Viterbi Algorithm for Intrusion Type Identification in Anomaly Detection System

Ja-Min Koo and Sung-Bae Cho

Dept. of Computer Science, Yonsei University
Shinchon-dong, Seodaemoon-ku
Seoul 120-749, Korea
icicle@candy.yonsei.ac.kr,
sbcho@cs.yonsei.ac.kr

Abstract. Due to the proliferation of the infrastructure of communication networks and the development of the relevant technology, intrusions on computer systems and damage are increased, resulting in extensive work on intrusion detection systems (IDS) to find attacks exploiting illegal usages or misuses. However, many IDSs have some weaknesses, and most hackers try to intrude systems through the vulnerabilities. In this paper, we develop an intrusion detection system based on anomaly detection with hidden Markov model and propose a method using the Viterbi algorithm for identifying the type of intrusions. Experimental results indicate that the buffer overflow is well-identified, while we have some difficulties to identify the denial of service attacks with the proposed method.

1 Introduction

As the worldwide proliferation in network environments, a variety of faster services have become a reality. However, the higher the reliance on computers, the more crucial security problems such as the overflow or manipulation by external aggression occur. Korea Computer Emergency Response Team and Coordination Center (CERTCC-KR) reports that hacking damages are significantly increased from 1998: 1,943 attempts in 2000, 5,333 attempts in 2001 and 15,192 attempts in 2002 [1]. In addition, anyone who has little basic knowledge on hacking can easily intrude computer system with tools for hacking, which will lead to the increase of the damage by hacking in the near future. As demand and interest in intrusion detection are raised, the most active effort in this area has mainly developed the system security mechanisms like firewalls. Especially intrusion detection system (IDS) is one of them [2]. Intrusion detection techniques are divided into two groups according to the type of data they use: misuse detection and anomaly detection. The former uses the knowledge about attacks, and the latter uses normal behaviors [3].

The intrusion detection markets have been grown rapidly from 183 million dollars in 2000 to 422 million dollars in 2002 [4], so that new products related with IDS are released continuously. In 2001, there are 15 classes of IDS for commercial, 8 classes

K. Chae and M. Yung (Eds.): WISA 2003, LNCS 2908, pp. 97–110, 2004.

of IDS for information, 58 classes of IDS for research and 12 classes of IDS are released to Korean market. Most of IDSs have been developed to improve the detection rates, and they have some technical inertia such as inability of detecting the cause and the path of intrusion. As a result, when intrusion is happened, even if an intrusion is detected by IDSs, it takes long time to do appropriate actions. Moreover, it is hard to get the data that include the type of attempted intrusion mainly in specific system. Therefore, we cannot consider appropriate countermoves of how to cope with the attacks.

In this paper, normal behaviors are modeled by using system call events included in BSM (Basic Security Module) auditing data from Solaris. It checks the auditing data for detecting the intrusion, and the Viterbi algorithm traces the state sequence of current intrusion to compare with the sequences of known intrusions to determine the type of intrusions.

2 Background

2.1 Anomaly Detection System

There are four representative approaches based on anomaly detection system such as expert system approach, statistical approach, neural network approach and hidden Markov model (HMM), and Table 1 summarizes them.

Table 1. Research of the representative anomaly detection system

Organization	Name	Period	Approach			
			A	B	C	D
AT&T	Computer Watch [5]	1987-1990	X			
UC Davis	NSM [6]	1989-1995			X	
	GrIDS [7]	1995-	X			
SRI International	IDES [8]	1983-1992			X	
	NIDES [9]	1992-1995			X	
	EMERALD [10]	1996-			X	
CS Telecom	Hyperview [11]	1990-1995		X	X	
New Mexico Univ.	C.Warender et. Al [12]	1999			X	X
Yonsei Univ.	Park and Cho [13]	2002				X

(A: Expert system; B: Neural network; C: Statistical approach; D: HMM)

First of all, statistical approach is the most widely used in IDS. It uses some information such as login and logout time of the session, continuation of the resources and so on. It detects the intrusion by analyzing the time of resources and pattern of the command of normal users.

Second, NIDES (Next-generation Intrusion Detection Expert Systems), sets a standard of the intrusion detection with the similarities between profiles of short time and

long time [9]. It can be widely applied and it has very high detection rate because of processing based on the past data. However, it has some disadvantages such as insensitive for the event sequence of specific behavior and limited to modeling the intrusion behavior.

