Robustness of the Markov-Chain Model for Cyber-Attack Detection

Nong Ye, *Senior Member, IEEE*, Yebin Zhang, and Connie M. Borror

***Abstract*—Cyber-attack detection is used to identify cyber-attacks while they are acting on a computer and network system to compromise the security (e.g., availability, integrity, and confidentiality) of the system. This paper presents a cyber-attack detection technique through anomaly-detection, and discusses the robustness of the modeling technique employed. In this technique, a Markov-chain model represents a profile of computer-event transitions in a normal/usual operating condition of a computer and network system (a norm profile). The Markov-chain model of the norm profile is generated from historic data of the system's normal activities. The observed activities of the system are analyzed to infer the probability that the Markov-chain model of the norm profile supports the observed activities. The lower probability the observed activities receive from the Markov-chain model of the norm profile, the more likely the observed activities are anomalies resulting from cyber-attacks, and vice versa. This paper presents the learning and inference algorithms of this anomaly-detection technique based on the Markov-chain model of a norm profile, and examines its performance using the audit data of UNIX-based host machines with the Solaris operating system. The robustness of the Markov-chain model for cyber-attack detection is presented through discussions & applications. To apply the Markov-chain technique and other stochastic process techniques to model the sequential ordering of events, the quality of activity-data plays an important role in the performance of intrusion detection. The Markov-chain technique is not robust to noise in the data (the mixture level of normal activities and intrusive activities). The Markov-chain technique produces desirable performance only at a low noise level.**

This study also shows that the performance of the Markov-chain techniques is not always robust to the window size: as the window size increases, the amount of noise in the window also generally increases. Overall, this study provides some support for the idea that the Markov-chain technique might not be as robust as the other intrusion-detection methods such as the chi-square distance test technique [35], although it can produce better performance than the chi-square distance test technique when the noise level of the data is low, such as the Mill & Pascal data in this study.

***Index Terms*—Computer audit data, computer security, intrusion detection, Markov-chain.**

Manuscript received June 16, 2001; revised July 19, 2002. This work was supported in part by the U.S. Air Force Research Laboratory—Rome (AFRL-Rome) under agreement number F30602-98-2-0005, and the U.S. Air Force Office of Scientific Research (AFOSR) under Grant Number F49620-98-1-0257. Responsible editor: N. Ye.

N. Ye is with the Information and Systems Assurance Laboratory, Tempe AZ 85287 USA (e-mail: NongYe@asu.edu).

Y. Zhang and C. M. Borror are with the Department of Industrial Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: yebinzhang@asu.edu, conni@asu.edu).

Digital Object Identifier 10.1109/TR.2004.823851

ACRONYM[1]

AFOSR	US Air Force Office of Scientific Research
AFRL-Rome	US Air Force Research Laboratory—Rome
BSM	basic security module
ISA	Information & Systems Assurance Lab. (Arizona State U.)
MC	Markov chain
MIT	Massachusetts Institute of Technology
norm	normal (usual)
OS	operating system
ROC	Receiver Operating Characteristic
s	statistical(ly)
SPARC	a specific model of Sun workstations
UNIX	a kind of OS for computers such as Sun workstations

NOTATION

E	an event
$N_{i,j}$	number of observation pairs, X_t & X_{t+1}, with X_t in state i, and X_{t+1} in state j
$N_{i.}$	number of observation pairs, X_t & X_{t+1}, with X_t in state i and X_{t+1} in any one of the states $1, \ldots, s$
N_i	number of X_t in state i
N	total number of observations
$p_{i,j}$	$\Pr\{\text{system is in a state } j \text{ at time } t+1 \mid \text{the system is in state } i \text{ at time } t\}$
q_i	$\Pr\{\text{system is in state } i \text{at time } 0\}$
X_t	observation of system state at time t

I. INTRODUCTION

A CYBER-ATTACK is launched through a series of computer actions to compromise the security (e.g., availability, integrity, and confidentiality) of a computer and network system. As there is increasing reliance on computer and network systems to support critical operations in defense, banking, telecommunication, transportation, electric power, *et al.*, cyber-attacks have become an important threat to our society with potentially severe consequences [1]–[10]. Cyber-attack detection aims at identifying cyber-attacks while they are acting on a computer and network system.

There are two general approaches to detecting cyber-attacks:

- pattern recognition, and
- anomaly detection,

which complement each other in cyber-attack detection [11]–[46].

[1]The singular and plural of an acronym are always spelled the same.

Pattern-recognition techniques identify & store signature patterns of known attacks. They then match the subject's observed behavior with those known patterns of attack signatures, and signal attacks when there is a match. Pattern-recognition techniques have been used in many commercial software & research prototypes [11], [16]–[26]. Attack signatures have been characterized as strings, event sequences, activity graphs, and attack scenarios (consisting of event sequences, their preconditions, and target compromised states). Rules [16]–[22], state transition diagrams [23], [24], colored Petri-net models [25], and decision trees [26] have been used to represent & match patterns of attack signatures. Although pattern recognition techniques are efficient in detecting known attacks, they cannot detect novel or unusual attacks whose signature patterns are unknown. Moreover, the repository of attack signature patterns must be constantly updated to remain useful in a dynamically changing system environment, e.g., configurations, protocols, architectures, and attack scenarios.

