AFRL-IF-RS-TR-2001-197
Final Technical Report
October 2001

A PROCESS CONTROL AND DIAGNOSIS APPROACH TO INDICATIONS AND WARNING OF ATTACKS ON COMPUTER NETWORKS

Arizona State University

Nong Ye

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2001-197 has been reviewed and is approved for publication.

APPROVED: John Feldman

JOHN FELDMAN
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFGB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
OCTOBER 2001

3. REPORT TYPE AND DATES COVERED
Final Oct 98 - Sep 99

4. TITLE AND SUBTITLE
A PROCESS CONTROL AND DIAGNOSIS APPROACH TO INDICATIONS AND WARNING OF ATTACKS ON COMPUTER NETWORKS

5. FUNDING NUMBERS
C - F30602-98-2-0005
PE - 61102F
PR - 2301
TA - 02
WU - 01

6. AUTHOR(S)
Nong Ye

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Arizona State University
Department of Industrial Engineering
PO Box 875906
Tempe AZ 85287

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFGB
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2001-197

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: John Feldman/IFGB/(315) 330-2664

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Cyber attacks are launched as a series of computer actions designed to compromise the security (e.g., availability, integrity, and confidentiality) of a computer and network system. In this report, we first illustrate a process control approach to system modeling for information assurance. A model-based design of attack detection techniques is then presented to demonstrate how a process model of a networked computer system can be applied to cyber attack detection. Then using audit data capturing activities on computer and network systems, we develop and present learning and inference algorithms of probabilistic networks with undirected links. This technique is used to represent the symmetric relations of audit event types during normal activities, to build a long-term profile of normal activities, and to perform anomaly detection. The resultant probabilistic network is then trained with audit data from both normal activities and computer attack activities. The test results demonstrate very promising performance in detecting cyber attacks.

14. SUBJECT TERMS
Process Control Modeling, Information Attack Detection, Intrusion Detection, Anomaly Detection, Probabilistic Networks

15. NUMBER OF PAGES
60

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figure Captions

List of Table Captions

Abstract

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. Attacks on information systems have presented serious threats to the reliable operation of information systems. The detection of those attacks plays an important role in assuring the reliability of information systems. In this report, we first illustrate a process control approach to system modeling for information assurance. A model-based design of attack detection techniques is presented to demonstrate how a process model of a computer and network system supports cyber attack detection. Then in order to use the audit data to capture activities on a computer and network system and detect cyber attacks, we develop and present the learning and inference algorithms of probabilistic networks with undirected links in this report. The technique of probabilistic networks with undirected links is used to represent the symmetric relations of audit event types during normal activities, build a long-term profile of normal activities, and perform anomaly detection for cyber attack detection. The probabilistic networks, trained with the audit data of normal activities, demonstrate a very promising performance in detecting attack activities during the testing with the audit data of both normal activities and attack activities.

1. Introduction

A cyber attack is an attack on a computer and network system, consisting of computer actions (e.g., remote or local connection, computer file access, program execution, etc.) to compromise the secure operation of the computer and network system. As we increasingly rely on information infrastructures to support critical operations in defense, banking, telecommunication, transportation, electric power and many other systems, cyber attacks have become a significant threat to our society with potentially severe consequences [1-2].

A computer and network system must be protected to assure security goals such as availability, confidentiality and integrity, through prevention, detection, isolation, assessment, reaction, and vulnerability testing. Attack prevention can be enforced through firewalls and guards, boundary control with security policies, authentication, and encryption. Attack detection identifies cyber attacks being leaked through the fence of prevention and acting on a computer and network system. Attack isolation reveals the source and path (course of actions or core events) of a cyber attack leading to observed attack symptoms, as well as affected entities (e.g., users, files, programs, hosts, and/or domains). Attack assessment determines the degree and nature of damage to affected entities with respect to security risk. Attack reaction takes control actions to get an attacker out of a computer and network system, maintains the system operation even in a degraded mode, and eventually recovers the system back to a normal state. Vulnerability testing looks for points (e.g., a weak password) of a computer and network system that make the system vulnerable to cyber attacks.

Information assurance activities present much resemblance to process control activities that are usually carried out to assure the safe operation of many engineering systems, such as energy generation systems in nuclear power plants. Process control activities include:

- System planning, to design, specify and implement laws and rules governing the safe system operation;
- System control, to assure the safe operation of the system through diagnostic control and routine maintenance:
- Diagnostic control, to monitor on-line system operation data for the presence of fault symptoms, trace the source and path of faults, assess the impact of faults, and take control actions to recover the system back to a normal state;
- Routine maintenance, to collect and use the historic data of system component reliability to set up an inspection and maintenance schedule, and perform scheduled inspection and maintenance.

When placing information assurance in the context of process control, attack prevention is analogous to system planning. Attack detection, isolation, assessment and reaction are parts of diagnostic control. Vulnerability testing is similar to routine maintenance.

An advantage of applying process control to information assurance is to gain a system-centered paradigm that assists in modeling a computer and network system and designing a model-based information assurance system. Existing techniques for information assurance fall short of system modeling, even though system modeling is imperative to understand how cyber attacks work in a computer and network system. An understanding of system operation and attack mechanisms is the foundation of designing and integrating information assurance activities.

In chapter 2 of this report, we illustrate a process control approach to system modeling for information assurance, which leads to a Cyber Attack Control System (CACS). A model-based design of attack detection techniques is presented to demonstrate how a process model of a computer and network system supports cyber attack detection.

As stated above, 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. For example, a denial-of-service attack attempts to flood the communication link to a host machine with large volumes of data packets and thus make services from the host machine unavailable. A remote-to-user attack attempts to gain unauthorized access to a user account on a host machine and then to damage the integrity and confidentiality of data on the host machine through the compromised user account. A regular user on a host machine launches a user-to-root attack to gain the privileges of a root user and then to compromise the availability, integrity and confidentiality of the host machine. The detection of those attacks plays an important role in assuring the reliability of information systems. Attack detection signals cyber attacks acting on information systems.

Existing attack detection techniques fall in two major categories: anomaly detection and signature recognition [3-6]. For a subject (e.g., user, file, and privileged program) of interest, anomaly detection techniques establish a profile of the subject's normal behavior (norm profile), compare the observed behavior of the subject with its norm profile, and signal attacks when the subject's observed behavior deviates significantly from its norm profile. Signature recognition techniques recognize the signatures of known attacks, match the subject's observed behavior with those known signatures, and signal attacks when there is a match. Signature recognition techniques cannot detect novel attacks whose signatures are unknown [7-16]. This study focuses on anomaly detection techniques for detection attacks on information systems.

Anomaly detection techniques consider the deviations of a subject's observed behavior from its norm profile as symptoms of attacks. If novel attacks generate behavior different from the norm profile, novel attacks can be detected through anomaly detection techniques. Several anomaly detection techniques exist using statistical-based, sequences-based, logic-based, and rules-based norm profiles [17-30]. In SRI International's IDES/NIDES systems, a statistical-based profiling technique is used to represent the expected normal behavior of a subject and variance due to noises [20-23]. The statistical-based profiling technique overcomes the problems with the sequences-based, logic-based and rules-based profiling technique in representing noises and variances. However, SRI's statistical-based profiling technique is not robust to the assumption of normally distributed data. If data are not close to a normal distribution, the technique cannot produce reasonable results.

