Efficient anomaly detection by modeling privilege flows using hidden Markov model

Sung-Bae Cho and Hyuk-Jang Park

Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea, {sbcho, twinkler}@candy.yonsei.ac.kr

Abstract

Anomaly detection techniques have been devised to address the limitations of misuse detection approaches for intrusion detection with the model of normal behaviors. A hidden Markov model (HMM) is a useful tool to model sequence information, an optimal modeling technique to minimize false-positive error while maximizing detection rate. In spite of high performance, however, it requires large amounts of time to model normal behaviors and determine intrusions, making it difficult to detect intrusions in real-time. This paper proposes an effective HMM-based intrusion detection system that improves the modeling time and performance by only considering the privilege transition flows based on the domain knowledge of attacks. Experimental results show that training with the proposed method is significantly faster than the conventional method trained with all data, without loss of detection performance.

Keywords: Anomaly detection; Privilege flows; System calls; Hidden Markov model; Intrusion detection

1. Introduction

Due to worldwide proliferation in network environments, a variety of faster services have become a reality. However, the higher the reliance on computers, the more important security problems become. Particularly, the intrusions to the network infrastructures on finance, defense, and electric power should be detected to minimize damage. As the demand and interest in intrusion detection have increased, the most active effort in this area has been mainly on the development of system security mechanisms like firewalls. However, because it is noted as nearly impossible to design an impenetrable security system, many intrusion detection systems (IDS) have been recently developed.

IDSs detect unauthorized usage and misuse by a local user, and modification of important data by analyzing system calls, system logs, activation time, and network packets of each operating system [1]. Most server computer systems support C2 security audit systems that can obtain detailed information on events generated within the system to aid in collecting information [2].

Although IDS technology has not fully matured yet, various experimental products have been developed in laboratories and companies. In general, IDS can be classified into two categories depending on the modeling methods used. One is misuse detection or knowledge-based method that looks for evidence of malicious behavior matched against equipped (or modeled) description of an attack. The other is anomaly detection or action-based method that builds a profile of normal behaviors and attempts to identify patterns of activity that deviate from the defined profile [3]. Misuse detection systems can detect known attacks efficiently, but it is nearly impossible to identify new attacks. On the other hand, anomaly detection systems have the advantage of being able to detect unknown attacks that exploit unpredictable vulnerability. The principal techniques used in anomaly detection are statistical approach, rule-based expert systems, neural networks, immune system, and hidden Markov model (HMM).

NIDES is a typical system that adopts a statistical approach. It observes the distribution of the audit data and compares the conformity of long-term event patterns and recent event patterns in the profile [4]. But this system is hardly practical due to the limitation of the

Computers & Security
Vol 22, No 1, pp 45-55, 2003
Copyright ©2003 Elsevier Science Ltd
Printed in Great Britain
All rights reserved
0167-4048/03

Sung-Bae Cho

Sung-Bae Cho received the B.S. degree in computer science from Yonsei University, Seoul, Korea, in 1988 and the M.S. and Ph.D. degrees in computer science from KAIST (Korea Advanced Institute of Science and Technology), Taejeon, Korea, in 1990 and 1993, respectively.

He worked as a Member of the Research Staff at the Center for Artificial Intelligence Research at KAIST from 1991 to 1993. He was an Invited Researcher of Human Information Processing Research Laboratories at ATR (Advanced Telecommunications Research) Institute, Kyoto, Japan from 1993 to 1995, and a Visiting Scholar at University of New South Wales, Canberra, Australia in 1998. Since 1995, he has been an Associate Professor in the Department of Computer Science, Yonsei University. His research interests include neural networks, pattern recognition, intelligent man-machine interfaces, evolutionary computation, and artificial life.

Dr. Cho was awarded outstanding paper prizes from the IEEE Korea Section in 1989 and 1992, and another one from the Korea Information Science Society in 1990. He was also the recipient of the Richard E. Merwin prize from the IEEE Computer Society in 1993. He was listed in Who's Who in Pattern Recognition from the International Association

number of attack descriptions. In the Los Alamos National Laboratory, NADIR and Wisdom & Sense systems have been developed, which are tools for modeling expert knowledge with rule-based language [5]. The systems are based on rule sets that describe appropriate behavioral decision to examine user actions and to identify those actions that deviate from normal patterns. It also inspects whether the security policy is properly applied. However, an expert system has a drawback on the abstraction of information of attacks and normal patterns. A typical IDS that uses neural networks is Hyperview designed by CS Telecom in France. User's action-learning with neural network maps data window to neural input values: Input values are command name, CPU usage, memory usage and 60 other audit data. But it cannot explain the connection between input and output values, and it also takes a long time in modeling [6]. Moreover, an immune system could not detect illegal access acquired from legal transactions [7].