For a subject (e.g., a user, file, privileged program, host machine, or network) of interest, the anomaly-detection techniques, consisting of establishing a norm profile (a profile of the subject's normal activities), observe the activities of the subject and signal attacks when the subject's observed activities differ appreciably from its norm profile [27]–[46]. Anomaly-detection techniques consider the deviations of a subject's observed activities from its norm profile, i.e., anomalies as symptoms of attacks. Denning [30] justifies the anomaly-detection approach to cyber-attack detection. Anomaly-detection techniques can detect both novel and known attacks if they demonstrate important differences from the norm profile. Because anomaly-detection techniques treat all anomalies as attacks, false alarms are anticipated when anomalies result from irregular behavior instead of cyber-attacks. Pattern recognition techniques & anomaly-detection techniques can improve detection capabilities when used together.

This paper presents the work on the anomaly-detection approach to cyber-attack detection and concentrates on the robustness of a MC technique. Several anomaly-detection techniques exist, and differ in the representation of a norm profile and the inference of a deviation from a norm profile.

Specification-based anomaly-detection techniques [27]–[29] describe security policies, and authorized activities of a well-defined subject such as a privileged program or a network server in terms of formal logic, rules, and/or graphs. However, it is often difficult to enumerate and specify all possible normal activities or security policies of a subject, especially a large-scale subject such as a host machine or a network where interactions of multiple objects & actions are inevitable. Moreover, activities of many subjects in a computer & network system (e.g., users & files) involve uncertainties and variations which cannot be well addressed by formal logic, rules, and/or graphs.

Statistical-based anomaly-detection techniques [30]–[36] construct a statistical profile (viz., a univariate or multivariate statistical distribution) of a subject's normal activities from historic data. These statistical profiles do not consider the order in which the events occur to the subject. Although these statistical-based anomaly-detection techniques have demonstrated promising performance for cyber-attack detection with respect to detection rate and false alarm rate, it is not clear whether further improvement in attack-detection performance can be obtained by considering the ordering of events (event transitions). Reference [26] shows that attack detection using a number of consecutive events during a given period produces better performance than using a single event at a given time. Because many cyber-attacks require a series of related events to accomplish, it is possible to improve attack detection performance by incorporating the ordering of events.

There exist several anomaly-detection techniques which investigate the ordering of events for cyber-attack detection [38]–[47]. Both recurrent and perceptron neural networks have been used to predict the next event (e.g., command or system call) from a series of the past events [37], [38]. Immunology-based anomaly-detection techniques [38]–[41] store a large set of event sequences as the norm profile from historic data of a subject's normal behavior, and use either negative selection or positive selection algorithms to detect incoming event sequences which differ from event sequences in the norm profile. Anomaly-detection techniques based on Bayes' parameter estimation for building a Markov model of a norm profile & likelihood ratio tests for detecting anomalies [42]–[45], Bayesian networks [46], hidden Markov models [41], or principal components analysis [47], are also investigated. However, the training of all these techniques, taking into account the ordering of events, is computationally intensive [41], [44]. The inference of anomalies in many of the above techniques [41], [38]–[45], [46] is also computationally intensive. Immunology-based anomaly-detection techniques require large amounts of memory to store a large set of event sequences. The anomaly-detection techniques based on the Markov model [42]–[45] are computationally intensive due to their use of the Bayes' parameter estimation for learning the norm profile & a likelihood ratio test for inferring anomalies. Considering large amounts & high frequency of events in a computer and network system, those techniques are not applicable to cyber-attack detection in real time.

Reference [48] presents a computationally efficient method of applying the MC model to computer intrusion detection. This paper presents the further study on this method, specifically on the robustness of the MC technique to noise as it is introduced into the system. Section II describes the MC model. Section III discusses the representation of the attack-detection problem using the MC model & two applications of the technique. Section IV presents the results of the applications along with a discussion of these results.

II. Markov-Chain Model

A discrete-time stochastic process can be described as an arbitrary family of random variables $X(t, \xi, \theta)$, where t is a series of discrete time points,

$$t_0 < t_1 < \cdots < t_i < \cdots < t_n \in T$$

and T is a parametric space. Here, ξ represents all the random factors of the system, and design parameter θ belongs to an infinite set of all possible design-parameters. The set of all possible values or states assumed by the process, S, is called the

state space. A realization of the discrete-time stochastic process can be described by plotting $X(t)$ against discrete time $t = t_0, t_1, \ldots, t_i, \ldots, t_n \in T$ when $\theta \in \Theta$ is assigned a value.

An MC is a special type of discrete-time stochastic process with the following assumption [49]–[52]:

$$\Pr\{X_{t+1} = i_{t+1} \,|\, X_t = i_t, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0\} = \Pr\{X_{t+1} = i_{t+1} \,|\, X_t = i_t\}, \quad (1)$$

for any $t \in T$ and $i_t \in S$. That is, the probability distribution of the state at time $t+1$ depends on the state at time t, and does not depend on the previous states leading to the state at time t. Therefore, an MC model considers only 1-step transition probabilities. If it is assumed that a state transition from time t to time $t+1$ is s-independent of time, the MC becomes a stationary MC [24] and is denoted

$$\Pr\{X_{t+1} = i_{t+1} \,|\, X_t = i_t\} = \Pr\{X_{t+1} = j \,|\, X_t = i\} = p_{i,j}, \quad \text{for all } i \text{ and all states} \quad (2)$$

