26

Mining Computer and Network Security Data

Nong Ye
Arizona State University

This chapter is based on the work performed with the funding support from the Air Force Office of Scientific Research (AFOSR) under grant number F49620-99-1-001, the Air Force Research Laboratory (AFRL), Rome, under agreement number F30602-98-2-0005, and the DARPA/AFRL-Rome under grant number F30602-99-1-0506. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFOSR, AFRL-Rome, DARPA, or the U.S. Government.

INTRODUCTION

As computer and network systems have been increasingly used to support critical operations in many fields, including defense, banking, telecommunication, transportation, E-commerce, education, and so on, intrusions into computer and network systems have increased. Intrusions compromise the security (i.e., integrity, confidentiality, and availability) of computer and network systems and have become significant threats to the quality of services from computer and network systems with severe consequences. Layers of defense can be set up against intrusions through prevention, detection, reaction, and so on.

Research on offline intrusion prevention focuses mainly on methodologies of secure software engineering (Viega & McGaw, 2002). Although secure software methodologies will continue to improve, bugs and vulnerabilities in computer and network systems are inevitable due to the difficulty in managing the complexity of such large-scale systems during their specification, design, implementation, and installation. Attackers explore bugs and vulnerabilities in systems to attack them. Online intrusion prevention techniques (Fisch & White, 2000; Pfleeger, 1997; Ye, 2003), such as firewalls for network packet filtering, cryptography for data encryption and decryption, and authorization/authentication/identification for access control, prevent intrusions by limiting access to computer and network assets (data and programs). Firewalls control access to a host machine or a network. Authentication/identification/authorization controls access to a local or remote service from a program. Cryptography controls access to data. By limiting access to various kinds of assets in computer and network systems, the online intrusion prevention techniques raise the difficulty of launching successful intrusions. However, intrusion prevention techniques cannot completely prevent intrusions, especially from determined attackers with resources and sophisticated skills. As long as access to system assets exists, vulnerabilities exist for that access to assets and can be discovered if they are unknown at present. Hence, it is expected that some intrusions can pass through online intrusion prevention mechanisms and act on computer and network systems to carry out intrusive activities. Moreover, intrusion prevention techniques can hardly prevent insider attacks.

The goal of intrusion detection is to detect intrusive activities while they are taking place on computer and network systems. System activities usually are monitored by collecting data of system activities and analyzing the data to detect intrusive activities. The detection of an intrusion then triggers intrusion reaction to assess the intrusion and damage and take actions for system recovery and further intrusion prevention.

This chapter presents applications of data mining techniques to intrusion detection, especially regarding which data are collected from computer and network systems for monitoring and detecting intrusive activities and how data mining techniques are applied to intrusion detection. The second section describes intrusive activities and data that capture system activities, including intrusive activities. The third section discusses the features extracted from system activity data and the representation of the features for intrusion detection. The fourth section reviews existing intrusion detection techniques. The fifth section presents an application of statistical anomaly detection techniques to intrusion detection. The final section summarizes the chapter.

INTRUSIVE ACTIVITIES AND SYSTEM ACTIVITY DATA

An intrusion usually includes a series of activities. Activities on computer and network systems, including intrusive activities, can be recorded for intrusion detection. Existing work on intrusion detection primarily has used data of system activities to detect intrusions, although other kinds

of data (e.g., system state and performance data) in computer and network systems also may be helpful for intrusion detection. In this chapter data of system activities for intrusion detection are described. Readers are referred to Ye (2003) for the description of other data that also may be useful for intrusion detection.

To understand the data of system activities and the detection of intrusive activities recorded in system activity data, intrusive activities are described by showing various phases of intrusions. Then two types of system activity data and how intrusive activities may manifest in these data are presented and discussed.

Phases of Intrusions

Successful intrusions often go through several phases. Different phases have different goals and attack different vulnerabilities of different system assets using different methods. There are mainly six phases of intrusions: reconnaissance, scanning, gaining access, maintaining access, launching further attacks, and covering tracks. Not every phase is used in every intrusion case. These phases are briefly described in following paragraphs. More details can be found in Skoudis (2002) and Ye (2003).

In the reconnaissance phase an attacker investigates a target computer and network system using publicly available information. Reconnaissance techniques include low-technology reconnaissance (e.g., social engineering to trick people into releasing information and physical break-in to obtain information), search of offline sources (e.g., phone books) and online sources, and so on. Examples of online sources to gain useful information about a target organization are general search engines (e.g., Google), whois databases, the domain name system (DNS), and the organization's Web site. Information obtained may include names of computer and network domains, IP (Internet protocol) addresses of computer resources within the organization, e-mail addresses of employees, and so on. Such information can be used in later phases of intrusions, e.g., the use of an IP address to construct an IP address spoofing attack for gaining access to a host machine.

In the scanning phase an attacker attempts to map the topology of a target computer and network system, find potential entry points into the system, and discover vulnerabilities that can be exploited in later intrusion phases, using information such as IP addresses and domain names obtained from the reconnaissance phase. Network mapping uses network tools such as a *traceroute* to find out the network topology of the system and active host machines. Active host machines are then accessed to find out available remote and local services such as modem dial-in and network services (e.g., Web). System responses to requests for available services on a host machine may reveal the type of the operating system running on the host machine. Then vulnerabilities of that particular operating system can be exploited in later phases of intrusions.

In the gaining-access phase an attacker exploits vulnerabilities on a target host machine to gain access to the host machine or gain higher privileges such as root user privileges. After gaining access to the host machine, the attacker may use assets (e.g., sensitive files and data) on the host machine in later phases of intrusions such as maintaining access, covering tracks, and launching further attacks. As there are a variety of system vulnerabilities, there are a variety of ways to gain access, including password guessing, buffer overflow attacks, IP spoofing attacks, Web attacks, viruses, and so on.

In the maintaining-access phase, an attacker attempts to establish an easy, safe access to a target host machine so that the attacker can come back later without going through the initial, complicated, and risky process of gaining access. Methods of maintaining access include creating a new user account, using a Trojan program to steal an existing user's password, creating a back door, and so forth.

After an attacker gains and maintains access to a target host machine, the attacker may launch further attacks on a larger scale and with a wider impact, for example, attacks on other host machines within or beyond the network domain of the compromised host machine. Typical examples of such further attacks are distributed Denial of Service (DoS) attacks that use compromised victim hosts simultaneously to attack a target computer and network system by remotely flooding the target system with network requests or network traffic. Note that there are some attacks on a target system, such as many forms of DoS attacks, that do not require access rights to the system, and therefore can be launched without going through the gaining-access and maintaining-access phases.

Host machines usually are equipped with auditing/logging facilities to record activities, including intrusive activities. If an attacker does not want to be caught, the attacker may want to alter audit/log files to remove the attacker's activity trails in these files.

Data of System Activities

Because a computer and network system has two kinds of components, host machines and network links, data can be collected to capture activities on both. Hence, there are mainly two types of system activity data: network traffic data capturing network activities and computer audit/log data capturing host machine activities.