In chapter 3 of this report, we present a probabilistic network technique that we develop to capture and represent the profile of the normal behavior. Probabilistic inference is used for detecting anomalies. Since no specific forms of statistical distributions and distribution-based statistical inferences are used in the profile representation and anomaly detection, our technique overcomes the problem with the normality assumption.

This paper presents the probabilistic network technique for cyber attack detection and the process control approach to cyber attack detection in chapter 2 and in chapter 3 respectively.

2. Process Control Approach to Cyber Attack Detection

In this chapter, we present a model-based design of attack detection techniques to demonstrate how a process model of a computer and network system supports cyber attack detection.

2.1. A Process Control Approach to System Modeling

For large-scale, complex engineering systems, system models are built through levels of abstraction to divide and conquer the system complexity by capturing different aspects of systems at different levels. Four levels of abstraction have been found important to process control of many engineering systems, including nuclear power plants and manufacturing systems [1-2]. The four levels are: objective, conceptual, functional and physical.

At the objective level, the goals of a system are stated. The goals of the system are transformed into the time-based operational behavior (e.g., states and state transitions) of the system at the conceptual level. Each operation at the conceptual level is supported by a group of tasks at the functional level. Those tasks take place in a functional architecture of the system that describes functional components of the system and their relationships. A functional architecture can be implemented in different physical forms. Functional tasks are implemented at the physical level through physical actions that occur in the physical anatomy of the system.

Hence, the conceptual level deals with temporal relationships (relationships between behaviors at different times) in the system, whereas the functional level and the physical level concern spatial relationships (relationships between components at different locations) in the system. As we move from the objective level to the physical level, we shift from a general, abstract understanding of the system to a specific, concrete understanding of the system. The system models at different levels are linked in that activities and entities at one level are aggregated into activities and entities at a higher level, and are decomposed into activities and entities at a lower level. The system models at all levels form a complete process model of the system.

The process model is two-dimensional in that it defines the system at multiple levels of abstraction (objective, conceptual, functional and physical) and at multiple scales of scope (components, subsystems and system). The dimension of scale may consist of more than three scales, if necessary. For example, subsystems can be further divided into subsystems of subsystems, and so on.

We develop a multi-level, multi-scale process model of a computer and network system to capture the security-related system behavior [31]. What entities and activities of a computer and network system are included in the process model depends on their relevance to system security. Figure 2-1 illustrates the process model of a security-aware computer and network system.

We first examine the process model in the dimension of the abstraction levels. The *objective* level states security goals of an entity (component, subsystem or system) such as confidentiality, integrity, and availability. The *conceptual* level describes security-critical states and state transitions of an entity. The state-transition model focuses on the temporal structure of the entity behavior. The *functional* level models the spatial structure of a functional entity (e.g., user, file, program, process, host, and enclave) which exists in the software domain. The spatial structure of a functional entity describes functional elements of the entity and functional relationships (e.g., message passing and function calls) among those elements. The *physical* level describes the spatial structure of a physical entity (e.g., cables, CPU, memory board, hard disk, printer, and other physical devices) that exists in the hardware domain. Hence, the conceptual level deals with temporal relationships among states of an entity, whereas the functional and physical levels deal with spatial relationships of elements in the entity.

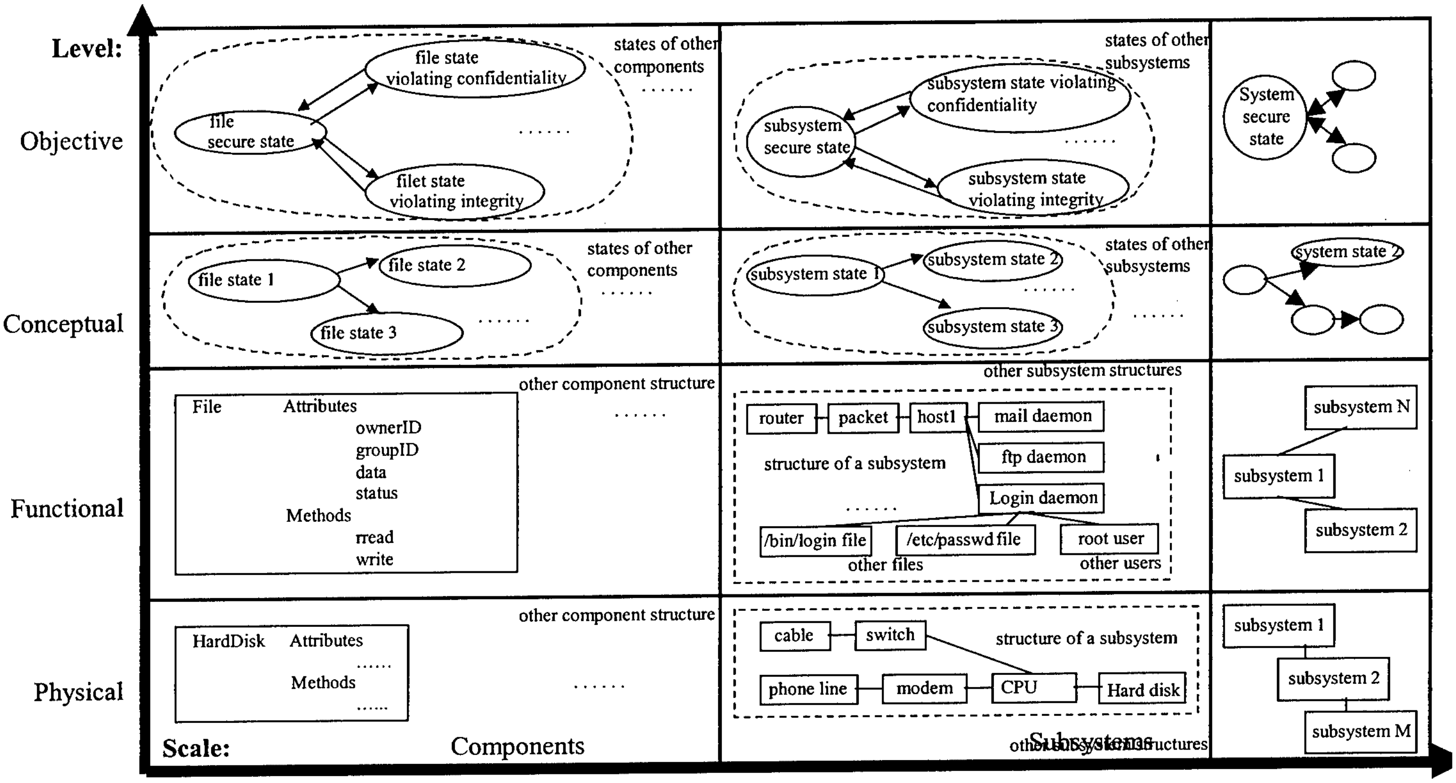


Figure 2-1. A process model of a security-aware computer and network system.

Entities and activities at different levels are associated. For example, the hard disk is a physical entity in the hardware domain. If a file resides on the hard disk, there is an association between the hard disk and the file. A physical activity (e.g., a "read" action) on the hard disk supports a functional activity (e.g., an UNIX "vi" command) on the file. A damage to the hard disk at the physical level changes an attribute of the file at the functional level and the state of the file from accessible to inaccessible at the conceptual level, which further leads to the unavailability of the file at the objective level (see the four cells at the components scale from the physical level to the objective level in Figure 2-1).

Hence, physical entities are associated with functional entities. Attributes of a physical entity are reflected in attributes of associated functional entities. Values of attributes of a functional entity at a given time constitute the state of the functional entity at that time. States and state transitions of the functional entity are modeled at the conceptual level. States or state transitions of the entity are further evaluated into a security posture with respect to security goals.