An IDS based on HMM for modeling and evaluating invisible events based on system calls was originally designed by S. Forrest at New Mexico University [8], and deployed by Arizona State University, RAID at New Mexico University, and Hong Kong University of Science and Technology. HMM is well known as a good probabilistic method to model temporal sequences, and is largely exploited in the speech recognition field. The proposed systems, based on HMM, show substantial decrease of false-positive error with increased detection rate compared to other anomaly detection techniques, but require huge amounts of time to model normal behaviors and detect intrusions and their error rates are relatively higher than misuse detection techniques [9, 10].

This paper presents an optimal measure abstraction method that analyzes various attack patterns and excludes unnecessary data to ameliorate system performance of HMM-based anomaly detection systems. The basic idea is in the use of privilege transition flows to model HMM so as to optimiz the intrusion detection rate while minimizing the false alarm rate. The notion of examining privilege transition flows has been discussed and established by Vaccaro [1, 11] and Smaha [12], even though they reduce voluminous system audit trails to short summaries of user behavior, anomalous events, and security incidents to help the systems security officer detect and investigate intrusions. This paper exploits this notion in the modeling of a machine learning tool, HMM.

2. Measure extraction in privilege flows

The ultimate vehicle of intrusion is to acquire root privilege and the most prevalent attack is known as buffer overflow. This attack penetrates into a system with unused user accounts or an account without any password, and acquires root privilege with advanced attack techniques. Here, we summarize the host-based attack types and address appropriate measures to each type. Most enterprises and institutions are aware of hacking types classified by Marcus J. Ranum, who developed free software, firewalls; the CERTCC-KR partially extended this to 10 categories [13]. Among those, buffer overflow, S/W security error, configuration error, denial-of-service attacks are included for host attacks.

The hacking report for January and February 2002 provided by CERTCC indicates that 44 attempts of host-based attacks such as S/W security error, buffer overflow, configuration error, and denial-of-service were made in January and 45 attempts in February 2002, as shown in Table 1. Moreover, buffer overflow vulnerability and user privilege configuration error comprised more than 90% of the attempts made.

In this paper we have collected 17 attack patterns for four attack types to observe the characteristics of attack behaviors and analyzed

Table 1: Recent report of CERTCC on the number of attacks.

Attack type	DEC 2001	JAN 2002	FEB 2002	MAR 2002
S/W security vulnerability	2	0	0	1
Buffer overflow	16	35	38	31
Control, setup vulnerability	3	4	4	3
Denial of service	3	4	6	5
Total	24	43	45	40

event information according to the attack types when an intrusion has occurred. Table 2 lists the exploit codes used in intrusion analysis. The analysis confirms that in all buffer overflow and S/W security error attacks, the effective user ID (EUID) has remained as root privilege even after user ID (UID) and EUID have changed.

Xlock, a typical buffer overflow attack, is a program that locks X Window display, and because it does not thoroughly examine the length of inputted parameters it can vulnerably overwrite the inner stack region of program. If you look at the attack file, SETUID is set to root giving the attacker the opportunity to execute an arbitrary program with root privilege. The core difference between a normal xlock file and an exploit file in execution is in the process of transition of EUID privilege. Generally, after the privilege flow, EUID changes to local user with the call to exit event, but an attack file executes root shell with execve call and maintains root privilege.

Therefore, we realize that the majority of host-based attacks exploit system bugs or a local user's misuse to acquire root privilege. Therefore, if we model information on the acquisition of normal root privilege to detect any indication of an attack, we can monitor more than 90% of illegal privilege flow attacks excluding denial-of-service attacks and this

Table 2: Attack types used in our analysis.