Network Traffic Data. Because network activities involve mainly the transmission of data, network activity data are a collection of data packets being transmitted over network links. Because data packets are the traffic on a network, network activity data are also called network traffic data. A data packet consists of the following two parts (Northcutt, Cooper, Fearnow, & Frederick, 2001; Stevens, 1994; Ye, 2003):

- Data payload. The data payload is generated by a network application program, for example, an FTP (file transfer protocol) application for transferring files, an SMTF (simple mail transfer protocol) application for sending and receiving emails, and an HTTP (hypertext transfer protocol) application for Web browsing.
- Header. The header information is added by various layers of network communication protocols such as TCP (transmission control protocol) and IP for enabling and controlling the transmission of the data payload from the source IP address to the destination IP address on the Internet.

A data packet travels over network links in binary form. Special software programs often are used to capture and interpret the binary information in a data packet. These software programs are called sniffers. Tcpdump is a common sniffer. Versions of tcpdump running in Solaris and Linux environments can be downloaded at www.tcpdump.org. WinDUMP is the Windows version of tcpdump and can be downloaded at netgroup-serv.polito.it/windump. The Mac OS X operating system comes with tcpdump. The following is an example of a data line produced by tcpdump for an intercepted data packet:

```
15:11:32.812372 122.111.1.30.43590 > 122.111.0.40.www: S
768602:768629(27) ack 888916 win 4096 <mss 1024> (ttl 60, id 33916)
XXXX XXXX . . . . . .
```

where

"15:11:32.812372" is the time of receiving the data packet, in which "812372" is the data packet's unique time-stamp because multiple data packets may arrive at any given second,
"122.111.1.30" is the source IP address of the data packet,
"43590" is the source port,
">" indicates the traffic direction,
"122.111.0.40" is the destination IP address,
"www" is the destination port recognized by tcpdump from the TCP port number of 80,
"S" is a control bit used by TCP indicating the request for establishing a network session,
"768602" and "768629" are the sequence numbers indicating the range of the data payload with the length of 27 bytes,
"ack 888916" indicates the acknowledge number,
"win 4096" indicates the window size for controlling the data flow,
"<mss 1024>" shows the maximum segment size (mss) of 1024 bytes,
"ttl" indicates 60 as the time to live (TTL) value,
"id 33916" is the session identification (ID), and
"XXXX XXXX ....." is the data payload in the binary form, where "X" denotes a hexadecimal digit.

Network traffic data can be useful in detecting intrusive activities in the reconnaissance and scanning phases, for example, access to online sources (including whois databases, DNS servers, and Web sites) for reconnaissance, and access to network services (including HTTP, FTP, and SMTP applications) for scanning active hosts and open ports for network services such as HTTP. Those intrusive activities are captured in network traffic data and can be identified using information in the header and data payload of a data packet. For example, the destination port in the header of a data packet indicates the requested network service. If the destination port is associated with a network service with known vulnerabilities, it indicates a possible attempt to exploit those known vulnerabilities for an intrusion. For another example, the time of access and the destination port from the header of each data packet can tell us the intensity of network traffic for a particular network service, and thus can be used to detect DoS attacks through the traffic flooding of that network service.

Computer Audit/Log Data. A variety of data can be collected from a host machine to capture activities on the host machine. These data include mainly computer audit data, system log data, and application log data (Ye, 2003).

Computer audit data capture host activities managed by the kernel of the operating system, that is, actions or events taking place in the kernel of the operating system. Auditable events are those actions that may have security implications. The following are some examples of auditable events:

- Actions involved in authentication/identification/authorization
- Addition and deletion of objects in a user's address space
- Actions of adding or deleting user accounts by system administrators
- Use of printer, network interface card, and other I/O (input/output) devices

Auditable information recorded for each auditable event may reveal, for example:

- Time of the event
- Type of the event
- User generating the event
- Process requesting the event
- Object accessed in the event
- Source location of the request leading to the event
- Return status of the event

Hence, from computer audit data we can obtain information about access to files, users, and processes (e.g., privileged processes for authentication/identification/authorization, user account creation, and access to I/O devices).

For instance, the Sun Solaris operating system comes with the SunSHIELD Basic Security Module (BSM) that is an auditing facility in Solaris for recording auditable events. In BSM, audit events are generated by either kernel-level system calls from the kernel of the operating system or user-level system calls from processes or applications outside the kernel of the operating system. We can select from the following classes of audit events for the system to audit:

- File read events
- File write events
- File attribute-related events such as changes of file attribute (e.g., chown, clock)
- Nonattribute related events
- File modification events
- File creation events
- File deletion events
- File close events
- Process events, such as exec, fork, exit
- Network events
- IPC events
- Administrative events
- Login/logout events
- Application events
- ioctl (IO control) events
- Program execution events
- Miscellaneous events

In Solaris 2.5.1 there are 284 different types of auditable events. A complete list of audit event types in Solaris can be found at www.sun.com/docs.

In BSM each auditable event produces an audit record. Audit records are kept in audit files. Each audit record includes a number of audit tokens. Each audit token describes an attribute of the audit event. The following is an example of a BSM audit record from the DARPA (Defense Advanced Research Projects Agency) Intrusion Detection Evaluation 2000 Data generated by the MIT Lincoln Laboratory (http://ideval.ll.mit.edu). Token IDs in these audit records are underlined to highlight separate tokens.

header,128,2,sendto(2),,Tue 07 Mar 2000 10:50:19 AM EST, +834856051 msec, argument,1, 0 × 4,fd,argument,3, 0 × 9,len,argument,4, 0 × 0,flags,argument,6, 0 × 1

0,tolen,socket,0 × 0002,0 × 246d,255.255.255.255,subject,root,root,root,root,root,2
808,2806,0 0 ppp5-213.iawhk.com.return,success,9,trailer,128

This audit record starts with a header token, where

"header" is the token ID indicating the token type,
"128" indicates the length of the audit record in bytes,
"2" is the Event ID identifying the event type,
"sendto(2)" is the Event ID monitor giving the descriptive information about the event type, and
"Tue 07 Mar 2000 10:50:19 AM EST, + 834856051 msec" shows the date and time when the audit event is recorded.

An argument token describes an argument of the system call generating the audit event. A socket token gives information about the socket for communication. A subject token describes the process that generates the system call for the audit event, including user IDs, group IDs, process ID, session ID, and so on. A return token describes the return status of the system call for the audit event. A trailer token at the end of an audit record allows the backward search of the audit record in an audit file.

BSM audit records are stored in the binary form. The *praudit* command allows the translation of audit data in the binary form into a user-readable format. The *auditreduce* command allows the audit events to be selected based on the time, user, and event.

System log data and application log data capture activities of given system programs and application programs, respectively (Ye, 2003). System log data may record user login and logout activities, access to privileged programs and network services, and so on. Application log data generated by a Web application may record information such as the host name of the user accessing a Web site, the user ID if a user ID is used for authentication, the date and time of the Web request, the name of the page requested and the protocol used for the request, the number of bytes returned for the request, and so on.