The physical level is important to detect close-in attacks through hardware subversion. However, most cyber attacks are launched through functional entities in the software domain. Hence, we focus on cyber attacks through functional entities. When building the process model of a computer and network system in our study, we start directly from the functional level.

We now examine the process model in the dimension of scales. At the components scale, entities are individual components of a computer and network system, such as software components (e.g., files, users, programs, and processes). At the subsystem scale, entities are subsystems (e.g., host machines) of the computer and network system. At the system scale, there is a single entity – the computer and network system as a whole (e.g., an enclave administrated independently by an organization).

In the process model there are two types of entities: component-wide entities and system-wide entities including subsystems and the system. Using the object-oriented methodology to represent the process model, an entity is represented by an object with attributes and methods (see Figure 2-2). Without the physical level, attributes of an entity contains mainly:

- entity ID,
- the spatial structure of the entity at the functional level,
- the temporal structure (states and state transitions) of the entity at the conceptual level,
- a security posture (a secure state, a state violating availability, a state violating confidentiality, a state violating integrity, and so on) of the entity at the objective level,
- an activity history (a history of method requests),
- IW (Indications & Warning) values including a composite IW and IWs from individual attack detection techniques, and
- the current state class.

A component-wide entity and a system-wide entity differ only in the spatial structure. The spatial structure of a component-wide entity describes properties of the entity (see the password file in Figure 2-2). The spatial structure of a system-wide entity includes a list of elements and their relationships.

For each entity, we keep a history of activities that have occurred to the entity. For example, the history of activities for a file keeps the time when a method is requested, which user requests the method, which method is requested, and what outcome of the method is (see Figure 2-2).

When a method of the entity is requested, this event is analyzed by various attack detection techniques (discussed later) to produce IW values indicating the likelihood of an attack associated with the event. An IW value of 1 denotes the 100% likelihood, whereas an IW value of 0 denotes the 0% likelihood. A composite IW value is generated by combining IW values from individual attack detection techniques.

The state class is an aggregate representation of the entity state. For example, despite slightly different attribute values, two similar states of the entity may fall into the same state class. In Figure 2-1, as we move from a group of components to a subsystem at the functional level, we shift our attention from properties of individual components to relationships of those components in the subsystem. As we move from the functional level of a component to the conceptual level of the component, we shift our attention from properties of the component at a given time to state transitions of the component over time. For the subsystem at the functional level or the component at the conceptual level, we are interested in spatial relationships of components in the subsystem or temporal relationships of states for the component. The subsystem may contain a large number of components, and the component may have a large number of states over time, which all prevent us from passing all detailed state information of a component for analysis of spatial or temporal relationships. Instead, we pass only the aggregate information of a component to a larger scale or a higher level. In our study, the composite IW value and the current state class together present the aggregate information of a component to be passed on to a larger scale and a higher level. In the process model, information transmission between different scales and different levels involves only aggregate information.

Methods of an entity represent activities that occur to the entity (see Figure 2-2). Attributes of an entity are changed by methods of the entity.

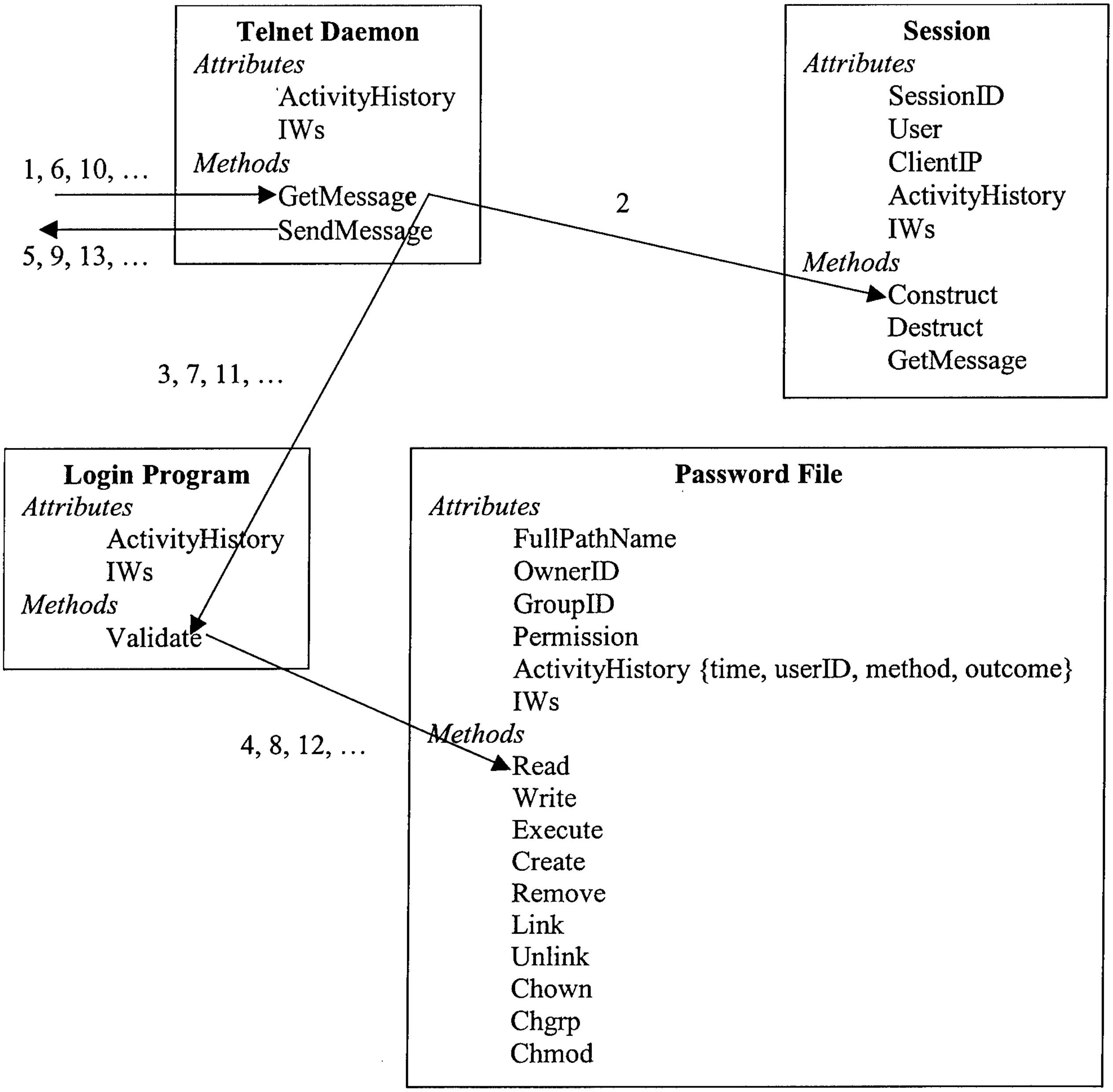


Figure 2-2. Activities in the process model during a password guessing attack.

The process model of a computer and network system is like a security-related mirror reflecting activities that actually occur in the computer and network system, including both normal and malicious activities. Figure 2-2 shows activities in a UNIX-based host machine when an attack is launched to gain the access to this host by guessing a user's password repeatedly until success. The numbers in Figure 2-2 indicate the sequence of activities.