$p_{i,j} \equiv \Pr\{\text{the system is in a state } j \text{ at time } t+1 \,|\, \text{the system is in state } i \text{ at time } t\}$. If the system has a finite number of states $(1, 2, \ldots, s)$, then the MC model can be defined by the transition probability matrix [49]–[52]:

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,s} \\ p_{1,1} & p_{1,2} & \cdots & p_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ p_{s,1} & p_{s,2} & \cdots & p_{s,s} \end{bmatrix}, \quad (3)$$

$$\sum_{j=1}^{s} p_{i,j} = 1. \quad (4)$$

The initial probability distribution [49]–[52] is represented by

$$Q = [q_1, q_2, \ldots, q_s], \quad (5)$$

$q_i \equiv \Pr\{\text{the system is in state } i \text{ at time } 0\}$.
The probability that a sequence of states $\mathrm{X}_1, \ldots, \mathrm{X}_T$ occurs in the context of the MC model is computed using

$$\Pr\{X_1, \ldots, X_T\} = q_{x_1} \cdot \prod_{t=2}^{T} p_{x_{t-1}, x_t} \quad (6)$$

A logarithmic transformation of (6) can be expressed as

$$\log(\Pr\{X_1, \ldots, X_T\}) = \log(q_{x_1}) + \sum_{t=2}^{T} \log(p_{x_{t-1}, x_t}). \quad (7)$$

Because this paper considers a very large sequence of events, and the logarithm of the probabilities are considered, the central limit theorem can be applied. According to the central limit theorem, the result values from (7) conform to the s-normal distribution which is easy to set the control limits for intrusion detection.

The transition probability matrix and the initial probability distribution of a MC model can be obtained from the past observations of the system state. Given the observations of the system state,

$$X_0, X_1, X_2, \ldots, X_{N-1}$$

one can estimate the transition probabilities and the initial probabilities using [53], [54]:

$$p_{i,j} = \frac{N_{i,j}}{N_{i.}}, \quad (8)$$

$$q_i = \frac{N_i}{N}; \quad (9)$$

- $N_{i,j} \equiv$ number of observation pairs, X_t & X_{t+1}, with X_t in state i, and X_{t+1} in state j;
- $N_{i.} \equiv$ number of observation pairs, X_t & X_{t+1}, with X_t in state i and X_{t+1} in any one of the states $1, \ldots, s$;
- $N_i \equiv$ number of X_t's in state i;
- $N \equiv$ total number of observations.

Bayes parameter estimation can also be used to estimate the transition probability matrix and the initial probability distribution from historic data. However, Bayes parameter estimation is not used in this study due to its high computational cost. Provided with sufficient historic data, the estimation of the transition probability matrix and the initial probability distribution via (8) & (9) can be quite stable.

III. Applications

This section defines the attack detection problem, including the data source and problem representation, by way of an example. The robustness properties of the MC technique are further illustrated through a second application.

A. Application #1

A computer & network system within an organization typically includes some host machines (e.g., machines running a UNIX OS and machines running the Windows NT OS) and communication links connecting those host machines. Currently two sources of data have been widely used to capture activities in a computer & network system for attack detection:

- network traffic data, and
- audit-trail data (audit data).

Network-traffic data contain data packets traveling over communication links between host machines, and thus capture activities over communication networks. Audit data capture activities occurring on a host machine. This study used audit data from a UNIX-based host machine (a Sun SPARC 10 workstation with the Solaris OS), and focused on attacks on a host machine which left trails in the audit data.

The Solaris OS from Sun Microsystems Inc. has a security extension called the BSM. The BSM extension supports the monitoring of activities on a host by recording security-relevant events. A BSM auditable event falls into one of two categories,

- kernel events, or
- user-level events.

Kernel events are generated by system calls to the kernel of the Solaris OS while user-level events are generated by application software.

There are more than 250 different types of BSM auditable events, depending on the version of the Solaris OS. There are also about 284 different types of BSM audit events on the host machine used in this study. A BSM audit record for each event contains a variety of information, including the event type, user ID, group ID, and process ID. This study extracts and uses only the *event type*, one of the most critical characteristics of an audit event. Therefore, activities on a host machine are captured through a continuous stream of audit events, each characterized by the event type.

By considering computer audit data with only event type, attacks on a host machine are detected. When an attack is detected from a series of audit events, the original audit records can be traced for the series of audit events and thus obtain more information such as the user(s) and process(es) from the original audit records to assist responses to an attack.

Both normal and attack activities on a host machine contain sequences of computer actions with uncertainty. Sequences of computer actions induce sequences of audit events also with uncertainty. Considering the 284 types of audit events as the 284 possible states of a host machine, an event sequence on the host machine can be represented as a discrete-time stochastic process. Discrete points in time for the discrete-time stochastic process are defined not by a fixed time interval but by times when audit events take place, because this study is mainly interested in event transitions. In another study [55], a technique was developed that examines the time interval or the event intensity for a given period of time from computer audit data. For attack detection, one should build a long-term norm profile of event sequence, and compare the event sequence in the recent past to the long-term norm profile for detecting a large difference. Using (8) & (9), a MC model of event sequence is trained & built as the long-term norm profile by learning the transition probability matrix & the initial probability distribution from a stream of audit events observed during the normal usage of the host machine.