Third, neural network approach is similar to statistical one but it is easier to represent non-linear relationship. There is an example of Hyperview developed in CS Telecom, which consists of 2 neural network modules and an expert system module [11]. Neural network modeling is good at representing the non-linear relationship and training automatically but the relationship modeled is usually in blackbox.

Finally, HMM proposed by C. Warender in New Mexico University is good at modeling and estimating with event sequence of which is the underlying unknown. It performs better to model the system than any others [12]. However, it takes long time to train and detect intrusions. To overcome such disadvantages, we can extract the events of changing the privilege before and after them, and we are able to model the normal behavior with them. By using this method, we can reduce the time to model and maintain good performance [13].

2.2 Intrusion Type

The main purpose of intrusion is to acquire the authority of roots, and buffer overflow is one of the main vehicles to do that. An intruder uses the buffer overflow with a general ID that is hardly used or has no password, and acquires the root privileges with advanced techniques. Hence, we must analyze the type of intrusions to make the host-based IDS to detect the intrusions. Intrusion types mainly used are classified by Markus J. Ranum who developed the open software of firewall at first and CERTCC-KR extended the intrusions into 10 classes [1]. Among these intrusions, prevalent are 4 types of intrusions such as buffer overflow, denial of service, setup vulnerability and S/W security vulnerability.

- Buffer overflow: This intrusion type is one of the most difficult things to be dealt with, because we have to understand the execution sequence of specific programs and structures of memory, and memory or stack structures are dependent on operating system. However, anyone can download and use it easily, because source codes of the buffer overflow and bugs of operating systems are opened at internet. Consequently, this attack is widely used for hacking. Especially, in the case of root program which sets the SETUID, someone can easily acquire the shell having the authority of roots easily using buffer overflow [14]. There are three kinds of vulnerabilities of the buffer overflow.

  - xlock vulnerability: It is program that locks local x display until a password is entered. Due to in sufficient bound checking on arguments which are supplied by users, it is possible to overwrite the internal stack space of thx lock program while it is executing. By supplying a carefully designed argument to the xlock program, intruders may be able to force xlock to execute arbitrary command. As xlock is

setuid root, this may allow intruders to run arbitrary commands with root privileges.

- lpset vulnerability: Solaris's lpset allows setting of printing configuration in /etc/printers.conf or FNS. This product has been found to contain a security vulnerability that allows a local user to obtain root privileges.

- kcms_sparc vulnerability: Solaris contains support for the Kodak Color Management System (KCMS), a set of openwindows compliant API's and libraries to create and manage profiles that can describe and control the color performance of monitors, scanners, printers and film recorders. It also allows obtaining root privileges.

- S/W security vulnerability: This intrusion plays a major role in the bugs to cause the failures of programs: security errors on programming and on execution.

- Setup vulnerability: It happens when we setup a specific system without considering security. Some intrusions are included in this class related with various kinds of disadvantages of services provided by system. These can be used easily because there is no need of execution codes for hacking.

- Denial of service: It is used to make a specific system not to provide services normally. This is different from the usual intrusions which acquire the authority of system administrators. This attack inflicts a loss on specific host by exhausting all the resources of system not to provide service completely [15].

2.3 Solaris Basic Security Modules

Unix system provides the log information of wtmp, utmp and sulog in some directories such as /var/log/message or /var/adm/. It has some advantages to obtain easily whenever we want but some weak points. For example, when intruders succeed to intrude they can delete their log files. Due to this reason, we use the audit data of system call level to detect intrusions: Basic Security Module (BSM) which is provided by Sun Microsystems.

The algorithms presented in this paper which operate on audit data use logs produced by the BSM of the Solaris operating system. Each event generates a record, where the record is composed of different types of tokens depending on the event and the type of data needs to be included in the record.

The user audit ID is a useful piece of information included in the audit records of all events which can be attributed to a specific user. It is a unique identification number assigned to every user when they login and inherited by all processes descended from the login shell. This number allows an intrusion detection system to easily identify which user caused a specific event, even if that user has changed identities (through the su command, for example).