The process model provides a structured, systematic view of activities in a computer and network system. Activities in the computer and network system are captured by sensors which are computer programs monitoring activities of interest to the process model. The process model is constantly updated to reflect activities occurring in the computer and network system. The design of sensors can be based on what attribute information needs to be collected and what methods need to be monitored for each entity in the process model. Information at different scales and different levels of the process model is analyzed by attack detection techniques to provide layered detection mechanisms. Attack isolation and assessment rely on spatial and temporal relationships in the process model. Security policies can be specified for entities in the process model at different scales and different levels. Therefore, the process model provides a systematic framework for design of information assurance.

2.2. Automated System Modeling

To assist in system modeling, we develop the following automated system modeling techniques:

- Object-oriented technique for automated creation of class instances,
- Sensing, probing and testing techniques,

- Discrimination and classification techniques for state information aggregation, and
- Statistical techniques to learn spatial and temporal structures in the process model,

to provide all the information in the process model.

A repository of object classes must be built manually, one class for each kind of entities in the process model (e.g., the file class for all files). Whenever an entity is created (or removed) in a computer and network system with a distinct value of the entity ID, an instance of the corresponding object class is automatically created (or removed).

In the process model, event-driven, object-oriented sensors are placed to monitor method requests to each component at the functional level. A method request to a component results in changes in properties of the component (changes in the spatial structure of the component). From spatial structures of all components in the process model, all other information in the process model is automatically derived.

To obtain the current state class for each entity, state information (values of attributes along with the current event) is classified automatically using a decision tree which is constructed using discrimination and classification techniques (e.g., ID3 for decision tree, and CART - Classification and Regression Trees). We use the current event and current values of attributes as the classification attributes in a decision tree, and use the composite IW value as the target attribute in the decision tree. The decision tree groups similar states into a state class by their IW values.

The spatial structure of a system-wide entity or the temporal structure of each entity in the process model are generated by a set of statistical-based attack detection techniques (discussed later). Those techniques establish a spatial link between two elements or a temporal link between two states using the occurrence frequency of such a link.

The security posture of each entity in the process model is mainly obtained from test results. Our CACS uses probes and tests are used in addition to sensors. A sensor monitors external events (method requests) to an entity. A probe goes inside the entity to obtain values of its attributes. A test initiates actions on the entity (e.g., login as a guest on a host) to reveal the security posture of the entity. Information from sensors is used to generate an initial hypothesis on a possible attack. The hypothesis can be further verified using information from probes and tests. Information from probes and tests is also important to attack isolation and assessment.

2.3. Model-based Attack Detection

Existing attack detection techniques fall into two major categories: anomaly detection and attack signature recognition. Signature recognition techniques store signatures of known attacks, match the subject's observed behavior with those known signatures, and signal an attack when there is a match. Attack signatures have been characterized as strings, event sequences, activity graphs, etc. Petri Nets and rules in expert systems have been used to represent attack signature and perform signature recognition.

Signature recognition techniques cannot detect novel attacks whose signatures are unknown. To overcome this limitation, anomaly detection should be used as a complement to detect novel attacks. Moreover, the repository of attack signatures must be continuously updated to remain useful in changing system configurations, protocols, architectures and environments. It is difficult to update a large repository of attack signatures manually. We develop data mining techniques for the automated learning and update of attack signatures from data of the past attack experience.

For a subject of interest, anomaly detection techniques establish a profile of the subject's normal behavior, compare the observed behavior of the subject with its normal profile, and signal attacks when the subject's observed behavior deviates significantly from its normal profile. Several profiling techniques have been developed based on statistical characteristics [32], predicates in formal logic [33], production rules, and strings [34]. The statistical-based technique, as known from IDES/NIDES and EMERALD [32], has not yet been effective to deal with many correlated attacks involving multiple entities of a computer and network system over time (e.g., multiple sessions). When specifying the expected behavior or security policies, production rules are more practical than predicates, because formal logic is difficult for most security managers to understand and use. Based on the analogy to the human immune system, the string-based profiling technique [34] models the system normal behavior as a set of binary strings [34]. Problems remain as to the robustness of this technique to noises and its expression power.

None of the above profiling techniques address system modeling to a full scale and a full level. Without the full-level, full-scale modeling of a computer and network system, questions would always remain as to whether we were missing any system objects and any aspects of system behavior that might be used in attacks. Our work addresses this problem by providing a conceptual framework for system modeling as well as automated system modeling techniques.

Figure 2-3 and Table 2-1 show two sets of model-based attack detection techniques that are developed in our Cyber Attack Control System (CACS).

Techniques in SET A are applied to individual components at the functional level of the process model. Either Petri nets or rules in expert systems can be used to implement the signature recognition technique. Both Petri nets and rules in expert systems can also be used for the specification technique to specify security policies and detect any violations. Statistical process control (SPC) techniques, such as Shewhart charts for univariate stationary processes, EWMA (Exponentially Weighted Moving Average) for univariate dynamic processes, and Hotelling's T^2 for multivariate processes, have been successfully used in manufacturing industries to detect faults of manufacturing systems by building statistical profiles of system performance. We develop the object-oriented application of SPC techniques as statistical-based anomaly detection techniques for attack detection. Using the Activity History attribute of the password file in Figure 2-2, for instance, EWMA can be applied to measures such as the total number of all method requests per unit time, and the total number of a particular outcome per unit time. Hotelling's T^2 can be applied to measures such as the relative distribution of method requests among different users per unit time, and the relative distribution of different methods per unit time.

The specification of policies and attack signatures is similar across all scales and all levels. In SET B policies and attack signatures involve multiple components, subsystems, or states, whereas in SET A, policies and attack signatures involve single or multiple properties of a single component at the functional level.