The event sequence in the recent past is defined by opening up an observation window of size N on the continuous stream of audit events to view the last N audit events from the current time t, $E_{t-N+1}, \ldots, E_t$. At the next time $t+1$, the observation window contains $E_{t-N+2}, \ldots, E_{t+1}$. $N = 100$ and $N = 10$ were investigated to determine the effect of the window size.

For the audit events, $E_{t-N+1}, \ldots, E_t$, in the window at time t, the type of each audit event was examined, and the sequence of states $\mathrm{X}_{t-(N-1)}, \ldots, \mathrm{X}_t$ appearing in the window was obtained.

- X_i: the state (type of audit event) that the audit event E_i takes.

Using (7), compute the probability that the sequence of states $X_{t-N+1}, \ldots, X_t$, occurs in the context of the normal usage, viz., the probability that the MC model of the norm profile supports the sequence of states $X_{t-N+1}, \ldots, X_t$:

$$\log(\Pr\{X_{t-N+1}, \ldots, X_t\}) = \log\left(q_{X_{i-N+1}} \cdot \prod_{i=t-N+2}^{t-1} p_{x_i, x_{i+1}}\right). \quad (10)$$

The larger the probability, the more likely the sequence of states results from normal activities. A sequence of states from attack activities is presumed to receive a low probability of support from the MC model of the norm profile.

It is possible that a sequence of states from a window of the testing data presents an initial state and/or some state transitions which are not encountered in the training, and thus have the probabilities of zero in the initial probability distribution, or in the transition probability matrix of the MC model. While using (10) to infer the probability of support to the sequence of states, the zero probabilities would dominate the final probability-result from (10) and make it zero. This would make it impossible to distinguish attack activities from normal activities, because normal activities could introduce a new initial state and/or new state-transitions. The amount of the new initial state and/or new state-transitions from normal activities is anticipated to be much less than the amount of the new initial state and/or new state transitions from attack activities.

If the host machine is observed for a very long time, there is the opportunity to observe every possible initial state and state transition. However, not every possibility appeared in the training data (of this project) due to the limited sample size of the data. For example, if the true probability of an initial state is 10^{-5}, and the sample size of the training data is 1613, the anticipated number of times that this initial state appears in the training data becomes $1613 \cdot 10^{-5} = 0.016\,13$. That is, we should not anticipate observing this initial state in the training data, although the true probability of this initial state is not zero.

Therefore, while using (10) to infer the probability of support to a sequence of states, the probability of zero for a new initial state or a new state transition is replaced by a small probability value. By assigning a small probability rather than zero to the new initial state or the new state transition, the small effects of new states and new state transitions in normal activities on the final probability result of (10) becomes distinguishable from the larger effect of attack activities on the final probability result of (10).

In this study, the sample size of the training data (to be presented in Section IV) is 1613. Because $1/1613 = 0.0006$, any initial state or state transition with a true probability less than 0.0006 is not anticipated to appear in the training data. Therefore, any small probability values less than 0.0006 can be chosen. A smaller probability value is better at magnifying the difference in the amount of the new initial state and new state transitions in attack activities than those resulting from noises in normal activities. This study assigns the probability value of 10^{-5} to an initial state or a state transition, which does not appear in the training data, while using (10) to infer the probability of support to a sequence of states. This replacement of zero with a small probability occurs during the inference and after the estimation of the MC model from the training data is complete.

The learning & inference of the MC model for attack detection are implemented using C++. In the C++ program used here, the data type "double" is used for the variable that keeps the probability of support to the sequence of states from a window. A variable of the data type "double" takes 64 bits of memory with the value range $[1.0\mathrm{E}-323, 1.0\mathrm{E}+323]$ in scientific ex-

pression. For probability values smaller than $1.0\mathrm{E}-323$, the variable is assigned the value of zero. The results & discussion of this application are provided in Section IV.

Audit data of normal activities are required for estimating a MC model of the norm profile. A sample of audit data is used for normal activities from the MIT Lincoln Lab, containing a stream of 3019 audit events [12] for this study (http://ideval.ll.mit.edu/). Computer audit data for normal activities from the MIT Lincoln Lab are generated by simulating activities in a real computer & network system. This sample of audit data for normal activities, consists of 2316 audit events, and is divided into 2 parts; one part consisting of 1613 audit events, and the other part consisting of the remaining 703 audit events. The part consisting of 1613 audit events is used for training an MC model of the norm profile using (8) & (9). The other part of the audit data, consisting of 703 audit events, is used for testing.

For testing, audit data of attack activities are generated in the ISA lab by simulating 15 attack scenarios, in a random order, which have been collected over several years from various information sources. Two examples of the attack scenarios are:

- password-guessing, and
- attempts to gain an unauthorized access remotely.

The attack scenarios are manually run on the host machine while the auditing facility is turned on, resulting in a stream of 1225 audit events. As a result, the training data consist of the 1613 audit events for normal activities from the MIT Lincoln Lab., and the testing data consisting of the 703 audit events for normal activities from the MIT Lincoln Lab. and the 1225 audit events from attack activities from the simulation in the ISA lab. Although there are two sources for obtaining data—MIT Lincoln Lab. and the ISA lab—the same OS is used at both labs. The characteristic under study, event type, is also the same for both sources.