The log files are initially written in a binary format by the auditing process. Sun provides a tool, named praudit, which reads the binary log files and produces a human readable ASCII equivalent. This ASCII representation is what the algorithms working with the BSM data use. While having to convert to ASCII text slows down the algorithms somewhat, it makes coding and debugging much easier. Furthermore, once an algorithm has been prototyped and found promising, extending it to directly read the binary files would eliminate the praudit text conversion overhead [16]. Figure 1 shows the audit data record of BSM.

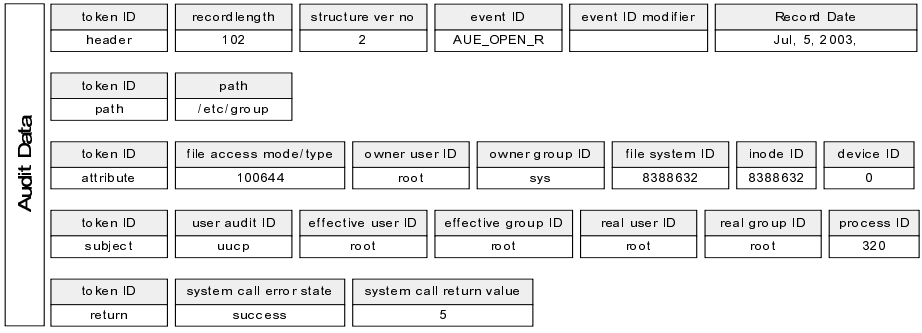


Fig. 1. BSM audit data record

3 Proposed Method

An IDS based on HMM collects and abridges normal auditing data, and it makes normal behavior models for a specific system. And then it detects intrusions with auditing data to detect the intrusions from it. Finally, to identify the type of intrusions, we analyze the state sequences of the system call events using the Viterbi algorithm.

Figure 2 shows the entire system structure. The proposed method consists of 4 parts: normal behavior modeling, intrusion detection, state sequence analysis and identification of the intrusion types.

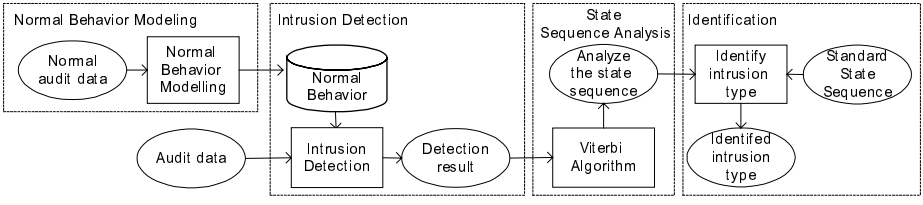


Fig. 2. System structure

3.1 Intrusion Detection

The basic log files can be easily obtained without any significant effort, but mostly it is hard to locate any evidence of privilege acquisition when buffer overflow attacks are attempted, and after a successful intrusion the attacker can easily erase his or her traces from those essential log files [17]. Because of the drawback, system call level audit data are used [12].

Especially, Sun Microsystems' BSM provides adequate representation of the behavior of programs, because any privileged activity that might be generated by a program is captured by BSM. Usually, audit trail from BSM consists of several measures. A system call, one of the measures from BSM, can be either perfectly normal or dangerous depending on situations. For example, the program attacked by buffer overflow generates system call events that are significantly different from the events generated at normal situation. Thus, we can detect intrusions effectively by building the model of normal system call events and noting significant deviation from the model. In this paper, we use the audit data of system call events from BSM.

3.1.1 Normal Behavior Modeling

In this paper, we use HMM to model the system auditing data, which is widely used for speech recognition or image recognition. HMM can be applied to model the sequence of system call events because it is very useful for modeling the sequence information [18].

An HMM λ is described as $\lambda=(A,B,\Pi)$. An HMM is characterized by a set of states Q, a set of possible observation symbols V, a number of observation symbols M, state transition probability distribution A, observation symbol probability B and initial state distribution Π [19]. The set of all system call events in audit data corresponds to that of possible symbol observations V, and a number of events correspond to M. The length of observation sequence T corresponds to the length of windows. The type of HMM that we use is a left-to-right model, which is known for modeling temporal signals better than any other models [20].