Computer audit/log data can be useful in detecting intrusive activities in the gaining-access, maintaining-access, launching-further-attack, and covering-track phases, for example, gaining root user privileges, creating a user account, installing a DoS attack program, and modifying audit/log files. Those intrusive activities are captured in computer audit/log data. For example, intrusive activities of gaining root user privileges, creating a user account and modifying audit/log files are not usually performed by regular users. Hence, those intrusive activities produce different types of audit events from the types of audit events from regular user activities. The header token of each BSM audit event reveals the type of the audit event that can be useful in detecting intrusions.

EXTRACTION AND REPRESENTATION OF ACTIVITY FEATURES FOR INTRUSION DETECTION

Network sniffing and computer auditing/logging have performance impact on the disk space and CPU (central processing unit) time of the host machine running the sniffing, auditing, or logging facility. Not all information included in a data packet or an audit/log record of an event is useful for intrusion detection. It is important to know what features of system activities, and thus

what information from large amounts of system activity data, are useful to intrusion detection or are effective to distinguish intrusive activities from normal activities. Using and keeping only the information about those features of system activities can enhance the performance and reduce the overhead of intrusion detection. The features of system activities that have been extracted from system activity data for intrusion detection in existing work are described in the next section.

Features of System Activities

In existing work on intrusion detection the following features of system activities have been extracted from network traffic data or computer audit/log data for intrusion detection (Ye, Li, Chen, Emran, & Xu, 2001).

1. Occurrence of single attributes. A single attribute can be an individual data field of network traffic data and computer audit/log data such as the destination port for a network application, the type of audit event, the error message, or the source IP address, or can be an aggregate attribute such as the duration of a connection derived from all the data packets in that connection session and the CPU time spent by a process derived from all the audit events in that process.
2. Frequency of single attributes, for example, count of consecutive password failures.
3. Occurrence of multiple attributes, for example, the source port of 60000 together with the destination port of 2140 that are used by a well-known intrusion.
4. Frequency histogram (distribution) of multiple attributes or multiple attribute values, for example, frequency distribution of audit event types.
5. Sequence or transition of attributes, for example, event transition and event sequence.
6. Intensity of events, for example, number of data packets per time unit and number of audit events per time unit.

The single occurrence of an attribute, for instance, the "su root" command in UNIX environments, may indicate a possible violation of normal network or host operations. Considering the relatively low frequency of an attribute in a normal condition, for example, login failure, a high frequency of the attribute may indicate a possible deviation from normal network or host operations. The frequency distribution of attributes or attribute values, such as commands invoked by a user may be used to reveal a compromised user account when the distribution in the recent past is different from the long-term historical profile. The event intensity, for example, the number of data packets to a host machine per second, may indicate a possible DoS attack at the host machine through traffic flooding.

Features 1–6 can be grouped into three general categories: frequency feature, order feature, and intensity feature. Frequency features include features 1–4 about the frequency of single attributes or the frequency distribution of multiple attributes or multiple attribute values. The single occurrence of attributes in features 1 and 3 can be considered as a frequency feature that has only two possible values, 0 and 1. An intrusion typically consists of a series of events. A frequency feature extracted from a series of events can provide information about the collective activity level of these events. For a given series of events, a frequency feature of a single attribute value from a single event, for example, the type of an audit event, carries less information but at lower cost than the frequency distribution of multiple attributes or multiple attribute values from the same series of multiple events, for instance, the frequency distribution of all audit event types. One study (Ye, Li, et al., 2001) reveals that the frequency feature of

a single attribute value from a single event is not sufficient for intrusion detection. Northcutt et al. (2001) also points out that any detection of an intrusion based on a single data packet is very difficult to validate.

The order feature includes feature 5 about the sequential order of attributes. The order feature contains more information but at higher cost than the frequency feature for the same series of events, because the frequency feature does not take into account the sequential order of events. Ye, Li, et al. (2001) reveals that the order feature of a given event sequence provides an additional advantage to the frequency feature of the same event sequence for intrusion detection.

The intensity feature includes feature 6 about the intensity of attributes. This feature is important in detecting DoS attacks. The intensity feature is different from the frequency feature in that the intensity feature is measured based on a time unit, whereas the frequency feature is not necessarily measured based on a time unit.

Feature Representation

For a frequency feature we can use a vector, $(X_1, \ldots, X_K)$, to represent the frequency of K attributes or attribute values, where K is an integer greater than or equal to 1, and X_k can take a binary value or an integer value for $k = 1, \ldots, K$. For the order feature we can use $(X_1, \ldots, X_T)$ to represent a time series data for a sequence of events, where X_t denotes an event occurring at time t, for $t = 1, \ldots, T$. Note that the vector representation of a frequency feature for a given event sequence, $(X_1, \ldots, X_K)$, can be derived from the times-series representation of the order feature for the same event sequence, $(X_1, \ldots, X_T)$. For the intensity feature, we can use $X(t)$ to represent the event intensity at time t.

The frequency feature, the order feature, or the intensity feature is often extracted from a sequence of events in the recent past of the current event to capture a short-term behavior. The short-term value of that feature is then compared with the long-term profile of that feature to determine the difference that is in turn used to detect intrusions. Three methods mainly have been used to obtain the measurement of a feature in the recent past of the current event: time unit, moving/sliding window, and decaying factor (Ye, 2003).

The time-unit method measures a given feature from all the events within a time unit, for example, second, minute, hour, day, and so on. The recent past can be defined as the last time unit, for instance, the last second. If the recent past can also be defined as the last T time units, the value of a given feature in the recent past can be obtained by averaging the values of the feature from those last T time units. For example, the intensity feature is often measured in terms of the number of events (e.g., data packets and audit events) per second.

The moving window method uses a window of a fixed or variable size to view a sequence of events up to the present event n. A measurement of a given feature is taken from the sequence of events viewed through the window. For the following sequence of events up to the present event n,

$$\mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_{n-6}, \mathrm{E}_{n-5}, \mathrm{E}_{n-4}, \mathrm{E}_{n-3}, \mathrm{E}_{n-2}, \mathrm{E}_{n-1}, E_n$$

the window of size 7 at the present event n allows us to view $\mathrm{E}_{n-6}, \mathrm{E}_{n-5}, \mathrm{E}_{n-4}, \mathrm{E}_{n-3}, \mathrm{E}_{n-2}, \mathrm{E}_{n-1}, \mathrm{E}_n$ as the event sequence in the recent past from which the measurement of a given feature is taken. Then the event is advanced to $n + 1$. The window is also moved to $n + 1$, and the event sequence of E_{n-5}, E_{n-4}, E_{n-3}, E_{n-2}, E_{n-1}, E_n, E_{n+1} is viewed in the window. In general, $\mathrm{E}_{n-\mathrm{L}+1}, \ldots,$ E_n is the event sequence in the recent past of the present event n for the window size of L.