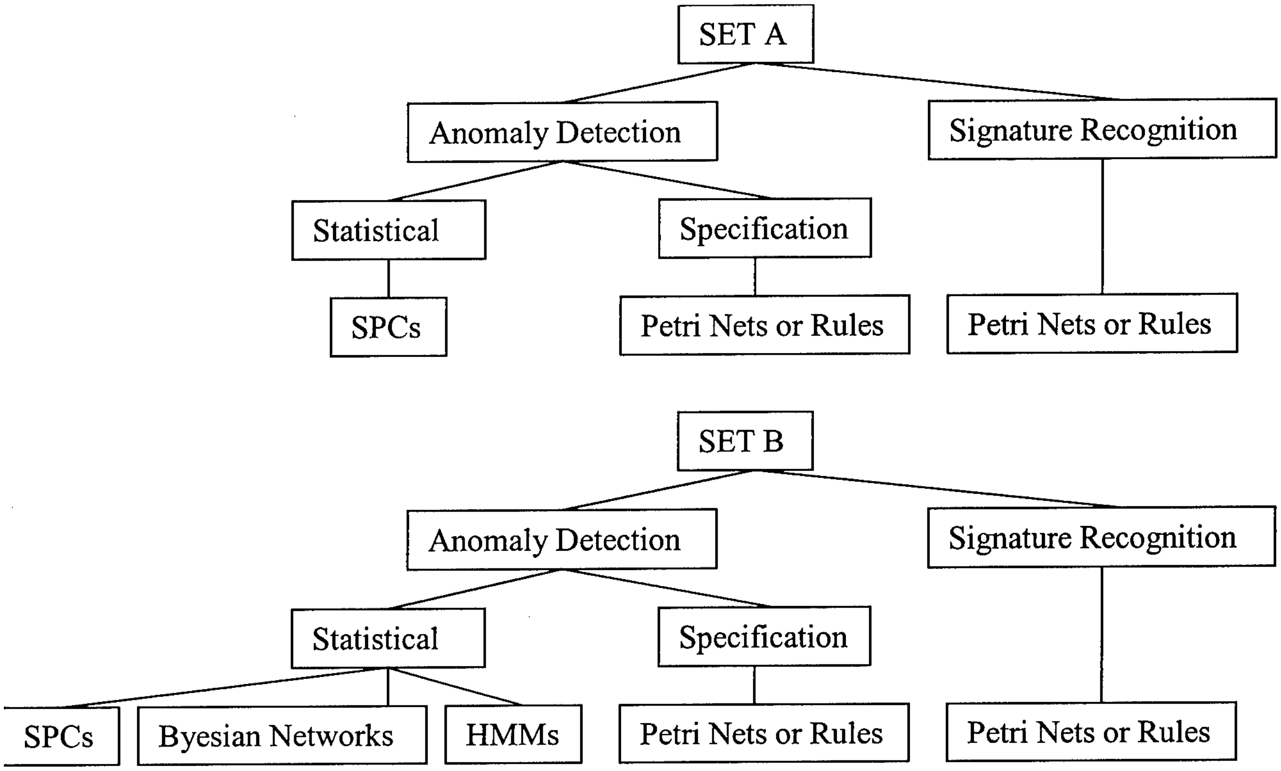


Figure 2-3. Model-based attack detection techniques.

Table 2-1. The fit of model-based attack detection techniques in the process model.

Levels	**Scales**		
	Components	Subsystems	System
Objective	SET B	SET B	SET B
Conceptual	SET B	SET B	SET B
Functional	SET A	SET B	SET B

SET B differs from SET A only in statistical-based anomaly detection techniques. Statistical-based anomaly detection in SET B must correlate aggregate information from multiple components, subsystems or states, and model probabilistic relationships for spatial or temporal structures rather than dealing with individual properties of components as in SET A. The Bayesian network technique [35] and Hidden Markov Models (HMM) technique are added in SET B to fill in the gap left by existing attack detection techniques on the detection of coordinated attacks at larger scales and higher levels. These two techniques learn probabilistic relationships from information of normal activities in a computer and network system, and use learned structural profiles to determine the likelihood of incoming activities to be normal. Bayesian networks and HMMs trained with attack data are unlikely to distinguish between incoming normal activities and incoming attack activities. Due to the heterogeneous nature of attacks, both incoming normal and attack activities would be considered different from attacks used in training.

2.4. Conclusion

The process control approach to cyber attack detection is promising for several reasons. First, it provides a conceptual framework to model a computer and network system in which both normal and attack activities occur. As normal and attack activities are systematically organized, understood and captured in the process model of a computer and network system, information assurance techniques can be designed effectively to cover attacks at various levels and scales of the system for layered, complimentary defense mechanisms.

As the system is modeled from a process control perspective, well-established techniques in process control such as SPC can be applied to overcome problems with existing techniques. Using engineering process control (EPC, such as feedback control) techniques, various information assurance functions (planning, detection, isolation, assessment and reaction) can be tightly interwoven into a feedback control loop for enhanced responsiveness. Research is required to adapt system modeling, SPC and EPC techniques for traditional engineering systems into the problem context of information assurance.

3. Probabilistic network Technique For cyber Attack Detection

3.1. Problem Definition

In this section we describe our attack detection problem, including the data source, problem representation, training data, and testing data.

3.1.1. Data Source

A computer and network system within an organization typically includes a number of host machines (e.g., machines running a UNIX operation system and machines running the Windows NT operating system) and communication links connecting those host machines. Currently two sources of data have been widely used to capture activities in a computer and 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 to capture activities over communication networks. Audit data capture activities occurring on a host machine. In this study, we use audit data from a UNIX-based host machine (specifically a Sun SPARC 10 workstation with the Solaris operating system), and focus on attacks on a host machine that leave trails in audit data. The Solaris operating system from the Sun Microsystems Inc. has a security extension, called the Basic Security Module (BSM). The BSM extension supports the monitoring of activities on a host by recording security-relevant events. Activities on a host machine are captured through a continuous stream of audit events.

3.1.2. Training and Testing Data

In this study, we use a sample of the audit data recording both normal activities and attack activities on host machines with Solaris 2.5. Normal activities and attack activities are simulated to produce these audit data. Normal activities are simulated according to normal activities observed in a real-world computer and network system [5]. A number of attacks are also simulated, including password guessing, use of symbolic links to gain the root privilege, attempts to gain an unauthorized remote access, etc. In the sample, the audit data of normal activities consist of 3019 audit events, and the audit data of attack activities consists of 1223 audit events. We use the first part of the audit data for normal activities as our training data set, and use the remaining audit data for normal activities and attack activities as our testing data set. The training data set consists of 1613 audit events for normal activities. The testing data set consists of 1406 audit events for normal activities and 1223 audit events for attack activities.

3.1.3. Knowledge Representation

An BSM audit record for each event contains a variety of information, including the event type, user ID, group ID, process ID, session ID, the system object accessed, and so on. Studies [28-30] show that types of events in information systems can be used to effectively detect many attacks. Hence, in this study we extract and use the event type from the record of each audit event. There are 284 different types of BSM audit events from Solaris 2.5 that is used to collect the audit data.

For attack detection, we want to build a long-term profile of normal activities, and to compare the activities in the recent past to the long-term norm profile for detecting a significant difference. We define activities in the recent past by opening up an observation window of size N on the continuous steam of audit events to view the last N audit events from the current time *t*:

$E_{t-(N-1)=t-N+1}$, …, E_t, where E stands for event.

At the next time *t+1*, the observation window contains E_{t-N+2}, …, E_{t+1}. In this study, we let N equal to 100, because attack activities produce in average about 100 audit events in the data sample.

We define 284 variables (X_1, …, X_{284}) to represent 284 event types, respectively. We investigate two ways to measure the activities in the recent past and obtain values of the 284 variables from the last N audit events. One way, called the count measurement, is to count the number of a certain event type appearing in the observation window. For example, if there are 10 audit events among the N number of audit events in the observation window fall into the 1st event type, then X_1 has the value of 10. If the 3rd event type does not show in the observation window at all, then X_3 has the value of 0. Another way, called the existence measurement, is to use 1 and 0 to represent the existence and non-existence of a certain event type in the observation window. For the same example, since the 1st event type shows up in the observation window (10 times), then X_1 has the value of 1. Since the 3rd event type does not appear in the observation window at all, then X_3 has the value of 0. Hence, for each observation window we can obtain a vector X including the values of (X_1, …, X_{284}).

3.1.4. Attack Detection Problem

Before training the norm profile, the stream of the 1613 audit events in the training data set is viewed through a moving window, which in turn creates 1514 (1613-99) window slices of audit events for the window size of 100 audit events. There is no window slice created for each of the first 99 audit events, because the current audit event and previous audit events are not sufficient to make up a complete window slice. For each window slice of audit events, a vector of (X_1, ..., X_{284}) is obtained. Hence, from the training data of the 1613 audit events, we obtain 1514 vectors that we use for training a probabilistic network as the norm profile.

For the testing, 1223 audit events for attack activities and 1406 audit events for normal activities are viewed through a moving window, creating totally 2431 window slices (1124 window slices for attack activities and 1307 window slices for normal activities, numbered from No.1-1124 and No. 1125-2431). Each of the first 99 audit events for either attack activities or normal activities does not create a window slice, because the event and previous events together are not sufficient to form a complete window slice of 100 audit events.