Next, an MC model of the norm profile is obtained using the training data which contain the 1613 audit events for normal activities. There are 284 possible types of audit events, of which only 11 types of audit events will appear in the training data. With 11 types of audit events, there are at most 132 parameters (equal to 121 parameters for the transition probability matrix$+11$ parameters for the initial probability distribution) in the MC model to estimate from the 1613 data points. Moreover, not all the 121 possible state transitions appear in the training data. The 1613 data points are sufficient to estimate the parameters in the MC model in this situation.

For testing, the 1225 audit events $(t = 0, 1, \ldots, 1224)$ from attack activities & the 703 audit events $(t = 0, 1, \ldots, 702)$ from normal activities, are viewed through a moving window of 100 events, creating

- 1126 windows ($t = 99, 100, \ldots, 1224$ or window # 1–1126) for attack activities, and
- 604 windows ($t = 99, 100, \ldots, 702$ or window # 1–604) for normal activities.

Each of the first 99 audit events for either attack activities or normal activities do not create a window, because the event and preceding events together do not provide 100 audit events for a complete window. The robustness of the MC technique is examined for $N = 10$ & 100, and the results are compared & contrasted.

Intrusions usually occur in a computer & network system while normal activities are also occurring in that system. In real time, intrusive audit data (signals to detect) are mixed with normal audit data (noises to ignore). Thus, in order to investigate the robustness of the MC technique or, how the performance of this technique depends on the quality of audit data, four sets of testing dataset were created using the 1225 audit events from attack activities and the 703 audit events from normal activities described early in this section. Each subsequent dataset has an increase in mixture of attack audit events with the normal events (i.e., the noise level increases from one dataset to the next).

For testing dataset #1 (Pure), the 1225 attack audit events are put after the 703 normal audit events. Obviously, no mixing is done for this set of data. For testing dataset #2, the attack and normal audit events are mixed by dividing the 1225 abnormal audit events evenly into 2 parts, $D_{a,1}$ & $D_{a,2}$, and also by dividing the 703 normal audit events evenly into 2 parts, $D_{n,1}$ & $D_{n,2}$. The parts are arranged as follows:

$$D_{n,1} \quad D_{a,1} \quad D_{n,2} \quad D_{a,2}.$$

Dataset #2 is referred to as the small noise level (SNL) dataset. For testing dataset #3, the 1225 attack audit events are divided into 7 parts by 7 attack sessions in which these 1225 attack events fall—not necessarily of the same size, $D_{a,1}, D_{a,2}, D_{a,3}, D_{a,4}, D_{a,5}, D_{a,6}, D_{a,7}$; while the 703 normal audit events are also divided into 7 parts evenly, $D_{n,1}, D_{n,2}, D_{n,3}, D_{n,4}, D_{n,5}, D_{n,6}, D_{n,7}$. Then all these parts are arranged as: $D_{n,1}, D_{a,1}, D_{n,2}, D_{a,2}, D_{n,3}, D_{a,3}, D_{n,4} \quad D_{a,4}, D_{n,5}, D_{a,5}, D_{n,6}, D_{a,6}, D_{n,7}, D_{a,7}$

Dataset 3 is referred to as the large noise level (LNL) dataset. For the testing dataset 4 (Random), the 1225 abnormal audit events and 703 normal audit events were mixed randomly. From a window at time t, a sequence of states $X_{t-N+1}, \ldots, X_t$, is obtained & evaluated against the MC model of the norm profile, to yield the probability that the sequence of states is supported by this MC model. A high probability indicates that the sequence of states more likely result from normal activities. The lower the obtained probability, the less likely the sequence of states is a result of normal activities.

B. Application #2

This application is presented to demonstrate further the MC technique using a larger dataset. This situation has 2 large datasets (a Mill dataset and a Pascal dataset) which were obtained from the MIT Lincoln Laboratory (http://ideval.ll.mit.edu/). Mill & Pascal are the names of the host machines of the network constructed by the MIT Lincoln Laboratory in 2000 to simulate the environment of the network in the real world, and thus provide a test bed of comprehensive evaluations for various intrusion detection systems. These Mill & Pascal audit datasets are collected from these host machines. The OS are Solaris 2.5.1 and Solaris 2.7, respectively. The Mill dataset includes 15 normal sessions consisting of 68 872 audit events & 7 attack sessions. The Pascal dataset includes 63

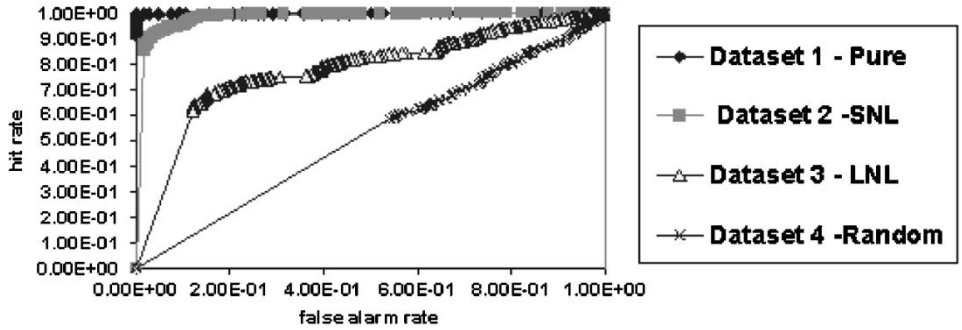

Fig. 1. ROC curves of the MC technique for 4 small datasets, $N = 100$.