Unlike the method of time unit, the method of moving window uses the number of events rather than the number of time units to define the recent past. Event intervals in time often

TABLE 26.1
An Example of Measuring the Frequency Distribution of Event Types in the Recent Past Using the Moving Window Method

Event	*Event Sequence in the Moving Window of the Present Event*	$(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$
n	E_{n-6},=2, E_{n-5},=4, E_{n-4}=8, E_{n-3}=3, E_{n-2}=4, E_{n-1}=7, E_n=1	(1, 1, 1, 2, 0, 0, 1, 1, 0, 0)
$n+1$	E_{n-5},=4, E_{n-4}=8, E_{n-3}=3, E_{n-2}=4, E_{n-1}=7, E_n=1, E_{n+1}=10	(1, 0, 1, 2, 0, 0, 1, 1, 0, 1)
$n+2$	E_{n-4}=8, E_{n-3}=3, E_{n-2}=4, E_{n-1}=7, E_n=1, E_{n+1}=10, E_{n+2}=6	(1, 0, 1, 1, 0, 1, 1, 1, 0, 1)

vary with different times of a day, for example, daytime and nighttime. In a given time unit there may be few or no events, making it difficult to take the measurement of a given feature for that time unit. A large variation of event intervals in time may result in a large variance of values for a given feature, which may not be desirable. Using the number of events rather than the number of time units helps overcome this problem and produces stable measurements for a given feature. Moreover, when we are interested in the frequency distribution of event types in the recent past, we focus on the mixture of different event types in an event sequence of the same length rather than in an event sequence within a time unit. In such a case, using the method of moving window is more reasonable than using the method of time unit to define the recent past.

Table 26.1 shows an example of measuring the frequency distribution of event types in the recent past defined by the window size of 7 from the following sequence of audit events:

$$E_{n-6},=2,\ E_{n-5},=4,\ E_{n-4}=8,\ E_{n-3}=3,\ E_{n-2}=4,\ E_{n-1}=7,\ E_n=1,\ E_{n+1}=10,\ E_{n+2}=6.$$

Because we are interested in only the information of the event type, only the event type of each event is given in the above sequence of events. Suppose that there are 10 different event types. We use $(X_1, \ldots, X_{10})$ to represent the frequency distribution of the 10 event types in the recent past.

Like the method of moving window, the method of decaying factor defines the recent past based on events rather than time units. However, in the method of moving window the events in the window are treated equally. In the above example E_{n-5}=4 and E_{n-2}=4 in the window for the present event n give us the frequency count of 2, although E_{n-2}=4 is more recent than E_{n-5}=4. In the method of decaying factor a weight is assigned to each event when computing the value for a given feature. The weight value assigned to an event depends on how recent the event is. A more recent event gets more weight in the computation of the value for a given feature. Hence, the weight reflects the decaying of an event, and the measurement of a given feature takes into account the decaying of events.

In recent studies on intrusion detection (Emran & Ye, 2002; Ye & Chen, 2001; Ye, Li, et al., 2001), the authors used the exponentially weighted moving average (EWMA) method to take into account the decaying of events. Let $X(n)$ be the measurement of a given feature at event n. The following formula is used to compute $X(n)$:

$$X(n) = \lambda \times x + (1 - \lambda) \times X(n-1) \tag{1}$$

where x is a value from event n, and λ is a constant in the range of [0, 1] that determines the decaying rate. Using this formula event n receives the weight of λ, event $n-1$ receives the weight of $\lambda(1-\lambda)$, and event $n-k$ receives the weight of $\lambda(1-\lambda)^k$. If we are interested in

TABLE 26.2
An Example of Computing the Frequency Distribution of Event Types in the Recent Past Using the Method of Decaying Factor

Event	$(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$
$n-6$	$X_1 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_2 = 0.3 \times 1 + (1-0.3) \times 0 = 0.3$
	$X_3 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_4 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_5 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_6 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_7 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_8 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_9 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_{10} = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
$n-5$	$X_1 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_2 = 0.3 \times 0 + (1-0.3) \times 0.3 = 0.21$
	$X_3 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_4 = 0.3 \times 1 + (1-0.3) \times 0 = 0.3$
	$X_5 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_6 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_7 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_8 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_9 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_{10} = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
$n-4$	$X_1 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_2 = 0.3 \times 0 + (1-0.3) \times 0.21 = 0.147$
	$X_3 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_4 = 0.3 \times 0 + (1-0.3) \times 0.3 = 0.21$
	$X_5 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_6 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_7 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_8 = 0.3 \times 1 + (1-0.3) \times 0 = 0.3$
	$X_9 = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$
	$X_{10} = 0.3 \times 0 + (1-0.3) \times 0 = 0.0$

the long-term measurement of a given feature in the past, we should use a small value of λ, for example, 0.0001. If we are interested in the short-term measurement of a given feature in the recent past, we should use a large value of λ, perhaps, 0.3. In general, if we wish to have an exponential weighting method that is equivalent to an L-event moving window method, λ is set as follows (Montgomery & Mastrangelo, 1991):

$$\lambda = \frac{2}{L+1} \tag{2}$$

Table 26.2 gives an example of computing the frequency distribution of event types in the recent past, $(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$, for the same sequence of events used in Table 26.1:

$$E_{n-6}{=}2,\ E_{n-5},{=}4,\ E_{n-4}{=}8,\ E_{n-3}{=}3,\ E_{n-2}{=}4,\ E_{n-1}{=}7,\ E_n{=}1,\ E_{n+1}{=}10,\ E_{n+2}{=}6$$

using the decaying factor method based on EWMA. $X_i(n)$ is computed as follows for $i = 1, \ldots, 10$:

$$X_i(n) = \lambda \times x_i + (1-\lambda) \times X_i(n-1) \tag{3}$$

where $x_i = 1$ if event n falls in the ith event type, or $= 0$ otherwise, provided that $(n - 6) = 1$ is the beginning of the event sequence, the value of $X_i(0)$ is initialized to zero for $i = 1, \ldots, 10$, and λ is set to 0.3.

EXISTING INTRUSION DETECTION TECHNIQUES

There are two general approaches to intrusion detection: signature recognition and anomaly detection (Debar, Dacier, & Wespi, 1999; Lippmann et al., 2000; Ye, Giordano, & Feldman, 2001; Ye, Li, et al., 2001). Signature recognition techniques, also referred to as "misuse detection" in some literature, compare activities in a computer and network system with signatures of known intrusions, and signal intrusions when there is a match. For a subject (user, file, privileged program, host, network, etc.) of interest, anomaly detection techniques establish a profile of the subject's long-term normal behavior (norm profile), compare the observed behavior of the subject with its norm profile, and signal intrusions when the subject's observed behavior deviates significantly from its norm profile. Signature recognition techniques and anomaly detection techniques often are used together to complement each other for intrusion detection.

Signature recognition techniques utilize intrusion signatures, profiles of intrusion characteristics, and consider the presence of an intrusion signature as evidence of an intrusion. Anomaly detection techniques use only data of normal activities in a computer system for training and building a norm profile. Signature recognition techniques rely on data of both normal and intrusive activities for learning intrusion signatures, either manually or automatically, through data mining.