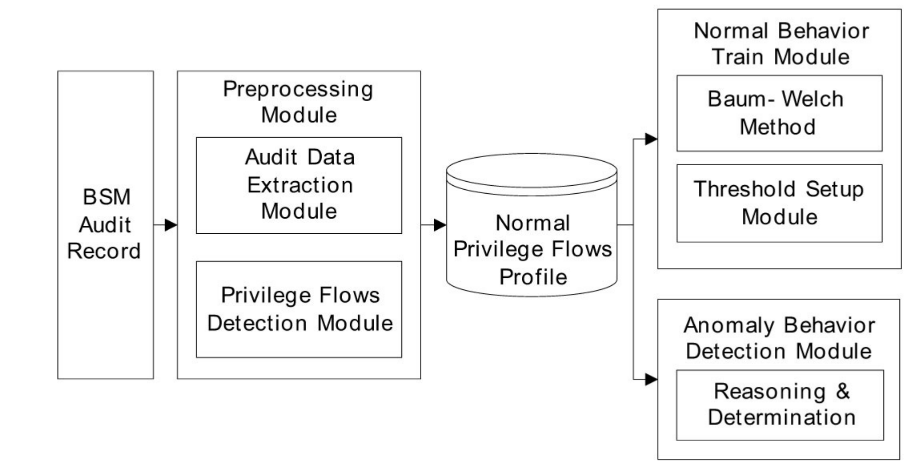
Version	Attack type	Attack descriptions
2.5.1	Buffer overflow	Solaris ufsrestore vulnerability
	Buffer overflow	rpcbind file overwrite vulnerability
	Buffer overflow	Libc (getopt() bug) stack overflow
	Buffer overflow	Eject exploit
	Buffer overflow	Passwd stack overflow
	Buffer overflow	Buffer overflow in /bin/fdformat
	Control error	Non-stealthy attack which resets IFS for a normal user and creates a root shell
2.7	Buffer overflow	OpenView xlock heap overflow
	Buffer overflow	Lpset -r buffer overflow
	S/W security vulnerability	DTMail mail environment variable buffer overflow
	Buffer overflow	Libc2_list_devices exploit
	Buffer overflow	Ufsrestore buffer overflow
2.8	Denial of service	Memory deplete, Process overflow
	Buffer overflow	Whodo buffer overflow
	Buffer overflow	Kcms_configure KCMS_PROFILES buffer overflow
	S/W security vulnerability	Mailx -F buffer overflow
	Buffer overflow	Libsldap buffer overflow

for Pattern Recognition in 1994, and received the best paper awards at International Conference on Soft Computing in 1996 and 1998. Also, he received the best paper award at World Automation Congress in 1998, and listed in Marquis Who's Who in Science and Engineering in 2000 and in Marquis Who's Who in the World in 2001. He is a Member of the Korea Information Science Society, INNS, the IEEE Computer Society, and the IEEE Systems, Man, and Cybernetics Society.

Hyuk-Jang Park

Hyuk-Jang Park received the B.S. degree in statistics from Dong-Guk University, Seoul, Korea, in 2000, and the M.S. degree in computer science from Yonsei University, Seoul, Korea, in 2002. His current research interests include intrusion detection systems, computer security, and performance evaluation.

Figure 1: The overview of intrusion detection system.

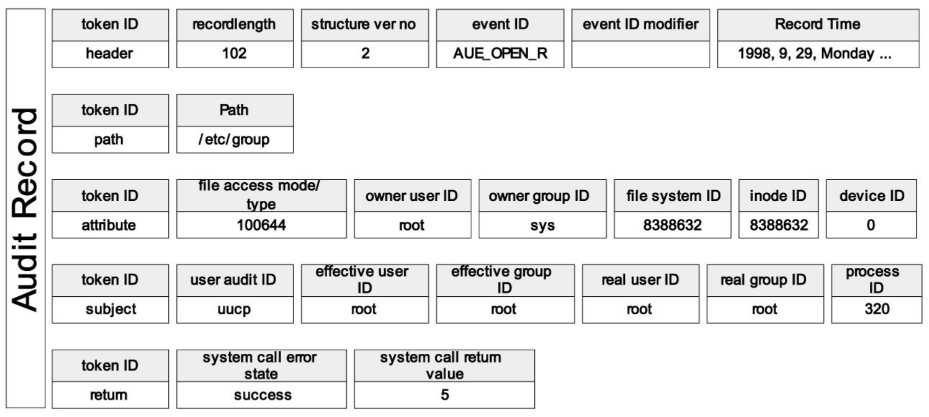

BSM Audit Record
Preprocessing Module
Audit Data Extraction Module
Privilege Flows Detection Module
Normal Privilege Flows Profile
Normal Behavior Train Module
Baum- Welch Method
Threshold Setup Module
Anomaly Behavior Detection Module
Reasoning & Determination

indicates that the majority of host-based attacks can be detected. In addition, if we effectively minimize the number of targets of intrusion detection, we can also minimize the consumption of system resources that makes it possible for practical operation in the real world.