This phase is to model the normal behavior, which determines HMM parameters to maximize $P(O|\lambda)$ with which input sequence O, because no analytic solution is known for it, by an iterative method called Baum-Welch reestimation [18]. Figure 3 shows an example of the left-to-right model of HMM.

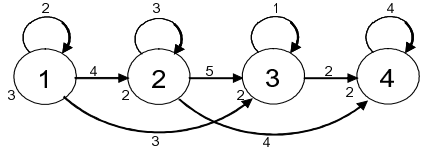


Fig. 3. Left-to-right model of HMM with 4 states

3.1.2 Intrusion Detection

Given λ, forward procedure can be used to calculate the probability $P(O|\lambda)$ with which input sequence O is generated out of model λusing forward variables [19]. The probability is used to decide whether normal or not with a threshold. Forward variable $a_t(i)$ denotes the probability at which a partial sequence $O_1, O_2, \cdots, O_t$ is observed and stays at state q_i.

$$\alpha_t(i) = P(O_1, O_2, \cdots, O_t, s_t = i \mid \lambda) \tag{1}$$

According to the above definition, $a_t(i)$ is the probability with which all the symbols in input sequence are observed in order and the final state reached at i. Summing up $a_t(i)$ can be calculated by the following procedure [18][19].

• Step 1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1) \tag{2}$$

• Step 2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{j=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \le t \le T \cdot \tag{3}$$

• Step 3. Termination:

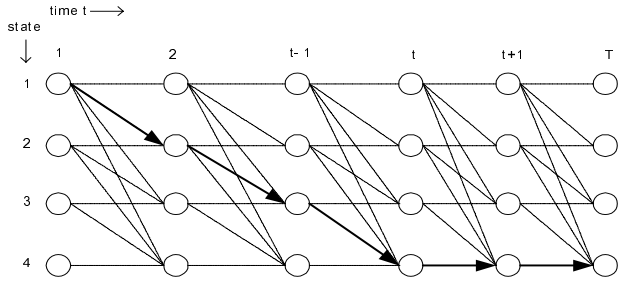
$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \cdot \tag{4}$$

Finally, if calculated value $P(O|\lambda)$ calculated log scale is smaller than the threshold, we decide that intrusion is occurred.

3.2 Intrusion Type Identification

3.2.1 Sequence Analysis with Viterbi Algorithm

The detected intrusion at the previous phase triggers to analyze the current state sequence of events. To identify the type of intrusions, we must know the state sequence, but HMM does not provide the state sequence explicitly. However, we can estimate the state sequence of the most probable ones using the Viterbi algorithm which finds the most-likely state transition path in a state diagram, given a sequence of symbols [20, 21]. It has been applied to speech and character recognition tasks where the observation symbols are modeled by HMM [22, 23]. The Viterbi algorithm can be easily combined with other information in real-time. In this paper, we apply the Viterbi algorithm to find optimal state sequence. Figure 4 shows the result of the Viterbi algorithm based on the HMM in Figure 3.



(b)

Fig. 4. Process of the viterbi algorithm

In left-to-right model of HMM, transition can only take place in a left to right manner and just one state can be skipped, thus only state transition a_{ij}, a_{ij+1} and a_{ij+2}.
For example in Figure 3, because we must start the state having the highest initial state distribution, we can do in state 1 at time 1 and transit with the highest transition probabilities among state 1, 2, 3 and 4 at time 2.

The procedures of Viterbi algorithm are as follows.

• Step 1: Initialization

$$\begin{aligned} &\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \\ &\Psi_q(i) = 0 \end{aligned} \tag{5}$$

$\delta_1(i)$ is the probability that symbol O_1 occurs at time t=1 at state i. The variable $\psi_t(j)$ stores the optimal states. In Figure 2(a) can be seen above $\delta_1(i)$ assigns a number which is beside each state. In Figure 2(a), $\delta_1(1)$=3, $\delta_1(2)$=2, $\delta_1(3)$=2 and $\delta_1(4)$=2.