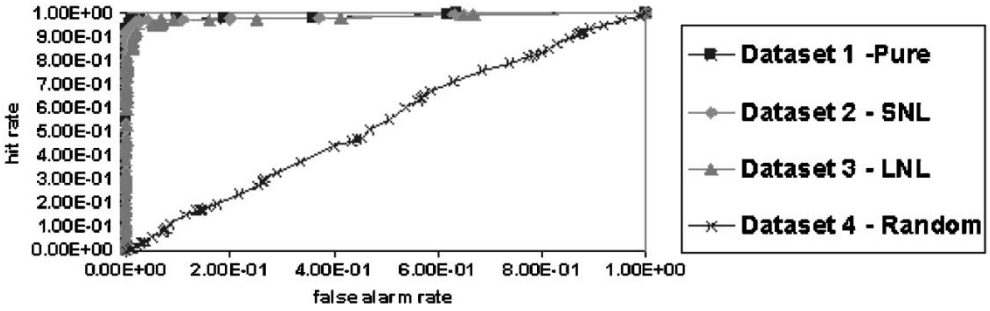

Fig. 2. ROC curves of the MC technique for 4 small datasets, $N = 10$.

normal sessions consisting of 81 756 audit events & 4 attack sessions.

For this MC technique, use the data collected in the last hour from each data set (Mill or Pascal), and use only the normal audit data, as the training dataset. The dataset collected in all 3 hours from the same machine was used as the testing dataset. Section IV provides the results & discussion of this application.

IV. Results & Discussion

A. ROC Curves

For a given testing data set, obtain the support probability value for each audit event in the testing data set from (10). Set a signal threshold such that an event is signaled as intrusive if its support probability is less than the signal threshold, and an event is considered as normal if its support probability is not less than the signal threshold. A false alarm would occur when normal activities were signaled as attack activities, and a hit would occur when a signal is observed and the event was an attack activity. The false-alarm rate is given by: (number of false alarms)/(total number of normal activities). Obviously, the false-alarm rate should be small. For example, assume there are 703 normal audit events investigated and 24 of these events signaled as attack activities. Thus the false-alarm rate is $24/703 = 0.0341$. Obviously, it is desirable to have the false-alarm rate be as small as possible. The hit rate, on the other hand, is given by (number of hits)/(total number of attack activities). In this situation, one prefers that the hit rate be as large as possible.

Hence, for a given signal threshold, obtain a pair of 'false-alarm rate' & 'hit rate' from the probability values for the audit events in the testing data set. Different signal thresholds lead to different pairs of 'false-alarm rate' & 'hit rate' which describe the performance of the detection technique according to signal detection theory [56]. A ROC curve plots pairs of 'false-alarm rate' & 'hit rate' as points when various signal thresholds are used.

To illustrate the usefulness of ROC curves in general, consider Fig. 1. The closer the ROC curve is to the upper-left corner (representing 100% hit rate & 0% false-alarm rate), the better the performance of the detection technique.

B. Application #1

Figs. 1 & 2 present the ROC curves for window sizes of $N = 100$ & $N = 10$, respectively. The results for each of the 4 small datasets described in Section IV-A are plotted for these window sizes.

From the ROC curves in Figs. 1 & 2, it is obvious that the detection performance of the MC technique depends strongly on the quality of audit data; i.e., as the noise level increases, the performance of the MC decreases. Also, the MC technique with $(N = 10)$ outperforms the MC technique with $N = 100$ as the amount of noise in the window increases.

C. Application #2

Fig. 3 shows the ROC curves for the Mill dataset. Fig. 4 shows the ROC curves for the Pascal dataset. These ROC curves show that the MC technique performs very well on both the Mill and Pascal datasets. The results for Application #1 demonstrate that the performance of the MC technique degrades as the noise level increases in the testing data set. The good performance of the MC technique might indicate a low noise level or the "clean" nature of the Mill & Pascal datasets.

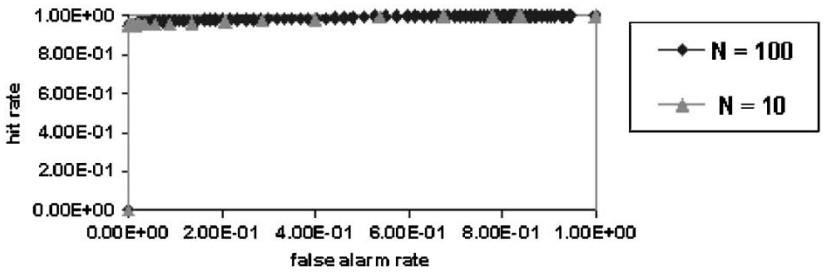

Fig. 3. ROC curves for Mill dataset.

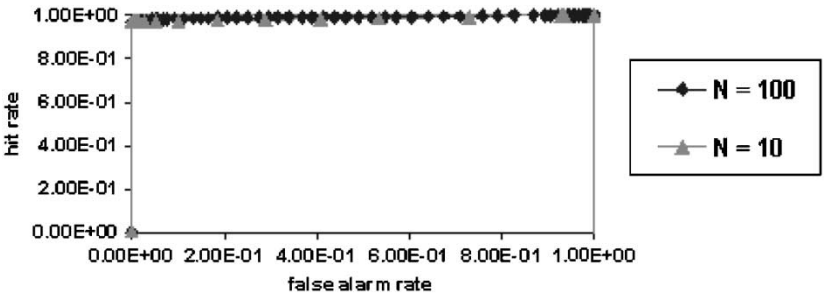

Fig. 4. ROC curves for Pascal dataset.