3. Privilege flows modeling

Figure 1 depicts the overall structure of our IDS based on user privilege flows. Basically, the system consists of preprocessing module, normal behavior modeling module, and intrusion detection module. Normal behavior modeling module generates HMM-based normal behavior models from collected normal audit data and intrusion detection module determines whether an intrusion has occurred by comparing accumulated target audit data with the normal behavior model. The core technology in the IDS lies in the use of hidden Markov model for modeling normal behaviors and filtering of audit data from abstracted information around the change of privilege flows.

Figure 2: BSM audit record format.


Audit Record

token ID	recordlength	structure ver no	event ID	event ID modifier	Record Time
header	102	2	AUE_OPEN_R		1998, 9, 29, Monday ...

token ID	Path
path	/etc/group

token ID	file access mode/ type	owner user ID	owner group ID	file system ID	inode ID	device ID
attribute	100644	root	sys	8388632	8388632	0

token ID	user audit ID	effective user ID	effective group ID	real user ID	real group ID	process ID
subject	uucp	root	root	root	root	320

token ID	system call error state	system call return value
return	success	5

3.1 Collection of audit record

The easiest way to collect and refine audit data is to use system log files. In UNIX systems, user log information such as wtmp, utmp, and sulog files is in /var/log/message or /var/adm/ directory. 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/her trace from those essential log files [14]. Because of such a drawback, system call level audit data are used. Basic security module provides audit data with the information as shown in Figure 2. The audit data collector collects related data from basic security modules and saves it in memory in a form that is prepared for later use as audit data. There are 242 types of system audit data where ID numbers up to 255 are reserved for system calls and ID numbers after 1000 are for user-defined system calls.

3.2 Preprocessing of audit record

In UNIX systems, several users share fixed amounts of disk and memory in a single system so that each user must have their own privilege. However, the problem of acquiring super user privilege caused by program error or misuse of a system is always possible. A normal privilege flow happens when an administrator logs in through a local user account and acquires root privilege using SU command or by the local user who executes temporarily a file whose SETUID is set to super user. SETUID is a rule applied to the file that temporarily changes the privilege of the user, and anyone who executes that file will execute the file with the same SETUID [15].

Most host attacks exploit vulnerabilities described as above. Figure 3 is an example of privilege flow when a buffer overflow attack is in action. Normally, if a user executes fdformat command from a normal state (Q0), the command is executed temporarily with super user privilege (Q1), and on termination of the task, it returns to its original state (Q0). However, if buffer overflow attack occurs during the transaction, it does not return to its original state (Q0), but retains super user privilege (Q2) and ultimately the user can access the system with super user privilege. But if the security module knows the sequence of normal privilege flow related to fdformat command, it can identify the transition to an abnormal state (Q2) caused by buffer overflow. The learning model for user privilege flows collects the sequence of normal system calls that happens before the occurrence of normal privilege change. Then, it models normal behaviors and detects intrusion by comparing each user's state of privilege flow.

3.3 Extraction of privilege flows

The filtering subsystem imports the reformatted audit data from the preprocessor and determines which system calls are eligible and which audit data fields are sufficiently available. From this

Figure 3: An example of privilege change by buffer overflow attack.

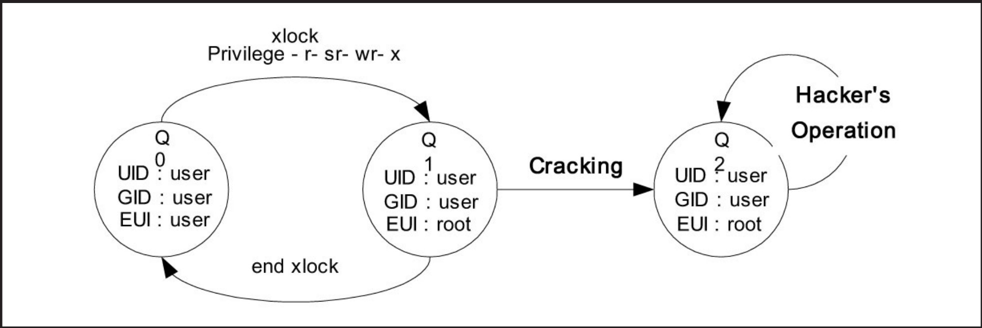


Table 3. Events related with privilege flows

Event ID	System call	Event ID	System call	Event ID	System call
2	fork	27	setpgrp	200	setuid
11	chown	38	fchroot	214	setegid
16	stat	39	fchown	215	seteuid
21	symlink	40	setreuid	237	lchown
23	execve	41	setrgid	6158	rsh
24	chroot	69	fchroot	6159	su

preliminary study, we analyze the data to detect privilege change from regular user to root or another user. By examining the results, we can assure that many events are not in use, but only 25% of the events have been used. In training an HMM, running time is roughly proportional to the number of different system calls used. For this reason, among the 267 different events audited by BSM, only 80 are used.