Most commercial intrusion detection systems are based on signature recognition techniques (Escamilla, 1998; Ye, 2003). Intrusion signatures have been characterized as strings (e.g., command names), event frequency distributions, event sequences, activity graphs, and intrusion scenarios with event sequences, event preconditions, and target compromised states. Intrusion signatures have been represented using finite state machines (Vigna, Eckmann, & Kemmerer, 2000), colored Petri nets (Kumar, 1995), association rules (Lee, Stolfo, & Mok, 1999), decision trees (Ye, Li, et al., 2001), clusters (Li & Ye, in press) and production rules of expert systems (Anderson, Frivold, & Valdes, 1995) to store and recognize intrusion signatures. Intrusion signatures are either manually encoded or automatically learned through data mining. However, signature recognition techniques have a limitation in that they cannot detect novel intrusions the signatures of which are unknown.

Anomaly detection techniques capture both known intrusions and unknown intrusions if the intrusions demonstrate a significant deviation from a norm profile. Existing anomaly detection techniques differ mainly in the representation of a norm profile and the inference of intrusions using the norm profile.

Forrest, Hofmeyr, and Somayaji (1997); Ghosh, Schwatzbard, and Shatz (1999); and Warrender, Forrest, and Pearlmutter (1999) collect short sequences of system calls or audit events that appear in normal activities, represent those short sequences as strings, store those strings as a norm profile, and employ either negative selection or positive selection to determine whether an observed string deviates from the string-based norm profile. Strings-based anomaly detection techniques model the order feature of normal activities, that is, event sequences. Ko, Fink, and Levitt (1997) uses predicates in formal logic to specify normal activities for a norm profile and employ logical reasoning to infer the consistency of observed activities using that norm profile.

A number of studies, (Denning, 1987; Emran & Ye, 2002; Javitz & Valdes, 1991, 1994; Jou et al., 2000; Li & Ye, 2002; Ye, Borror, & Zhang, in press; Ye & Chen, 2001; Ye, Emran,

Chen, & Vilbert, 2002; Ye, Li, et al., 2001) use statistical distributions to model the frequency feature and the intensity feature of normal activities for a norm profile and employ statistical tests to determine whether observed activities deviate significantly from the norm profile. An advantage of statistical-based anomaly detection techniques is their capability of explicitly representing and handling variations and noises in normal activities. Anomaly detection techniques based on formal logic lack such a capability of noise handling and variance representation. Strings-based anomaly detection techniques must rely on a large, costly repository of short sequences of normal events to capture variations of normal activities.

Both artificial neural networks and stochastic models (e.g., Markov chain model and hidden Markov model) have been used to model the order feature of normal activities (e.g., event transitions or event sequences) for a norm profile and detect intrusions based on the deviation of the observed events from the expected event or based on the probabilistic support of the norm profile to the observed events (Debar, Becker, & Siboni, 1992; DuMouchel, 1999; Eskin, Lee, & Stolfo, 2001; Ghosh, Schwatzbard, & Shatz, 1999; Schonlau et al., 1999; Scott, 2002; Warrender et al., 1999; Ye, Ehiabor, & Zhang, 2002; Ye, Zhang, & Borror, in press). Ye et al. (2002) reveals the better performance of the Markov chain model compared with artificial neural networks for intrusion detection. Despite that, the Markov chain technique for intrusion detection is not robust to the degree in which normal activities (noises) and intrusive activities (signals to detect) are mixed in system activity data (Ye et al., in press). Noise cancellation techniques have been developed to remove effects of normal activities and thus make intrusive activities more distinct for detection (Ye, Chen, & Mou, 2002).

As can be seen from the above review of existing intrusion detection techniques, many data mining techniques have been applied to intrusion detection, including decision trees (Ye, Li, et al., 2001), association rules (Lee, Stolfo, & Mok, 1999), clustering algorithms (Li & Ye, 2002), artificial neural networks (Debar et al., 1992; Ghosh et al., 1999), Markov models (DuMouchel, 1999; Ju & Vardi, 1999; Schonlau et al., 1999; Scott, 2002; Warrender et al., 1999; Ye et al., 2002, in press), and statistical anomaly detection (Emran & Ye, 2002; Li & Ye, 2002; Ye, Borror, & Zhang, in press; Ye & Chen, 2001; Ye, Emran, et al., 2002; Ye, Li, et al., 2001). In the next section a multivariate statistical anomaly detection technique to intrusion detection is described in detail.

APPLICATION OF STATISTICAL ANOMALY DETECTION TECHNIQUES TO INTRUSION DETECTION

As an example, this section applies Hotelling's T^2 test and the chi-square distance test to intrusion detection through anomaly detection. Materials in this section are taken from Ye, Emran, Chen, & Vilbert (2002) with the permission of IEEE Press. In this section Hotelling's T^2 test and the chi-square distance test are first described. Then the data set used to test the performance of Hotelling's T^2 test and the chi-square distance test are presented. The testing performance of these two multivariate statistical anomaly detection techniques is also provided.

Hotelling's T^2 Test and Chi-Square Distance Test

Let $X_i = (X_{i1}, X_{i2}, \ldots, X_{ip})$ denote the ith observation of p measures on a process or system. Assume that when the process is operating normally (in control), the population of X follows a multivariate normal distribution with the mean vector μ and the covariance matrix Σ. Using

a data sample of size n, the sample mean vector $\bar{X}$ and the sample covariance matrix S are usually used to estimate μ and Σ (Chou, Mason, & Young, 1999), where

$$\bar{X} = (\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_p) \tag{4}$$

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})'. \tag{5}$$

Hotelling's T^2 statistic for an observation X_i is determined by the following:

$$T^2 = (X_i - \bar{X})' \, S^{-1} (X_i - \bar{X}). \tag{6}$$

A large value of T^2 indicates a large deviation of the observation X_i from the in-control population. We can obtain a transformed value of the T^2 statistic, $\frac{n(n-p)}{p(n+1)(n-1)} T^2$ which follows an F distribution with p and $n - p$ degrees of freedom, by multiplying T^2 by the constant

$$\frac{n(n-p)}{p(n+1)(n-1)}.$$

If the transformed value of the T^2 statistic is greater than the tabulated F value for a given level of significance, α, then we reject the null hypothesis that the process is in control (normal) and thus signal that the process is out of control (anomalous).

If X does not follow a multivariate normal distribution, the transformed value of the T^2 statistic may not follow an F distribution. As a result, we cannot use the tabulated F value as a signal threshold to determine whether a transformed value of the T^2 statistic is large enough for an out-of-control signal. However, if the p variables are independent and p is large (usually greater than 30), T^2 follows approximately a normal distribution according to the central limit theorem, regardless of what distribution each of the p variables follows. Using an in-control sample of T^2 values, the mean and standard deviation of the T^2 population can be estimated from the sample mean $\bar{T}^2$and the sample standard deviation S_{T^2}. The in-control limits to detect out-of-control anomalies are usually set to 3-sigma control limits as determined by $[\bar{T}^2 - 3S_{T^2}, \bar{T}^2 + 3S_{T^2}]$. Because we are interested in detecting significantly large T^2 values for intrusion detection, we need to set only the upper control limit to $\bar{T}^2 + 3S_{T^2}$ as a signal threshold. That is, if the T^2 for an observation is greater than $\bar{T}^2 + 3S_{T^2}$, we signal an anomaly.

