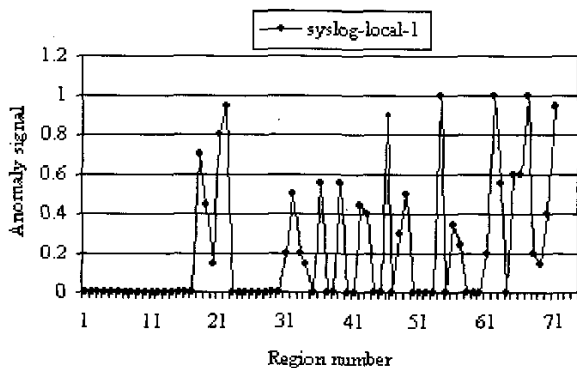


Fig. 7. Anomaly signal on $k=11$ of *syslog-local 1* intrusion

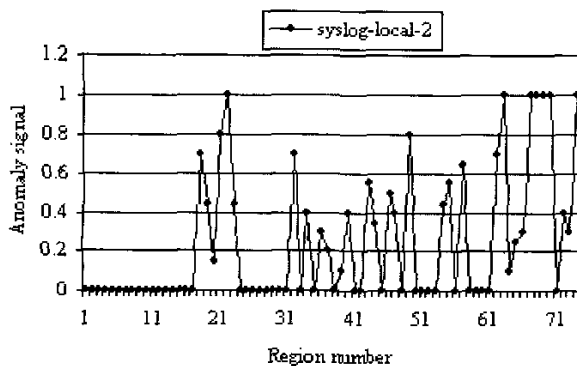


Fig. 8. Anomaly signal on $k=11$ of *syslog-local 2* intrusion

V. DISCUSSION

A. Anomaly signal strength and real time detection

Forrest, Hofmeyr, Somayaji, and Longstaff [1] measured anomaly signal by counting the number of mismatches and calculating the percentage of the mismatches out of the length of the test trace. Table 4 shows the anomaly signal of [1] versus our scheme's anomaly signal for the same intrusions.

TABLE 4
ANOMALY SIGNALS OF [1] AND OUR SCHEME

Anomalous Traces	Anomaly signal of [1]		Anomaly signal on average of our scheme (%)
	Number of mismatch	Percent (%)	
sm5x	453	2.7	66.4
sm565a	89	0.6	79.0
syslog-remote-1	1047	5.1	67.0
syslog-remote-2	286	1.7	71.6
syslog-local-1	688	4.0	51.7
syslog-local-2	883	5.3	56.1

The measurement approach in [1] is problematic because the anomaly signal is heavily depends on the test trace length, especially when the program running continuously produces trace with infinite length. Moreover, [1] needs a very low threshold in order to detect intrusion such as *sm565a* (0.6%). However, a very low threshold will result in large number of false positive alarms. In addition, it needs a full length of trace data to calculate the mismatch percentage, which makes it infeasible for online intrusion detection. However, it could be used for off-line analysis of intrusion when the program's execution is complete. On Fig. 3, the intrusion *sm565a* is clearly detected with our method from the region number 8 to 11 with the signal from 20% to 100% although it occurred in a very short period of time. The result also shows that our new scheme based on the temporally local region is able to detect the intrusions and anomalous activities as soon as they happen. This makes it well suited for online intrusion detection systems. For other intrusions, our method produces much stronger signals than those in [1] as shown in Table 4 and Fig. 4, 5, 6, 7 and 8.

B. Reducing False alarm rate

In order to reduce false alarm rate, we integrate the frequency of short sequences into the normal database. We take all the rare sequences and mismatches into HMM for further analysis. The output of HMM will determine whether the sequence is normal or abnormal. This can solve the *problem of mismatch due to incomplete normal database*.

C. Open issues

The parameters of the model such as the sequence length, region length, region anomaly signal threshold, and sequence probability are determined by experiments. They are sensitive parameters to the model. How to determine them automatically is a problem for future work.

As mentioned earlier, training an HMM is expensive. The calculation load for each trace in each pass through the training data takes the complexity of $O(TS^2)$, where T is the length of the trace in number of system calls, and S is the number of states (and symbols). Also, storage requirements are high. The "trellis" of intermediate values that must be kept while performing the calculations for a particular trace requires $T(2S+1)$ floating point values [3]. This is one of the drawbacks of the model and we need to do more research on the optimisation of the HMM training process.

VI. CONCLUSION

In this paper, a multi-layer intrusion detection model is presented. We integrate HMM model with enumerating method to help reduce the false positive alarms. The training of HMM model is expensive but the testing is more efficient. The integrated system is able to produce higher level of anomaly signal as soon as the intrusion happens. By processing the test trace in temporally local regions, the method is suitable for online detection. Although the proposed model is a host-based intrusion detection scheme, it has the potential for use in networked environments.

REFERENCES

- [1] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996.
- [2] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security* Vol. 6, pp. 151-180, 1998.
- [3] C. Warrander, S. Forrest, and B. Perlmutter, "Detecting intrusions using system calls: Alternative data models," in *the Proceedings of the 1999 IEEE Computer Society Symposium on Research in Security and Privacy (Berkeley, CA, May)*. IEEE Computer Society Press, Los Alamitos, CA, pages 133-145, 1999.
- [4] D. E. Denning, "An intrusion detection model," in *IEEE Transactions on Software Engineering*, Los Alamos, CA, IEEE Press, 1987.
- [5] L. R. Rabiner, and B. H. Juang, "An introduction to Hidden Markov Models", in *IEEE ASSP Magazine*, January 1996.
- [6] R. Dugad, and U. B. Desai, "A tutorial on Hidden Markov Models," *Technical report No.: SPANN-96.1*, May 1996.
- [7] T. Kanungo, "Hidden Markov Models," www.cfar.umd.edu/~kanungo.
- [8] W. Lee, and S. J. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," In *ACM Transactions on Information and System Security*, Vol. 3, No. 4, pages 227-261, November 2000.
- [9] C. C. Micheal and A. Ghosh, "Simple, state-based approaches to program-based anomaly detection," *ACM Transactions on Information and System Security*, Vol. 5, No. 3, August 2002, pages 203-237.
- [10] R. Sekar, M. Bendre, Dhurjati, and P. Bullineni, "A fast automaton-based method for detecting anomalous program behaviours," *IEEE Symposium on Security and Privacy*, 2001. S&P 2001, Page(s): 144 - 155.
- [11] Y. Qiao, X.W. Xin, Y. Bin and S. Ge, "Anomaly intrusion detection method based on HMM," *IEEE Electronic Letters Online No: 20020467*, 2002.
- [12] University of New Mexico's Computer Immune Systems Project, <http://www.cs.unm.edu/~immsec/systemcalls.htm>.
- [13] J. Pikoulas, W. J. Buchanan, M. Mannion, and K. Triantafyllopoulos, "An agent-based Bayesian forecasting model for enhanced network security," In *ECBS 2001 Proceedings on Engineering of Computer Based Systems - the Eighth Annual IEEE International Conference and Workshop*, pages 247-254, 2001.
- [14] T. Lane, and C. E. Brodley, *Approaches to Online Learning and Concept Drift for User Identification in Computer Security*. American Association for Artificial Intelligence press, 1998.