It can be a good reduction technique that extracts the privilege transition flows. Table 3 shows the events related with privilege flows. Because it takes significantly longer time to train abundant data using system calls, the reduction technique can save computational cost radically. Proposed privilege change model is evaluated with the fixed data sequence bases on the situation where transitions between UID and EUID occur, which are derived from BSM data. However, during the detailed analysis of theses attacks, acquiring root privilege can be not only from the change of user but also from that of group. Considering these cases, privilege flows of both user and group must be detected.

3.4 Anomaly detection

Anomaly detection matches current behavior against the normal behavior models and calculates the probability with which it is generated out of each model. In this paper, we have utilized hidden Markov model. An HMM is a doubly stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [16, 17]. This is a useful tool to model sequence information and can be thought of a graph with N nodes called ‘state’ and edges representing transitions between those states. Each state node contains initial state distribution and observation probability at which a given symbol is to be observed. An edge maintains a transition probability with which a state transition from one state to another state will be made.

Given an input sequence, HMM can model this sequence with its own probability parameters using Markov process though state transition process cannot be seen outside. Once a model is built, the probability with which a given sequence is generated from the model can be evaluated. A model λ is described as λ = (A, B, π) using its characteristic parameters. The parameters used in HMM are as follows:

T = length of the observation sequence
N = number of states in the model
M = number of observation symbols
$Q = \{q_1, q_2, \ldots, q_N\}$, states
$V = \{v_1, v_2, \ldots, v_M\}$, discrete set of possible symbol observations
$A = \{a_{ij} \mid a_{ij} = \Pr(q_j \text{ at } t+1)\}$, state transition probability distribution
$B = \{b_j(k) \mid b_j(k) = \Pr(v_k \text{ at } t \mid q_j \text{ at } t)\}$, observation symbol probability disribution
$\pi = \{\pi_i \mid \pi_i = \Pr(q_i \text{ at } t = 1)\}$, initial state distribution

Given an HMM λ, forward-backward procedure can be used to calculate the probability $\Pr(O|\lambda)$ with which input sequence O is generated out of model λ using forward variables. Forward variable α denotes the probability at which a partial sequence O_1, O_2,..., O_T is observed and stays at state q_i.

$$\alpha_t(i) = \Pr(O_1, O_2, \ldots, O_t, i_t = q_i \mid \lambda)$$

According to the above definition, $\alpha_t(i)$ is the probability with which all the symbols in input sequence are observed in order and the final state reaches at i. Summing up $\alpha_i(i)$ for all i yields the value $\Pr(O|\lambda)$. $\alpha_t(i)$ can be calculated by the following procedure [16][17].

Step 1. Initialization :

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

Step 2. Induction :

for $t = 1$ to $T-1$

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

Step 3. Termination :

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

3.5 Normal behavior modeling

The behavior modeling can be derived using simple 'occurrence counting' arguments or using calculus to maximize the auxiliary quantity. With maximum-likelihood estimation (ML), we can maximize the probability of given sequence of observations. Generally, parameter estimation for HMM is performed using a standard Baum-Welch algorithm with the ML criterion. The Baum-Welch algorithm for HMM is simple, well-defined, and stable. This requires additional two variables: $\zeta_t(i, j)$ is defined as the probability with which it stays at state q_i at time t and stays at state q_j at time t+1.

$$\begin{aligned}\xi_t(i,j) &= \Pr(i_t = q_i, i_{t+1} = q_j \mid O, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\Pr(O \mid \lambda)}\end{aligned}$$

Backward variable $\beta_t(i)$ can be calculated using similar process to that of α as follows.

Step 1. Initialization :

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

Step 2. Induction :

for $t = 1$ to $T-1$

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

Step 3. Termination :

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

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

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

Summing up the two variables over time t respectively, we can get the probability with which state i transits to state j and the expectation that it stays at state i. Given the above variables calculated, a new model $\bar{\lambda} = (\bar{A}, \bar{B}, \pi)$ can be adjusted using the following equations:

$$\begin{aligned}\bar{\pi}_i &= \text{expected frequency (number of times) in state } i \text{ at time } (t=1) \\ &= \gamma_1(i) \\ \bar{a}_{ij} &= \frac{\text{expected number of transistions from state } i \text{ to state } j}{\text{expected number of transistions from state } i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ \bar{b}_j(k) &= \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \\ &= \frac{\sum_{\substack{t=1 \\ s.t. O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}\end{aligned}$$

After $\bar{\lambda}$ is adjusted from sequence O, $\Pr(O|\bar{\lambda})$ is compared against $\Pr(O|\lambda)$. If $\Pr(O|\lambda)$ is greater than $\Pr(O|\bar{\lambda})$, it implies that a critical point in likelihood has been reached, thereby finishing the re-estimation. Otherwise, $\Pr(O|\lambda)$

Table 4: The exploit code used in experiment.