An out-of-control signal from the T^2 test can be caused by a shift from the in-control mean vector (mean shift), a departure from the in-control covariance structure or variable relationships (counterrelationship), or combinations of the two situations. In a mean shift situation one or more of the p variables is out of control. In a counterrelationship situation a relationship between two or more of the p variables changes from the variable relationship established in the covariance matrix. Although the T^2 test detects both mean shifts and counterrelationships, the T^2 test is more sensitive to counterrelationships than mean shifts, because the T^2 test relies largely on the correlated structure of variables (covariance matrix) for anomaly detection (Chou et al., 1999).

The computation of the covariance matrix and its inverse in Equation 6 is computationally costly for a large data set with many variables. Hence, a scalable multivariate statistical anomaly detection technique is developed based on the chi-square distance measure. If we have p variables to measure and X_j denotes an observation of the jth ($1 \leq j \leq p$) variable, the χ^2 distance measure is given by the following:

$$\chi^2 = \sum_{j=1}^{p} \frac{(X_j - \bar{X}_j)^2}{\bar{X}_j} \tag{7}$$

This test statistic measures the distance of a data point from the center of a data population. Hence, we refer to this test as the chi-square distance test. When the p variables are independent and p is large (usually greater than 30), the X^2 statistic follows approximately a normal distribution according to the central limit theorem, regardless of what distribution each of the p variables follows. Using an in-control sample of χ^2 values, the mean and standard deviation of the χ^2 population can be estimated from the sample mean $\bar{\chi}^2$ and the sample standard deviation S_{χ^2}. The in-control limits to detect out-of-control anomalies are usually set to 3-sigma control limits as determined by $[\bar{\chi}^2 - 3S_{\chi^2}, \bar{\chi}^2 + 3S_{\chi^2}]$. Because we are interested in detecting significantly large χ^2 values for intrusion detection, we need to set only the upper control limit to $\bar{\chi}^2 + 3S_{\chi^2}$ as a signal threshold. That is, if the χ^2 for an observation is greater than $\bar{\chi}^2 + 3S_{\chi^2}$, we signal an anomaly.

Both the T^2 test statistic and the χ^2 test statistic measure the distance of an observation from the multivariate mean vector of a population. The T^2 test statistic uses the statistical distance that incorporates the multivariate variance-covariance matrix, whereas the χ^2 test statistic uses the chi-square distance. The chi-square distance is similar to a Euclidean distance but uses the average value on each variable to scale the Euclidean distance on that variable or dimension.

In general, anomalies involving multiple variables can be caused by shifts from the means of these variables (mean shifts), departures from variable relationships (counterrelationships), or combinations of mean shifts and counterrelationships. In contrast to the T^2 statistic, the χ^2 statistic does not account for the correlated structure of the p variables. Only $\bar{X}$ is estimated to establish the norm profile according to Equation 4. Hence, the T^2 test detects both mean shifts and counterrelationships, whereas the χ^2 test detects only the mean shift on one or more of the p variables.

Data Source and Representation

BSM audit data from the Sun SPARC 10 workstation with the Solaris 2.5.1 operating system is used. There are 284 different types of events in BSM audit data from Solaris 2.5.1. Each audit event is characterized by the type of audit event. That is, only the event type information from each audit record is used for intrusion detection. Many studies have shown the effectiveness of the event type information for intrusion detection (Ye, Li, et al., 2001).

Nine weeks of audit data obtained from the DARPA Intrusion Detection Evaluation 1998 data generated by the MIT Lincoln Laboratory (http://ideval.ll.mit.edu) contain 7 weeks of labeled training data and 2 weeks of unlabeled testing data. Because the testing data are not labeled, it is difficult to evaluate the performance of the techniques used. Hence, the training data set was used for both training and testing in a study. During training, audit data of activities with the label of normal event was used to build the norm profile. During testing, the label of audit data was removed and the techniques previously discussed here were used to generate a label (normal or intrusive). The labels of activities from testing were then compared with the given labels for evaluating the performance of the techniques.

There are in total 35 days of data in the 7 weeks of training data. Four days of data are chosen as a representative of the entire training data set. Two days are chosen with relatively few intrusions and 2 days with comparatively more intrusions, varied in length. Week 1, Monday data was designated as day 1 data; Week 4, Tuesday data as day 2 data; week 4, Friday data as day 3 data; and week 6, Thursday data as day 4 data. Table 26.3 summarizes the statistics about these 4 days of data.

As we can see from Table 26.3, the average session length was comparatively smaller in day 2 and day 3 than that in day 1 and day 4. Around 3% of audit events on day 2 and day 4 were due to intrusive activities, whereas less than 0.80% of audit events in day 1 and day 3

TABLE 26.3
Statistics About the Data Set for Intrusion Detection

	Day 1	*Day 2*	*Day 3*	*Day 4*
Event information				
Number of events	744,085	1,320,478	2,249,505	925,079
Number of intrusive events	3,090	36,575	16,524	31,476
Percentage of intrusive events	0.42%	2.77%	0.73%	3.40%
Session information				
Number of sessions	298	503	339	447
Number of intrusive sessions	2	131	29	14
Percentage of intrusive sessions	0.67%	26.04%	8.55%	3.13%
Number of events per normal session				
Average	2,503	3,451	7,203	2,064
Minimum	1	69	74	96
Maximum	253,827	462,287	1,019,443	214,705
Number of events per intrusive session				
Average	1,545	279	570	2,248
Minimum	1,101	142	166	1,107
Maximum	1,989	1,737	4,986	2,841

Note: © 2002 IEEE. Reprinted, with permission, from Ye, Emran, Chen, & Vilbert (2002).

data were from normal activities. In terms of sessions, almost one fourth of the sessions in day 2 and one twelfth of the sessions in day 3 were intrusion sessions. Day 1 contained mostly normal sessions, and day 4 also had few intrusion sessions. A total of 176 instances of 9 types of intrusions are present in these 4 days of data. Because the objective is to detect any ongoing intrusions rather than any particular type of intrusion, the concern is about how many of these intrusion sessions can be detected.

Only the normal events of the first 2 days of data were used for training. Day 1 and day 2 contain 740,995 and 1,283,903 audit events arising from 296 and 372 normal sessions, respectively. Day 3 and day 4, which contain 2,232,981 and 893,603 audit events for normal activities, respectively, were used for testing. Numbers of audit events for intrusive activities in these 2 days were 16,524 and 31,476 arising from 29 and 14 intrusion sessions, respectively. Day 3 contained 339 sessions and day 4 contained 447 sessions in total comprising both normal and intrusive sessions. In this data set an intrusion occurred in one session only.

Activities on a host machine are captured through a sequence of audit events, each of which is characterized by the event type. For intrusion detection, we want to build a long-term profile of normal activities, and to compare the activities in the recent past of the present event to the long-term norm profile for detecting a significant deviation of the present event. We define audit events in the recent past of the present event n by a vector of $(X_1, X_2, \ldots, X_{284})$ for the frequency distribution of the 284 event types, respectively, using the decaying factor method described previously. An observation for the nth event, $(X_1, X_2, \ldots, X_{284})$, is obtained as follows.

$$X_i(n) = \lambda * 1 + (1 - \lambda) * X_i(n - 1) \tag{8}$$

if the nth audit event belongs to the ith event type

$$X_i(t) = \lambda * 0 + (1 - \lambda) * X_i(t - 1)$$

if the type of the nth audit event is different from the ith event type, where $X_i(n)$ is the observed value of the ith variable in the vector of an observation for the nth audit event, λ is a

smoothing constant that determines the decay rate, and $i = 1, \ldots, 284$. We initialize $X_i(0)$ to 0 for $i = 1, \ldots, 284$, and set λ to 0.3.