• Step 2: Recursion

$$\begin{aligned} &\delta_t(j) = \max_i [\delta_{t-1}(i)a_{ij}]\, b_j(O_t), \qquad 2 \le t \le T, 1 \le j \le N \\ &\Psi_t(j) = \arg\max_i [\delta_{t-1}(i)a_{ij}]\, b_j(O_t), \quad 2 \le t \le T, 1 \le j \le N \end{aligned} \tag{6}$$

$\delta_t(i)$ denotes the weight accumulated when we are in state i at time t as the algorithm proceeds, and $\psi_t(j)$ represents the state at time t-1 which has the lowest cost corresponding to the state transition to state j at time t. In Figure 2(a), $\delta_2(1)$=5, $\delta_2(2)$=7, $\delta_2(3)$=6 and $\delta_2(4)$ is not defined. Therefore, $\psi_2(2)$= 2.

• Step 3: Termination

$$\begin{aligned} &P^* = \max_{s \in S_F} [\delta_T(s)] \\ &S_T^* = \arg\max_{s \in S_F} [\delta_T(s)] \end{aligned} \tag{7}$$

At the final time T, there are N probabilities δ_t, t=1, 2, ···, N. The highest probability among these probabilities becomes the candidate for the optimal state sequence. S_T^*

stores the corresponding state. Now, the final task is to backtrack to the initial state following the variable named ψ_t .

• Step 4: Backtracking

$$s_t^* = \Psi_{t+1}(s_{t+1}^*), \quad t = T-1, T-2, \cdots, 1 \tag{8}$$

When step 4 is finished, we can get the optimal state sequence '1-2-3-4-4-4' as shown in Figure 4.

3.2.2 Intrusion Type Identification

After backtracking, the similarity is compared between the average state sequence for every intrusion type and the state sequence of current intrusion using Euclidean distance. The formula is as follows.

$$d = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{9}$$

By assigning the analyzed state sequence of current intrusion to x_i, and doing the state sequence of every intrusion type to y_i, we calculate the Euclidean distance d. The smaller the value is, the higher similarity is. Hence, the intrusion type that we have found has the least value d.

Table 2. The state sequence of intrusions of buffer overflow
(A: xlock, B: lpset, C: kcms_sparc)

	1	2	3	4	5	6	7	8	9	10
A	0	2	4	6	8	10	12	14	16	17
B	0	2	2	2	4	6	8	10	12	14
C	0	2	4	6	8	10	11	13	15	17

Table 3. The similarity with actual intrusion

	Distance
A – actual	1.732
B – actual	9.381
C – actual	0

Table 2 shows the state sequence of each type of intrusions. For instance, if the state sequence of current intrusion is "0-2-4-6-8-10-11-13-15-17" the similarity between kcms_sparc and actual intrusion becomes 0, between xlock and actual intrusion is 1.732, and between lpset and actual intrusion is 9.381 as shown in Table 3. Actual intrusion is identified as kcms_sparc intrusion, because the Euclidean distance is smaller than anything else.

4 Experimental Results

4.1 Experimental Environments

We have collected normal behaviors from six users for 2 weeks using Solaris 7 operating system. They have mainly used text editor, compiler and program of their own writing. In total 13 megabytes (160,448 events are recorded) of BSM audit data are used. Main types of intrusions are used: Buffer overflow gets root privileges by abusing systems' vulnerability. Denial of service is a kind of intrusions which disturbs to provide the service well. These are the main intrusions which are subject to happen in host based systems. The attack types and the number of attempts are shown in Table 4.

Table 4. Intrusion types and the number of trials

Attack Category	Intrusion Type	Count
Buffer Overflow	OpenView xlock Heap Overflow	9
	Lpset –r Buffer Overflow Vulnerability	7
	Kcms_sparc Configuration Overflow	4
Denial Of Service	Process creation	9
	Exhausting the memory	7
	Fill the Disk	9

4.2 Results

We conduct experiments of the HMM with different number of states and observation lengths. Table 5 shows the result of the HMM-based IDS, and we experiment with the

Table 5. The performance of HMM-based IDS (no. of states is 20)

Length	Threshold	Detection Rate	F-P Error
10	-9.43	100%	2.626
15	-9.43	100%	3.614
10	-14.42	100%	1.366
15	-14.42	100%	2.718
10	-16.94	100%	0.789
15	-16.94	100%	2.618
10	-18.35	100%	0.553
15	-18.35	100%	2.535
10	-19.63	100%	0.476
15	-19.63	100%	2.508
10	-20.83	100%	0.372
15	-20.83	100%	2.473

threshold value of -20.83 which minimizes the false-positive error rate. The state symbol sequence of each intrusion type is as shown in Figure 5, where their values are from 30 runs.