Attack name	Attack descriptions
ufsrestore	The ufsrestore utility is to set uid as root by default, and vulnerable to a buffer overflow attack
fbconfig	Buffer overflow using the fbconfig UNIX system command leads to root shell
libc getopt()	A vulnerability has been identified in the getopt (3) function in the libc library
eject	Buffer overflow using eject program on Solaris leads to a user→root transition if successful
passwd	Under Solaris passwd, yppasswd and nispasswd can be overflowed in an internal function
fdformat	Buffer overflow using the fdformat UNIX system command leads to root shell

Table 5: Comparison of running time.

Modeling	State/sequence	Number of sequence	Timestamp
All data	5/20	767218	5 hours 07 min
All data	15/30	767208	6 hours 29 min
Privilege change data	5/20	5950	26.3 sec
Privilege change data	15/30	5792	57.5 sec

is substituted by $\Pr(O|\bar{\lambda})$ and re-estimation continues.

4. Experimental results

To model the HMM with normal behaviors, we have collected audit data from 10 users who have conducted several transactions such as programming, navigating Web pages, and transferring FTP data for one month. Because it is crucial to define normal traffic for IDS based on anomaly detection, we have attempted to balance the data with every-day usage of computer systems. The number of users and period of collection time should be extended at the future work.

In this paper, the BSM auditing tool of Sun Solaris operating environment is used and a total of 767 237 system calls have been collected. To get this normal data, we have gathered audit data from Sun UltraSparc 10 that installed Solaris 2.5.1. The performance of IDS is measured against 19 times u2r (user-to-root) intrusions with the observation of the variation of the Receiver Operating Characteristic (ROC) curve. Most attacks are attempted to acquire super user privilege by exploiting a subtle temporary file creation and

Table 6: The anomaly detection rate for two modeling methods against 10 attacks.

Modeling	State/sequence	HMM threshold	Detection rate	F-P error
All data	5/20	-92.4	100%	4.234%
	10/25	-102.1	100%	3.237%
	10/20	-93.7	100%	1.6165%
	15/30	-118.3	100%	12.426%
Privilege change data 6.707%		5/20	-53.8	100%
	15/25	-65.2	100%	2.367%
	10/27	-73.1	100%	2.439%
	10/30	-81.2	100%	0.602%

Table 7: Comparison of d' values.

State/sequence	Modeling	Detection rate	F-P error	d'
5/25	All	100%	13.80%	3.4157
	Privilege change	100%	2.974%	4.2174
5/30	All	100%	22.10%	3.0952
	Privilege change	100%	2.931%	4.2110

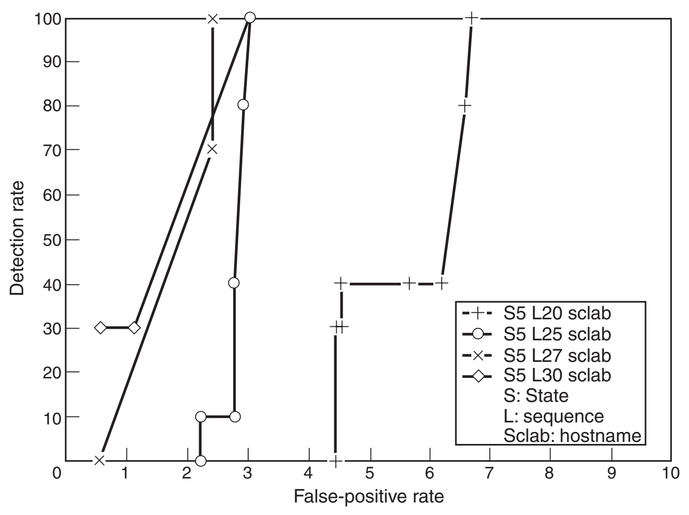
race condition bug. The exploit codes used in our experiment are as shown in Table 4.

To show the usefulness of the proposed method, we compare the performance of modeling meth-od with partial sequence data only around the privilege transition and that with all sequence data. At first, we have conducted experiments to decide the optimal HMM parameters for effective intrusion detection. A desirable intrusion detection system would show high intrusion detection rate with low false-positive error rate. The detection rate is the percentage of attacks detected, and the false-positive error rate is the percentage of normal data that the system alarms as attack. If the false-positive error rate is high, the IDS may not be practical because users may be warned from the IDS frequently even with the normal behaviors.