For each window slice, a vector of (X_1, ..., X_{284}) is obtained and evaluated against the norm profile to yield the probability that the normal profile supports this vector. The larger the probability is, the more support the vector receives from the normal profile, and the more likely the vector is a part of normal activities. The smaller the probability is, the less likely the vector is normal, and the more likely it is a part of attack activities.

3.2. Learning and Inference of Probabilistic Networks

The probabilistic network technique is developed based on the theory of Bayesian networks. In this section, we briefly review the theory of Bayesian networks. Then we describe the probabilistic network technique.

3.2.1. Bayesian Networks

Bayesian networks are also called Bayesian belief networks, because they are used to represent and infer the beliefs in a set of variables through probabilistic representation and reasoning [36-44]. In a Bayesian network, a set of variables make up the nodes of the Bayesian network, and a set of directed links connect pairs of nodes. A directed link from node X to node Y represents a dependency of Y on X, such as a direct influence of X on Y, a causal relationship of X and Y (X causing Y), a temporal relationship of X and Y (X preceding Y), and so on. X is called a parent of Y. Each variable has a number of states. For example, variable X may represent a particular event and has two states denoting the occurrence or non-occurrence of this event. Each node has a conditional probabilistic table that describes the probabilistic distribution of states for the corresponding variable given the states of its parent nodes. If there is no directed link to the node, the conditional probabilistic table becomes simply a probabilistic table of states for the corresponding variable. A Bayesian network is represented through a directed acyclic graph. No directed cycles are allowed in a Bayesian network.

The information in a Bayesian network provides a joint probability distribution of all the variables $X_1, \ldots, X_n$ in the Bayesian network. If we label these variables in an order that is consistent with their directed links in the Bayesian network,

$$\begin{aligned}
&P(X_1, \ldots, X_n) \\
&= P(X_n \mid X_{n-1}, \ldots, X_1) * P(X_{n-1}, \ldots, X_1) \\
&= \prod_{i=1}^{n} P(X_i \mid X_{i-1}, \ldots, X_1) \qquad (1) \\
&= \prod_{i=1}^{n} P(X_i \mid parents(X_i))
\end{aligned}$$

Hence, the conditional probability tables in a Bayesian network provide a decomposed representation of the joint probability table for all the variables in the Bayesian network. From this joint probability table, we can derive the probability of any state involving any combination of the variables. Since many pairs of variables are conditionally independent (no directed links between them in a Bayesian network), the joint probability table is simplified into the conditional probability tables in a Bayesian network.

In many real-world problems, we are not interested in the direction of dependence but the strength of dependence. For example, in this study we are interested in how likely various audit events correlate (co-occur) in a moving window during normal activities (for a norm profile) or attack activities (for an attack profile). We may consider the causal or temporal relationship of audit events as the direction of dependence between audit events. However, since one audit event may cause or precede another audit event in some activities or vice versa in other activities, a directed cycle between these two audit events exists, which is not allowed in a Bayesian network. Therefore, in this study we consider the strength of dependence without the direction, where the dependence between audit events is characterized by the co-occurrence of audit events. We also need to develop the learning and inference algorithms for a probabilistic network with undirected links.

3.2.2. Probabilistic Networks with Undirected Links

To build a probabilistic network with undirected links, we go back to the joint probability table of the variables where no direction is implied. Then we simply the joint probability table into a probabilistic network with undirected links. An undirected link is placed between a pair of variables if they have a strong dependency. A joint probability table is created for a group of variables which are fully dependent on each other. For example, if variables A, B, C and D are fully linked to form a fully connected graph, a joint probability table is created for the relation of variables A, B, C and D. Hence, the joint probability table of all the variables are simplified into a probabilistic network with undirected links, or in other terms, a set of smaller joint probability tables for all the nodes and relations.

3.2.3. Learning of Joint Probability Tables

The joint probability table for a node involving a single variable X is estimated from the training data as follows [40].

$$P(X = i) = \frac{N_{X = i}}{N} \tag{2}$$

where

N is the total number of observations in the training data,

$N_{X=i}$ is the number of observations with X in state *i*, and

P denotes the probability that X is in state *i*.

The joint probability table for a relation involving more than one variable such as X_1, …, X_k is estimated from the training data as follows [40].

$$P(X_1 = i_1, \ldots\ldots, X_k = i_k) = \frac{N_{X_1 = i_1, \ldots\ldots X_k = i_k}}{N} \tag{3}$$

where

N is the total number of observations in training data,

$N_{X=i}$ is the number of observations with X_1 in state i_1, …, and X_k in state i_k, and

P denotes the probability that X_1 is in state i_1, …, and X_k is in state i_k.

3.2.4. Learning of the Structure of a Probabilistic Network

The structure of a probabilistic network consists of nodes and undirected links between nodes. Given a set of variables and a number of their observations in the training data, nodes in a probabilistic network are constructed by creating one node for each variable. Undirected links between variable nodes are constructed by learning the dependence between variables from the training data.

The structure is learned through two phases. Phase I is to learn an initial structure of the probabilistic network. Phase II uses an iterative procedure of search and scoring to find the optimal structure of the probabilistic network.

To construct the initial structure of the probabilistic network in Phase I, we use the chi-square test of independence to determine the strength of dependence Q between a pair of variables X_i and X_j as follows.

$$X^2 = \sum_{k=1}^{K}\sum_{l=1}^{L}\frac{(N_{X_i = k,\, X_j = l} - N * P_{kl})^2}{N * P_{kl}} \tag{4}$$

$$P_{kl} = P_{k.} * P_{.l} \tag{5}$$

$$P_{k.} = P(X_i = k) = \frac{N_{X_i = k}}{N} \tag{6}$$

$$P_{.l} = P(X_j = l) = \frac{N_{X_j = l}}{N} \tag{7}$$

$$Q = [2^{\frac{d}{2}}\Gamma(\frac{d}{2})]^{-1}\int_{X^2}^{\infty} t^{\frac{d}{2}-1} e^{\frac{t}{2}} dt \tag{8}$$

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt \tag{9}$$

where

X^2 denotes the chi-square statistic,

N is the total number of observations,

$N_{Xi=k,\ Xj=l}$ is the number of observations with X_i in state k and X_j in state l,

$N_{Xi=k}$ is the number of observations with X_i in state k,

$N_{Xj=l}$ is the number of observations with X_j in state l, and

d denotes the degree of freedom which is equal to (m-1)*(n-1), m is the number of states that X_i has, and n is the number of states that X_j has.

The larger the X^2 value is, the larger difference exists between the observed frequency and the expected frequency of X_i in state *k* and X_j in state *l* (expected under the hypothesis of X_i and X_j are independent). The larger the X^2 value is, the smaller the Q value is, the less likely X_i and X_j are independent, and the stronger dependence exists between X_i and X_j.

After computing the strength of dependence between every pair of the variables, we rank the pairs of variables by the ascending order of Q values. The pair of the variables that is ranked first has the strongest strength of dependence. We establish the links between the first B pairs of the variables, with a constraint that a maximum of D links can be established from a variable. The parameters B and D are introduced to control the complexity of the initial structure of the probabilistic network, so that links are established for only strongly dependent pairs of variables, while limiting the number of links from each variable. For example, considering the following rank of variable pairs with Q values.