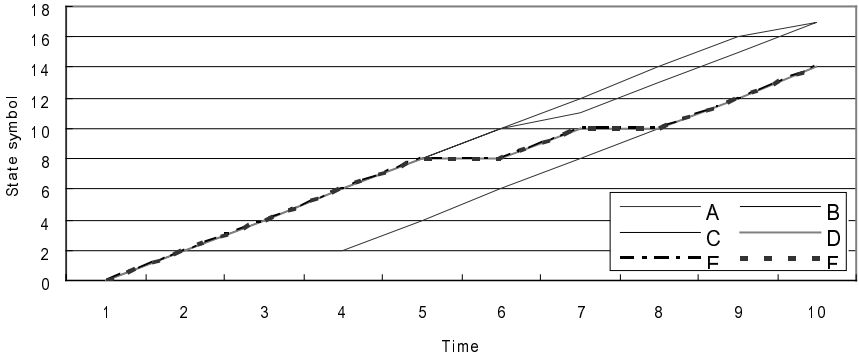


Fig. 5. State transition with 20 states and observation length of 10 (A:xlock, B: lpset, C: kcms_sparc, D: processe creation, E: fill the disk, F: exhausting the memory)

We make experiments to identify the intrusion type with auditing data for detecting the intrusion. Table 6 and Table 7 show the results.

Table 6. Result when state is 20 and observation length is 10

	A	B	C	D	E	F	Rate
A	8	1	-	-	-	-	88%
B	-	6	1	-	-	-	86%
C	-	-	4	-	-	-	100%
D	-	-	-	3	-	6	33%
E	-	-	-	4	-	3	0%
F	-	-	-	2	1	6	86%

Table 7. Average of results

Attack	Trial	Correct	Incorrect	Rate
Buffer Overflow	20	18	2	90%
Denial of Service	25	9	16	36%
Total	45	27	18	60%

A, B, C, D, E and F are the type of intrusions such as xlock Heap overflow, lpset overflow, kcms_configure buffer overflow, processes creation, fill the disk intrusion and exhausting the memory, respectively. Columns indicate the type of intrusions that we use actually and rows indicate the identified type of intrusions from experiments.

As a result, buffer overflow such as xlock, lpset and kcms_sparc intrusion are identified effectively. On the other hand, it is hard to identify the specific intrusion type of

denial of service. The state sequence is analyzed to find the reason. In case of having the low identification rate for the intrusions of denial of service is relatively low, because their state sequences are very identical.

The proposed method makes mistakes for processes creation to exhaust memory over 60%. Also, it misses filling the disk to process creation and exhausting memory over 40% and 50%, respectively. We calculate the similarity among three kinds of intrusions with Euclidean distance. We use the standard state sequence for calculating the similarity for every intrusion type. The distance value is 0 between exhausting the memory and filling the disk, and the one is 0 between exhausting memory and processes creation, and the last is 0 between filling the disk and processes creation. Three values of Euclidean distance are 0, so that we discover that three intrusion types are very similar.

Similarly, the number of states, one of the most important variables, has an effect on identifying the intrusion type. We carry out the experiments with different number of states to 5, 10, 15 and 20, respectively. However, we cannot identify the type of intrusions, because the state sequence is identical when the number of states is 5, 10 and 15. On the other hand, the state sequences are different for every intrusion type in case of the number of states is 20.

5 Concluding Remarks

In this paper, we have proposed a method to identify the type of intrusions in the anomaly detection system based on HMM. The proposed process calculates the Euclidean distance to compare the similarity between the standard state sequence and current state sequence which are obtained by using Viterbi algorithm when intrusion occurs. Experiments are executed in the intrusion detection system based on HMM with 100% intrusion detection rates. We change the number of states from 5 to 30 and the length of observation symbols from 10 to 20 in the experiments. As a result, the system detects all the intrusions when the number of states is more than 20.