Standard HMM has a fixed number of states, so that one must decide the size of the model before training. Preliminary study indicates that the number of optimal states roughly equals to that of distinct system calls used by the program [9]. A range of 5-15 states are examined in HMM, and Figure 4 shows the performance change of the system when the length of sequence changes as 20, 25, 27 and 30.

To measure the runtime-performance, the program is executed 10 times in UltraSparc 10 and the results of the shell are recorded using time command. Table 5 summarizes the results of time for modeling the HMMs with different settings. As mentioned previously, training an HMM requires very expensive computation, but the model with privilege transition flows has obtained approximately 250 times faster than

Figure 4: ROC for the model with privilege change data.



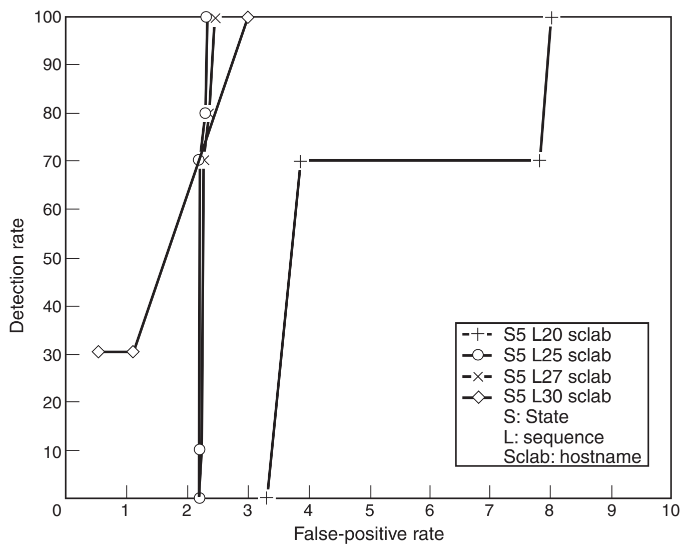
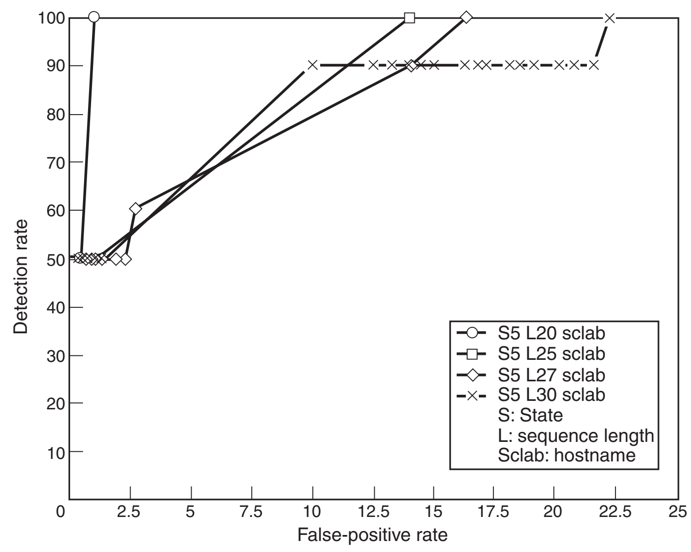
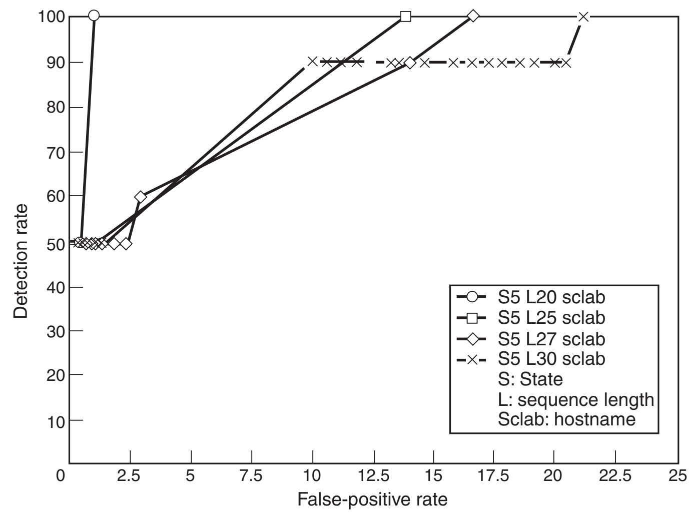


Figure 5: ROC for the model with all data.





that with all data. We can expect to reduce the time consumption substantially with privilege transition flows.