1. (X1, X2) with Q = 0.10
2. (X2, X3) with Q = 0.15

3. (X2, X4) with Q = 0.20

4. (X3, X4) with Q = 0.25

5. (X2, X5) with Q = 0.30

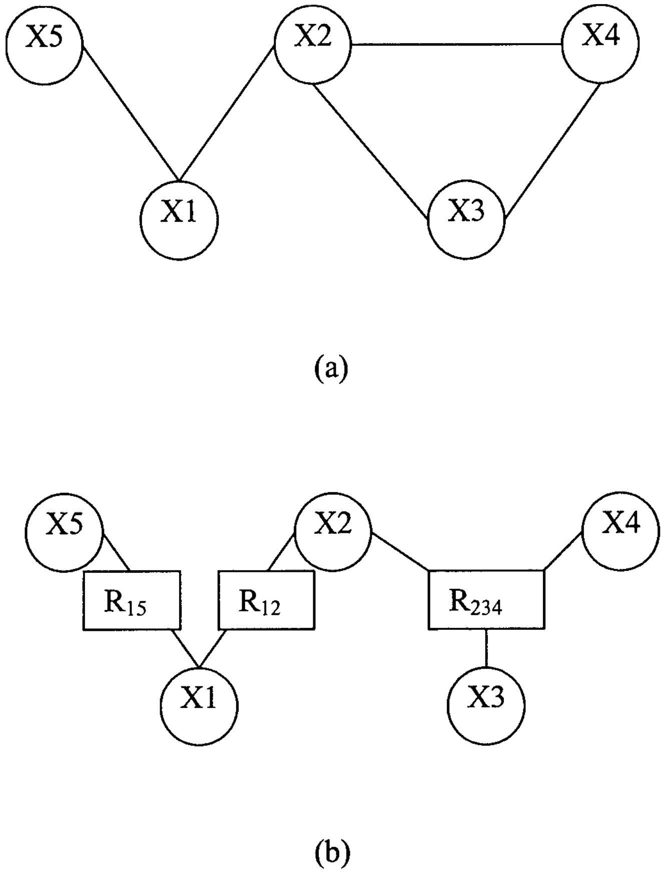
6. (X5, X1) with Q = 0.35

7. (X5, X4) with Q = 0.40.

If B is set to 5 and D is set to 3, the initial structure of the probabilistic network looks like the graph in Figure 3-1. There is no link between the pair of (X2, X5), because at the maximum three links are allowed for variable X2. There is no link between the pair of (X5, X4), because the total number of links is limited up to 5.

After obtaining the initial structure of the probabilistic network, we search and find all fully connected graphs that exist in the initial structure of the probabilistic network, from the largest fully connected graphs to the smallest fully connected graphs which include only two nodes. For each fully connected graph, we establish a relation node. For example, in Figure 3-1 the largest fully connected graph has three nodes, X_2, X_3, and X_4. A relation node R_{234} is established. Next, we find two fully connected graphs, each of which has only two nodes, and establish two relation nodes, R_{12} and R_{15}. Then we obtain a joint probability table for each variable node using formula (2) and a joint probability table for each relation node using formula (3). With the structure (nodes and links) and the joint probability tables, the initial probabilistic network is completed.

Phase II is based on the minimum description length principle to search for the optimal structure of the probabilistic network that best supports the training data [40]. The initial probabilistic network is evaluated to produce the following score [40].



(a)

(b)

Figure 3-1. An example of the initial structure of a probabilistic network and relations among variable nodes.

$$Score = A * DL(TD \mid PN) + C * DL(PN) \tag{10}$$

where

TD stands for the training data,

PN stands for the probabilistic network,

DL(TD | PN) denotes the description length of the training data, given the probabilistic network, which measures the support (fitness) of the probabilistic network to the training data (the better the support, the smaller the description length), DL(PN) represents the description length of the probabilistic network which measures the complexity of the probabilistic network,

A is a weighting factor on the measure of how well the probabilistic network supports the training data, and

C is a weighting factor on the complexity measure of the probabilistic network.

In general, the more complex the probabilistic network is, the better support (fitness) the probabilistic network can provide to the training data (including noises in the training data). The probabilistic network should closely fit the training data, while preventing the overfitting of the probabilistic network to the training data. We use a following measure of support as the description length of the training data given the probabilistic network.

$$DL(TD \mid PN) = \sum_{O_i \in TD} -\log_{10}(P(O_i \mid PN)) \qquad (11)$$

where O_i is an observation in the training data, and $P(O_i \mid PN)$ is the probability that O_i is supported by the probabilistic network. $P(O_i \mid PN)$ is determined by the equations in the next section. The smaller the DL(TD | PN) value is, the better support the probabilistic network provides to the training data.

The overfitting problem is often controlled using the Minimum Description Length (MDL) principle [40] to minimize the complexity of a model. The complexity of a probabilistic network depends on three factors: the number of variable nodes, the number of relation nodes, and the size of the joint probability table at each node. If we count the number of cells in each joint probability table, the sum of the counts over all the joint probability tables accounts for all the three factors. Hence, we let the sum of the counts be the description length (DL) of the probabilistic network.

In general, a more complex probabilistic network leads to a better support of the probabilistic network to the training data. Our goal is to look for a probabilistic network that minimizes the weighted sum of DL(DN | PN) and DL (PN) in formula (10), subject to the inherently inverse relationship between DL(DN | PN) and DL (PN).

We conduct a heuristic search for an alternative probabilistic network with a smaller score (a better probabilistic network) than the initial probabilistic network. The only way to change the initial probabilistic network is to change its links, since the number of variable nodes in the probabilistic network is fixed and the joint probability tables are determined from the training data. In the heuristic search, we explore the one-link change for all possible pairs of variable nodes in the initial probabilistic network. If there is a link between a pair of variable nodes, a one-link change for the pair of variable nodes is to remove the link. If there is no link between the pair of variable nodes, a one-link change is to add a link between the pair of variable nodes. For a probabilistic network with *n* nodes, there are n*(n-1)/2 possible pairs and thus the same number of possible one-link change. Each one-link change leads to a new structure of the probabilistic network. For each new structure, we compute the joint probability tables and the evaluation score in formula (10). We then compare the evaluation scores for the n*(n-1)/2 probabilistic networks, and select one with the smallest evaluation score. Then the next round of search and scoring continues with the selected probabilistic network, until the evaluation score no longer improves (reaching a global or local minimum).

3.2.5. Inference Algorithm

The support of the probabilistic network to an observation O_i in the training data in formula (11), P(Oi | PN), is determined as follows. The observation O_i is a vector of (X_1, …, X_n).

$$P(X_1,\ldots,X_n) = P(X_1) * P(X_2 \mid X_1) * \ldots * P(X_n \mid X_{n-1},\ldots,X_1) \qquad (12)$$

The conditional probability $P(X_k \mid X_{k-1}, \ldots, X_1)$ represents the updated joint probability table of X_k based on new evidences on $X_{k-1}, \ldots, X_1$, where k = 2, …, n. Each $P(X_k \mid X_{k-1}, \ldots, X_1)$ term in formula (13) is determined through probability update and propagation as shown below.