However the deviation of identification rates is very extreme for the type of intrusions. Especially, identification rates of intrusions belonging to denial of service are very low, because the state sequence among three intrusion types – processes creation, exhausting the memory and filling the disk – are identical. In addition, the number of states which is one of the most important variables has an effect to identify the type of intrusions. The proposed system needs more than 20 states. Moreover, it has difficulty to identify various types of intrusions of denial of service. Therefore, we must investigate to identify types of intrusions with HMM in smaller number of states.

Acknowledgement. This research was supported by University IT Research Center Project.

References

1. CERTCC-KR, Korea Computer Emergency Response Team and Coordination Center, http://www.certcc.or.kr, 2003.
2. H. S. Vaccaro and G. E. Liepins, "Detection of anomalous computer session activity," *In Proc. of IEEE Symposium on Research in Security and Privacy*, pp. 280–289, 1989.
3. T. F. Lunt, "A survey of intrusion detection techniques," *Computers & Security*, vol. 12, no. 4, pp. 405–418, June 1993.
4. IDC, *Plugging the Holes In eCommerce: The Market for Intrusion Detection and Vulnerability Assessment Software,* 1999-2003.
5. C. Dowel and P. Ramstdet, "The computer watch data reduction tool," *In Proc. of the 13th National Computer Security Conference,* pp. 99-108, Washington DC, USA, October 1990.
6. T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood and D. Wolber, "A network security monitor," In *Proc. of the 1990 IEEE Symposium on Research in Security and Privacy,* pp. 296–304, Los Alamitos, CA, USA, 1990.
7. S. Stanford-Che, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle, "GrIDS-A graph based intrusion detection system for large networks," *In Proc. of the 19th National Information Systems Security Conference,* vol. 1, pp. 361–370, October 1998.
8. T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali and P. G. Neuman, "A real-time intrusion-detection expert system (IDES)," *Technical Report Project 6784,* CSL, SRI International, Computer Science Laboratory, SRI International, February 1992.
9. D. Anderson, T. F. Lunt, H. Javits, A. Tamaru and A. Valdes, "Detecting unusual program behavior using the statistical components of NIDES," *NIDES Technical Report,* SRI International, May 1995.
10. P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," *In Proc. of the 20th National Information Systems Security Conference,* pp. 353–365, Baltimore, Maryland, USA, October 1997.
11. H. Debar, M. Becker and D. Siboni, "A neural network component for an intrusion detection system," *In Proc. of 1992 IEEE Computer Society Symposium on Research in Security and Privacy,* pp. 240–250, Oakland, CA, May 1992.
12. C. Warrender, S. Forrest and B. Pearlmutter, "Detecting intrusion using calls: Alternative data models," *In Proc. of IEEE Symposium on Security and Privacy,* pp. 133–145, May 1999.
13. S.-B. Cho and H.-J. Park, "Efficient anomaly detection by modeling privilege flows using hidden Markov model," *Computers & Security*, vol. 22, no. 1, pp. 45–55, 2003.
14. D. Larochelle and D. Evans, "Statically detecting likely buffer overflow vulnerabilities," *In Proc. of USENIX Security Symposium,* pp. 177–190, August 2001.
15. F. Lau, S. H. Rubin, M. H. Smith and L. Trajkovic, "Distributed denial of service attacks," *2000 IEEE International Conference on Systems, Man and Cybernetics,* pp. 2275–2280, 2000.
16. S. E. Webster, "The development and analysis of intrusion detection algorithms," *S. M. Thesis, Massachusetts Institute of Technology,* June 1998.
17. S. Axelsson, "Research in intrusion detection system: A survey," *Chalmers University of Technology*, 1999.
18. L.R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. of IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.
19. L. R. Rabiner and B .H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine,* vol. 3, no. 1, pp. 4–16, 1986.

20. G. D. Forney Jr., “Maximum-likelihood sequence detection in the presence of intersymbol interference,” *IEEE Transactions on Information Theory,* vol. 18, no. 30, pp. 363–378, May 1972.
21. G. D. Forney Jr., “The viterbi algorithm,” *Proc. of IEEE,* vol. 61, no. 3, pp. 268–278, March 1973.
22. L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition,* Chapter 6, Prentice Hall, Englewood Cliffs, New Jersey, 1993.
23. J. Picone, “Continuous speech recognition using hidden markov models,” *IEEE ASSP Magazine,* vol. 7, no. 3, pp. 26–41, July 1990.