In addition to the running time detection rates are compared. Table 6 shows the intrusion detection rates and false-positive error rates for two cases. The results indicate that privilege transition flows data are useful to detect intrusions when we have modeled normal behaviors by HMM. The model with 10 states and 30 sequences in privilege transition flows data is seen to be the most effective in this experiment, whilst the model with 10 states and 30 sequences in all train data is not good enough. The lowest false-positive error has been obtained when the number of states is 10 and the number of sequences is 20 in all data.

As shown in Figures 4 and 5, the model with privilege transition flows yield higher performance than that with all data. There exists a clear gap between the minimum and maximum detection rates for the model with all data. In order to differentiate the results quantitatively d' values are also calculated for some of the different number of states and sequences in Table 7. As can be seen, the d' values for the model with privilege transition flows are significantly larger than those with all data. This result confirms the usefulness of the proposed method that only considers the privilege transition flows.

5. Concluding remarks

In this paper, we have proposed an anomaly detection-based IDS using the privilege transition flows data. Experimental results show that the training of the privilege transition flows is substantially faster than that of conventional data without any loss of detection rate. Also, modeling privilege change data has less error than conventional modeling. With sufficiently large numbers of training data, it might show better performance and reduction of computational costs. The proposed method can open a new way of utilizing computation-intensive anomaly detection technique in the real world, based on behavioral constraints imposed by security policies and on models of typical behavior for users.

Currently, the ROC curve that shows the change of the error rate as the threshold changes is adopted to demonstrate the detection capability of the system. For the deployment of the system in the real world, an automatic mechanism to adjust the threshold for HMM appropriately is needed. In the future, studies on the discriminative event extraction and modeling among user behaviors should be followed.

Acknowledgements

This paper was supported by a grant from the Ministry of Information and Communication in Korea.

References

[1] Vaccaro, H.S. and Liepins, G.E., 1989. Detection of anomalous computer session activity. *Proc. IEEE Symp. on Research in Security and Privacy*, 1989, pp. 280-289.

[2] Price, K.E., 1997. *Host-based misuse detection and conventional operating system's audit data collection.* MSc. Dissertaion, Purdue University, Purdue, IN, USA, 1997.

[3] Lunt, T.F., 1993. A survey of intrusion detection techniques, *Computers & Security*, Vol. 12, No. 4, 1993.

[4] Javitz, H.S. and Valdes, A., 1994. The SRI IDES statistical anomaly detector. *NIDES Technical Report*, 1994.

[5] Hochberg, J. et al., 1993. Nadir: An automated system for detecting network intrusion and misuse. *Computers & Security*, Vol. 12, No. 3, 1993, pp. 235-248.

[6] Debar, H., Becker, M. and Siboni, D., 1992. A neural network component for an intrusion detection system. *Proc. 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, USA, 1992, pp. 240-250.

[7] Hofmeyr, S. and Forrest, S., 2000. Architecture for an artificial immune system. *Evolutionary Computation Journal*, 2000.

[8] Warrender, C., Forrest, S. and Pearlmutter, B., 1999. Detecting intrusion using calls: Alternative data models. *IEEE Symposium on Security and Privacy*, May 1999.

[9] Choy, J. and Cho, S.-B., 2000. Intrusion detection by combining multiple hidden Markov models. *Lecture Note in Artificial Intelligence*, Vol. 1886, 2000, pp. 829.

[10] Yeung, D.Y. and Ding, Y., 2001. Host-based intrusion detection using dynamic and static behavioral models. *The Journal of the Pattern Recognition Society*, December 2001.

[11] Liepins, G.E. and Vaccaro, H.S., 1992. Intrusion detection: Its role and validation. *Computers & Security*, Vol. 11, No. 4, 1992, pp. 347-355.

[12] Smaha, S.E., 1988. Haystack: An intrusion detection system. *Aerospace Computer Security Applications Conference*, 1988, pp. 37-44.

[13] CERTCC-KR, Korea Information Security Agency, http://www.certcc.or.kr/ (in Korean).

[14] Axelsson, S., 1999. *Research in intrusion-detection systems: A survey.* Chalmers University of Technology, 1999.

[15] Kuperman, B.A. and Spafford, E.H., 1998. Generation of application level audit data via library interposition. *CERIAS TR 99-11*, COAST Laboratory, Purdue University, West Lafaytte, IN, USA, 1998.

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

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