Application of Hotelling's T^2 Test and Chi-Square Distance Test

For the T^2 test, the long-term profile of normal activities is modeled by the sample mean vector $\bar{X}$ and the sample covariance matrix S. The audit events for normal activities in the training data give us a sample of $(X_1, \ldots, X_{284})$'s to obtain the sample mean vector $\bar{X}$ and the sample covariance matrix S. For each of the audit events in the training data and the corresponding observation of $(X_1, \ldots, X_{284})$, we compute the T^2 statistic according to Equation 6. We then determine the upper limit of T^2 in terms of $\bar{T}^2 + 3S_{T^2}$ as the signal threshold. For each of the audit events in the testing data and the corresponding observation of $(X_1, \ldots, X_{284})$, we compute the T^2 statistic according to Equation 6. The T^2 value is small if the observation is close to the norm profile. The upper limit, $\bar{T}^2 + 3S_{T^2}$, is used as the signal threshold to determine whether we should signal an event as intrusive based on the T^2 value of this event. If the T^2 value exceeds the signal threshold, we signal the audit event as intrusive. Because an intrusion session corresponds to one intrusion, we compute the session signal ratio for each session in the testing data by counting how many of the audit events in that session are signaled and dividing the signal count by the total number of audit events in that session. If the session is an intrusion session, we expect a high session signal ratio. If it is a normal session, we expect the session signal ratio to be low.

For the χ^2 distance test, the long-term profile of normal activities is modeled by the sample mean vector $\bar{X}$ only. The audit events for normal activities in the training data give us a sample of $(X_1, \ldots, X_{284})$'s to obtain the sample mean vector $\bar{X}$. For each of the audit events in the training data and the corresponding observation of $(X_1, \ldots, X_{284})$, we compute the χ^2 statistic according to Equation 7. We then determine the upper limit of χ^2 in terms of $\bar{\chi}^2 + 3S_{\chi^2}$ as the signal threshold. For each of the audit events in the testing data and the corresponding observation of $(X_1, \ldots, X_{284})$, we compute the χ^2 statistic according to Equation 7. The χ^2 value is small if the observation is close to the norm profile. The upper limits, $\bar{\chi}^2 + 3S_{\chi^2}$, is used as the signal threshold to determine whether we should signal an event as intrusive based on the χ^2 value of this event. If the χ^2 value exceeds the signal threshold, we signal the audit event as intrusive. For each session in the testing data, we compute the session signal ratio.

Testing Performance

Based on the session signal ratio for each session in the testing data from a given test (the T^2 test or the χ^2 distance test), we can set a session-wise signal threshold on the session signal ratio. If the session signal ratio for a given session exceeds the session-wise signal threshold, the session is signaled as intrusive; otherwise the session is considered normal. We have a hit if a truly intrusive session is signaled by a given test as intrusive. We have a false alarm if a truly normal session is signaled by a given test as intrusive. We can then compute the false alarm rate as the ratio of the number of false alarms on the normal sessions to the total number of normal sessions in the testing data and compute the hit rate as the ratio of the number of hits on the intrusive sessions to the total number of intrusive sessions in the testing data. We thus obtain a pair of false alarm rate and hit rate using a given session-wide signal threshold. For different values of session-wise signal threshold, we can obtain different pairs of false

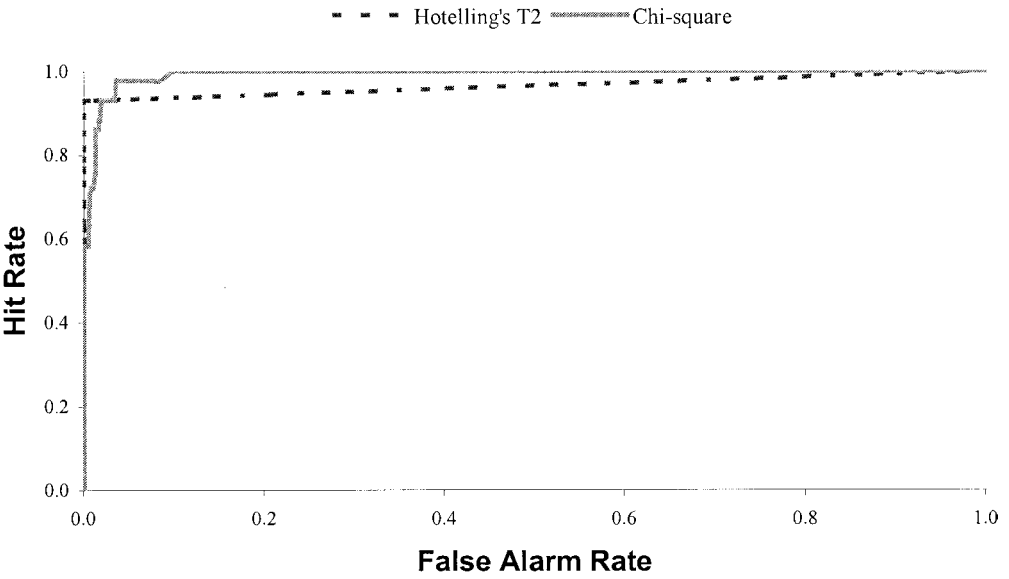

FIG. 26.1. The ROC Curves of the T^2 test and the χ^2 test based on session signal ratios. (© 2002 IEEE. Reprinted, with permission, from Ye, Emran, Chen, & Vilbert (2002)).

alarm rate and hit rate. A receiver operating characteristic (ROC) curve from signal detection theory (Kantowitz & Sorkin, 1983) can be used to plot pairs of false alarm rate and hit rate when various signal thresholds are used. The closer the ROC curve is to the top-left corner (representing the 100% hit rate and the 0% false alarm rate) of the ROC chart, the better the detection performance. Hence, the ROC curve shows the overall detection performance of a given test.

Figure 26.1 shows the ROC curves of the χ^2 and T^2 tests, respectively, based on the session signal ratios from the tests. It appears that both the χ^2 test and the T^2 test perform very well in distinguishing intrusion sessions from normal sessions. The T^2 test achieves the 95% hit rate at the 0% false alarm rate. The χ^2 test achieves the 60% hit rate at the 0% false alarm rate. However, after the 5% false alarm rate the χ^2 test performs slightly better than the T^2 test. The χ^2 test is much more scalable in processing large amounts of audit data than the T^2 test. Because large amounts of computer audit/log data must be processed in real time for intrusion detection, the scalable χ^2 test provides a more viable solution for intrusion detection than the T^2 test.