ACKNOWLEDGMENT

The U.S. government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright annotation thereon. The views & conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFRL-Rome, AFOSR, or the US Government.

REFERENCES

[1] W. Stallings, *Network and Inter-network Security Principles and Practice*: Prentice Hall, 1995.
[2] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*: Prentice Hall, 1995.
[3] T. Escamilla, *Intrusion Detection: Network Security Beyond the Firewall*: John Wiley & Sons, 1998.
[4] B. Simons, "Building big brother," *Commun. ACM*, vol. 43, no. 1, pp. 31–32, Jan. 2000.
[5] P. G. Neumann, "Risks of insiders," *Commun. ACM*, vol. 42, no. 12, p. 160, Dec. 1999.
[6] M. Godwin, "Net to worry," *Commun. ACM*, vol. 42, no. 12, pp. 15–17, Dec. 1999.
[7] S. Jajodia, P. Ammann, and C. D. McCollum, "Surviving information warfare attacks," *Computer*, vol. 32, no. 3, pp. 57–63, Apr. 1999.
[8] P. Mann, "Pentagon confronts mounting cyber risks," *Aviation Week and Space Technology*, vol. 150, no. 12, pp. 82–83, Mar. 1999.
[9] B. H. Barnes, "Computer security research: A British perspective," *IEEE Software*, vol. 15, no. 4, pp. 30–33, Sept./Oct., 1998.
[10] A. Boulanger, "Catapults and grappling hooks: The tools and techniques of information warfare," *IBM Systems J*, vol. 37, no. 1, pp. 106–114, 1998.
[11] H. Debar, M. Dacier, and A. Wespi, "Toward a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, pp. 805–822, 1999.
[12] R. Lippmann *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proc. DARPA Information Survivability Conf. & Exposition*, Jan. 2000, pp. 12–26.
[13] D. Schnackenberg and K. Djahandari, "Infrastructure for intrusion detection and response," in *Proc. DARPA Information Survivability Conf. & and Exposition*, Jan. 2000, pp. 3–11.
[14] T. Bass, "Intrusion detection systems and multi-sensor data fusion," *Commun. ACM*, vol. 43, no. 3, pp. 99–105, Apr. 2000.
[15] M. Stillerman, C. Marceau, and M. Stillman, "Intrusion detection for distributed applications," *Commun. ACM*, vol. 42, no. 6, pp. 62–69, July 1999.
[16] U. Lindqvist and P. A. Porras, "Detecting computer and network misuse through the production-based expert system toolset (P-BEST)," in *Proc. 1999 IEEE Symp. Security and Privacy*, May 1999.
[17] P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," in *Proc. NISSC*, Oct. 1997.
[18] P. G. Neumann and P. A. Porras, "Experience with EMERALD to date," in *Proc. USENIX Workshop on Intrusion Detection and Network Monitoring*, Apr. 1999, pp. 73–80.
[19] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. 1998 USENIX Security Symp*.
[20] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Proc. 1999 IEEE Symp. Security & Privacy*.
[21] ——, "Mining audit data to build intrusion detection models," in *Proc. 1998 Int.l Conf. Knowledge Discovery and Data Mining*.
[22] ——, "Mining in a data-flow environment: Experience in network intrusion detection," in *Proc. 1999 ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*.
[23] G. Vigna and R. Kemmerer. NetStat: A network-based intrusion detection appoach. presented at Proc. 1998 Ann. Computer Security Applications Conf. [Online]. Available: http://www.cs.ucsb.edu/~kemm/netstat.html/
[24] G. Vigna, S. T. Eckmann, and R. A. Kemmerer, "The STAT tool suit," in *Proc. 2000 DARPA Information Survivability Conf. & Exposition*, pp. 46–55.
[25] S. Kumar, "Classification and Detection of Computer Intrusions," Ph.D. Dissertation, Dept. Computer Science, Purdue University, 1995.
[26] N. Ye, X. Li, and S. M. Emran, "Decision trees for signature recognition and state classification," in *Proc. 2000 IEEE SMC Information Assurance and Security Workshop*.
[27] C. Ko, G. Fink, and K. Levitt, "Execution monitoring of security-critical programs in distributed systems: A specification-based approach," in *Proc. 1997 IEEE Symp. Security and Privacy*, pp. 134–144.
[28] S. Staniford-Chen *et al.*, "GrIDS—A graph-based intrusion detection system for large networks," in *Proc. 1996 Nat. Information Systems Security Conf.*.
[29] T. Bowen *et al.*, "Building survivable systems: An integrated approach based on intrusion detection and damage containment," in *Proc. 2000 DARPA Information Survivability Conf. Exposition*, vol. II, pp. 84–99.
[30] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Software Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[31] D. Anderson, T. Frivold, and A. Valdes, "Next-Generation Intrusion Detection Expert System (NIDES): A Summary,", Menlo Park, CA, Technical Report SRI-CSL-97-07, May 1995. SRI Int'l.
[32] H. S. Javitz and A. Valdes, "The SRI statistical anomaly detector," in *Proc. 1991 IEEE Symp. Research in Security and Privacy*.
[33] ——, "The NIDES Statistical Component Description of Justification,", Menlo Park, CA, Technical Report A010, Mar. 1994. SRI Int'l.
[34] Y. Jou *et al.*, "Design and implementation of a scalable intrusion detection system for the protection of network infrastructure," in *Proc. 2000 DARPA Information Survivability Conf. Exposition*, pp. 69–83.
[35] N. Ye, Q. Chen, and S. M. Emran, "Chi-square statistical profiling for anomaly detection," in *Proc. 2000 IEEE SMC Information Assurance and Security Workshop*.
[36] ——, "Hotelling's T2 multivariate profiling for anomaly detection," in *Proc. 2000 IEEE SMC Information Assurance and Security Workshop*.
[37] H. Debar, M. Becker, and D. Siboni, "A neural network component for an intrusion detection system," in *Proc. 1992 IEEE Computer Society Symp. Research in Security and Privacy*, pp. 240–250.
[38] A. K. Ghosh, A. Schwatzbard, and M. Shatz. Learning program behavior profiles for intrusion detection. presented at Proc. 1999 USENIX Workshop on Intrusion Detection and Network Monitoring. [Online]. Available: http://www.rstcorp.com/~anup/
[39] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Commun. ACM*, vol. 40, no. 9, pp. 88–96, Oct., 1997.
[40] H. Debar, M. Dacier, M. Nassehi, and A. Wespi, "Fixed vs. variable-length patterns for detecting suspicious process behavior," in *1998 Proc. European Symp. Research in Computer Security*, pp. 1–15.
[41] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proc. 1999 IEEE Symp. Security and Privacy*, pp. 133–145.
[42] W. DuMouchel. Computer Intrusion Detection Based on Bayes factors for Comparing Command Transition Probabilities. National Institute of Statistical Sciences. [Online]. Available: http://www.niss.org/downloadabletechreports.html
[43] W.-H. Ju and Y. Vardi. A Hybrid High-Order Markov-Chain Model for Computer Intrusion Detection. National Institute of Statistical Sciences. [Online]. Available: http://www.niss.org/downloadabletechreports.html
[44] M. Schonlau *et al.*. Computer Intrusion: Detecting Masquerades. National Inst. of Statistical Sciences. [Online]. Available: http://www.niss.org/downloadabletechreports.html
[45] S. L. Scott, Detecting Network Intrusion Using a Markov Modulated Nonhomogeneous Poisson Process.
[46] N. Ye, Q. Zhong, and M. Xu, "Probabilistic networks with undirected links for anomaly detection," in *Proc. 2000 IEEE SMC Information Assurance and Security Workshop*.
[47] W. DuMouchel and M. Schonlau, "A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities," in *Proc. #30 Symp. Interface: Computing Science and Statistics*.
[48] N. Ye, Cyber attack detection through a Markov-chain model of computer event transitions, in *IEEE Trans. Systems, Man and Cybernetics*. submitted.
[49] W. L. Winston, *Operations Research: Applications and Algorithms*: Duxbury Press, 1994.
[50] P. Buttorp, *Stochastic Modeling of Scientific Data*: Chapman & Hall, 1995.
[51] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*: Chapman & Hall, 1995.
[52] I. L. MacDonald and W. Zucchini, *Hidden Markov and Other Models for Discrete Valued Time Series*: Chapman & Hall, 1997.
[53] T. M. Mitchell, *Machine Learning*: McGraw-Hill, 1997.
[54] F. V. Jensen, *An Introduction to Bayesian Networks*: UCL Press, 1996.
[55] N. Ye and Q. Chen, "EWMA techniques for detecting computer intrusions through anomalous changes in event intensity," *IEEE Trans. Rel.*, vol. 52, no. 1, Mar. 2004.
[56] B. H. Kantowitz and R. D. Sorkin, *Human Factors: Understanding People-System Relationships*: John Wiley & Sons, 1983.

Nong Ye is a Professor of Industrial Engineering and an Affiliated Professor of Computer Science and Engineering at Arizona State University. She holds a Ph.D. in Industrial Engineering from Purdue University, a M.S. in Computer Science from the Chinese Academy of Sciences, and a B.S. in Computer Science from Peking University. Her research interests are in assuring process quality and preventing faults and errors in information systems. Dr. Ye is an Associate Editor for IEEE Transactions on Reliability and IEEE Transactions on Systems, Man, and Cybernetics. She is a senior member of the Institute of Industrial Engineers, and a senior member of IEEE.

Yebin Zhang received his B.S. in Automatic Control from Central South University of Technology, Changsha in 1993 and the M.S. degree in Control Engineering and Application from Nakai University, Tianjin in 1999. From 1993 to 1996, he worked as an Assistant Professor in Central South University of Technology. Currently he is a Ph.D. student in the Industrial Engineering Department of Arizona State University. His research interests are in applied statistics in intrusion detection, data mining, and stochastic optimization & simulation.

Connie M. Borror is a Senior Lecturer in the Department of Industrial Engineering at Arizona State University. She received her Doctorate in Industrial Engineering from Arizona State University with an emphasis of Quality Engineering. Her research interests include new developments in statistical process control, design of experiments, and response surface methodology. She is a member of the American Society for Quality, the American Statistical Association, the Royal Statistical Society, the Institute of Industrial Engineering, the Caucus for Women in Statistics, and the American Society for Engineering Education.