SUMMARY

This chapter showed examples of computer audit/log data and network traffic data that can be collected for intrusion detection. It presented the features of activities in computer and network systems that have been extracted from computer audit/log data and network traffic data for intrusion detection. It also reviewed existing intrusion detection studies and the use of data mining techniques in those studies. The chapters illustrated the application of data mining techniques based on multivariate statistical anomaly detection to intrusion detection.

REFERENCES

Anderson, D., Frivold, T., & Valdes, A. (1995). *Next-generation intrusion detection expert system (NIDES): A summary* (Tech. Rep. SRI-CSL-97-07). Menlo Park, CA: SRI International.

Chou, Y.-M., Mason, R. L., & Young, J. C. (1999). Power comparisons for a Hotelling's T^2 statistic. *Communications in Statistics—Simulation and Computation, 28,* 1031–1050.

Debar, H., Becker, M., & Siboni, D. (1992). A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy* (pp. 240–250). Los Alamitos, CA: IEEE Computer Society Press.

Debar, H., Dacier, M., & Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks, 31,* 805–822.

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering, SE-13,* 222–232.

DuMouchel, W. (1999). *Computer intrusion detection based on Bayes factors for comparing command transition probabilities* (National Institute of Statistical Sciences Tech. Rep. No. 91). Retrieved August, 2002 from http://www.niss.org/downloadabletechreports.html

Emran, S. M., & Ye, N. (2002). Robustness of chi-square and Canberra techniques in detecting intrusions into information systems. *Quality and Reliability Engineering International, 18,* 19–28.

Escamilla, T. (1998). *Intrusion detection: Network security beyond the firewall*. New York: Wiley.

Eskin, E., Lee, W., & Stolfo, S. J. (2001). Modeling system calls for intrusion detection with dynamic window sizes. In *Proceedings of the Second DARPA Information Survivability Conference and Exposition (DISCEX II)* (pp. 165–175). Los Alamitos, CA: IEEE Computer Society Press.

Fisch, E. A., & White, G. B. (2000). *Secure computers and networks: Analysis, design and implementation*. Boca Raton: CRC Press.

Forrest, S., Hofmeyr, S. A., & Somayaji, A. (1997). Computer immunology. *Communications of the ACM, 40*(10), 88–96.

Ghosh, K., Schwatzbard, A., & Shatz, M. (1999). Learning program behavior profiles for intrusion detection. In *Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring*.

Javitz, H. S., & Valdes, A. (1991). The SRI statistical anomaly detector. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*. Los Alamitos, CA: IEEE Computer Society Press.

Javitz, H. S., & Valdes, A. (1994, March). *The NIDES statistical component description and justification* (Tech. Rep. A010). Menlo Park, CA: SRI International.

Jou, Y., Gong, F., Sargor, C., Wu, X., Wu, S., Chang, H., & Wang, F. (2000). Design and implementation of a scalable intrusion detection system for the protection of network infrastructure. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (pp. 69–83). Los Alamitos, CA: IEEE Computer Society.

Ju, W. H., & Vardi, Y. (1999). *A hybrid high-order Markov chain model for computer intrusion detection*. (National Institute of Statistical Sciences, Tech. Rep. No. 92). Retrieved August, 2002 from http://www.niss.org/downloadabletechreports.html

Kantowitz, B. H., & Sorkin, R. D. (1983). *Human factors: Understanding people-system relationships*. New York: Wiley.

Ko, C., Fink, G., & Levitt, K. (1997). Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy* (pp. 134–144). Los Alamitos, CA: IEEE Computer Society Press.

Kumar, S. (1995). *Classification and detection of computer intrusions*. Unpublished doctoral dissertation, Purdue University, West Lafayette, Indiana.

Lee, W., Stolfo, S. J., & Mok, K. (1999). Mining in a data-flow environment: Experience in network intrusion detection. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*. New York: ACM Press.

Li, X., & Ye, N. (2002). Grid- and dummy-cluster-based learning of normal and intrusive clusters for computer intrusion detection. *Quality and Reliability Engineering International, 18,* 231–242.

Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., & Zissman, M. (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (pp. 12–26). Los Alamitos, CA: IEEE Computer Society.

Montgomery, D. C., & Mastrangelo, C. M. (1991). Some statistical process control methods for autocorrelated data. *Journal of Quality Technology, 23,* 179–193.

Northcutt, S., Cooper, M., Fearnow, M., & Frederick, K. (2001). *Intrusion signatures and analysis*. Indianapolis, IN: New Riders.

Pfleeger, C. P. (1997). *Security in computing*. Upper Saddle River, NJ: Prentice Hall PTR.

Schonlau, M., DuMouchel, W., Ju, W. H., Karr, A. F., Theus, M., & Vardi, Y. (1999). *Computer intrusion: Detecting masquerades* (National Institute of Statistical Sciences Tech. Rep. No. 95). Retrieved August, 2002 from http://www.niss.org/downloadabletechreports.html.

Scott, S. L. (2002). *Detecting network intrusion using a Markov modulated nonhomogeneous Poisson process*. Retrieved August, 2002 from http://www-rcf.usc.edu/∼sls/fraud.ps

Skoudis, E. (2002). *Counter hack*. Upper Saddle River, NJ: Prentice Hall PTR.

Stevens, W. R. (1994). *TCP/IP illustrated* (Vol. 1). Boston: Addison-Wesley.

Viega, J., & McGaw, G. (2002). *Building secure software*. Boston: Addison-Wesley.

Vigna, G., Eckmann, S., & Kemmerer, R. (2000). The STAT tool suite. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (pp. 46–55). Los Alamitos, CA: IEEE Computer Society.

Warrender, C., Forrest, S., & Pearlmutter, B. (1999). Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy* (pp. 133–145). Los Alamitos, CA: IEEE Computer Society Press.

Ye, N. (2003). *Modeling and Analysis of Cyber-Security Data*. London: Springer-Verlag.

Ye, N., Borror, C., & Zhang, Y. (in press). EWMA techniques for computer intrusion detection through anomalous changes in event intensity. *Quality and Reliability Engineering International.*

Ye, N., & Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International, 17,* 105–112.

Ye, N., Chen, Q., & Mou, J. I. (2002). Univariate and multivariate noise cancellation for computer intrusion detection. In *Proceedings of the Third IEEE SMC Information Assurance Workshop* (pp. 284–291). New York: IEEE Press.

Ye, N., Ehiabor, T., & Zhang, Y. (2002). First-order versus high-order stochastic models for computer intrusion detection. *Quality and Reliability Engineering International, 18,* 243–250.

Ye, N., Emran, S. M., Chen, Q., & Vilbert, S. (2002). Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on Computers, 51,* 810–820.

Ye, N., Giordano, J., & Feldman, J. (2001). A process control approach to cyber attack detection. *Communications of the ACM, 44*(8), 76–82.

Ye, N., Li, X., Chen, Q., Emran, S. M., & Xu, M. (2001). Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Transactions on Systems, Man, and Cybernetics, 31,* 266–274.

Ye, N., Zhang, Y., & Borror, C. M. (in press). Robustness of the Markov chain model for cyber attack detection. *IEEE Transactions on Reliability*.