If Xi and Xj are linked through relation R, P(Xj | Xi) is determined by the following formulas which are similar to those for probability updating in a Bayesian network [24].

$$P^{(m)}(R) = normalize\left[P^{(m-1)}(R)\frac{P^{(k)}(X_i)}{P^{(0)}(X_i)}\right] \qquad (13)$$

$$P^{(n)}(X_j) = normalize\left[P^{(n-1)}(X_j)\frac{m\arg inalize\left[P^{(m)}(R)\right]}{P^{(0)}(X_j)}\right] \qquad (14)$$

where

R denotes a relation,

X_j and X_i are variable nodes involved in relation R,

$P^{(m)}(R)$ denotes the updated joint probability table of relation R based on the new evidence on X_i,

$P^{(m-1)}(R)$ denotes the updated joint probability table of relation R based on another new evidence which is also involved in R,

$P^{(0)}(R)$ is the prior state distribution of R before any update,

$P^{(n)}(X_j)$ denotes the updated joint probability table of X_j based on the updated joint probability table $P^{(m)}(R)$ if there is any update of R, or based on the prior joint probability table $P^{(0)}(R)$ if there is no update of R,

$P^{(n-1)}(X_j)$ denotes the updated joint probability table of X_j based on another relation which also involves $X_{j,}$ and

$P^{(0)}(X_j)$ is the prior state distribution of X_j before any update.

Formula (13) is used to update the joint probability table of relation R from a new evidence on X_i. Formula (14) is used to update the joint probability table of X_j from the updated joint probability table of relation R.

Figure 3-2 shows a probability network consisting of four variable nodes (X_1, X_2, X_3, and X_4) and two relation nodes (R_{12} and R_{234}). The initial joint probability tables are given in part (a) of Figure 3-2. Part (b) of Figure 3-2 shows the computations for the probability update and propagation. Given a new evidence on X_1, the updated joint probability table of X_2 is determined by first updating the joint probability table of relation R_{12} using formula (13).

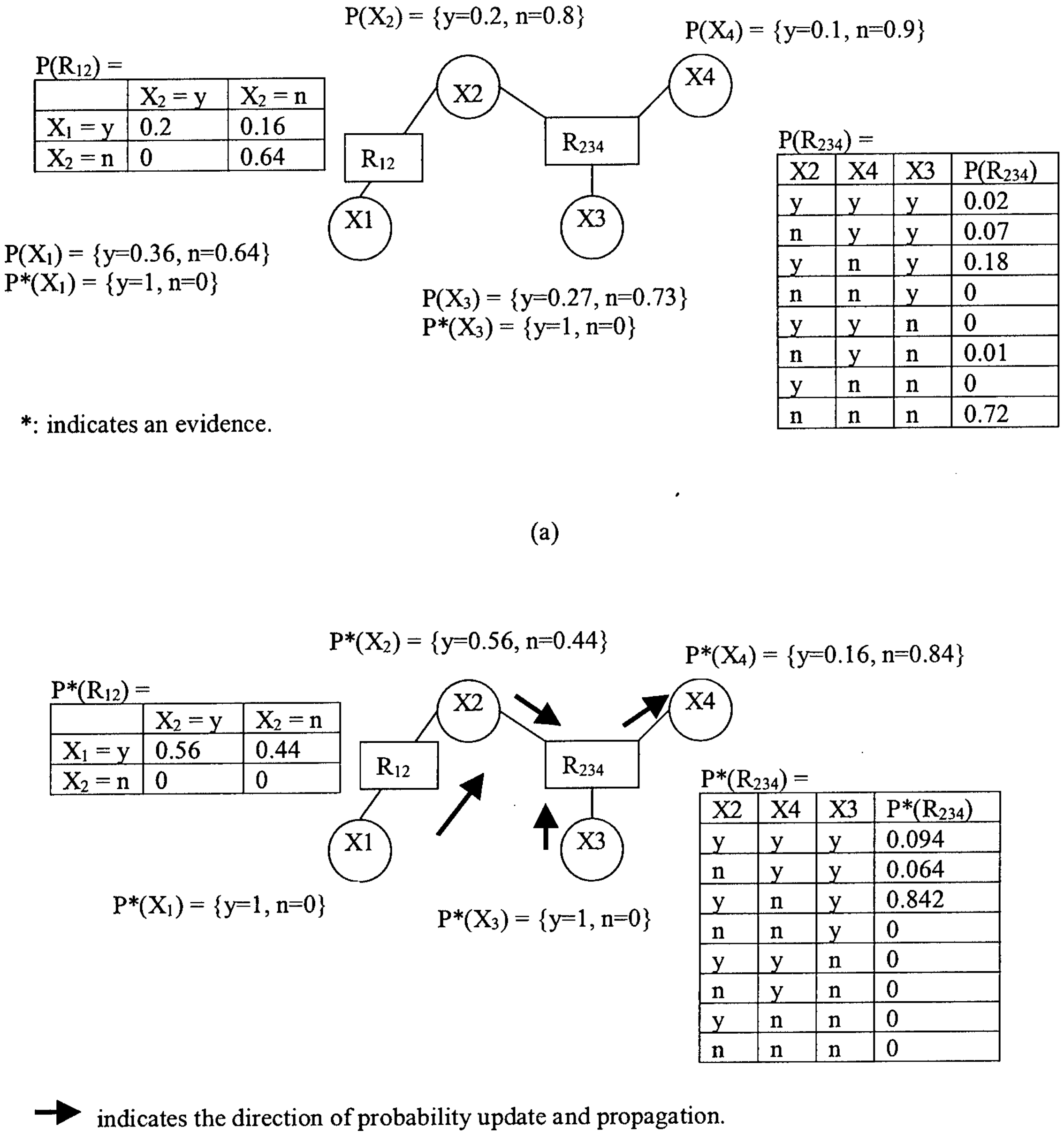
$$P(X_1 = y, X_2 = y) = 0.2 * \frac{1}{0.36} = 0.56$$

$$P(X_1 = y, X_2 = n) = 0.16 * \frac{1}{0.36} = 0.44$$

$$P(X_1 = n, X_2 = y) = 0 * \frac{0}{0.64} = 0$$

$$P(X_1 = n, X_2 = n) = 0.64 * \frac{0}{0.64} = 0$$

Since the sum of the above four values is 1, the normalization is not needed. We then update the joint probability table of X2 by marginalizing X1 out of the updated joint probability table of relation R12 as follows.



$P(X_2) = \{y{=}0.2, n{=}0.8\}$

$P(X_4) = \{y{=}0.1, n{=}0.9\}$

$P(R_{12})$ =

	$X_2 = y$	$X_2 = n$
$X_1 = y$	0.2	0.16
$X_2 = n$	0	0.64

$P(X_1) = \{y{=}0.36, n{=}0.64\}$
$P^*(X_1) = \{y{=}1, n{=}0\}$

$P(X_3) = \{y{=}0.27, n{=}0.73\}$
$P^*(X_3) = \{y{=}1, n{=}0\}$

$P(R_{234})$ =

X2	X4	X3	$P(R_{234})$
y	y	y	0.02
n	y	y	0.07
y	n	y	0.18
n	n	y	0
y	y	n	0
n	y	n	0.01
y	n	n	0
n	n	n	0.72

*: indicates an evidence.

(a)

$P^*(X_2) = \{y{=}0.56, n{=}0.44\}$

$P^*(X_4) = \{y{=}0.16, n{=}0.84\}$

$P^*(R_{12})$ =

	$X_2 = y$	$X_2 = n$
$X_1 = y$	0.56	0.44
$X_2 = n$	0	0

$P^*(X_1) = \{y{=}1, n{=}0\}$

$P^*(X_3) = \{y{=}1, n{=}0\}$

$P^*(R_{234})$ =

X2	X4	X3	$P^*(R_{234})$
y	y	y	0.094
n	y	y	0.064
y	n	y	0.842
n	n	y	0
y	y	n	0
n	y	n	0
y	n	n	0
n	n	n	0

➡ indicates the direction of probability update and propagation.

(b)

Figure 3-2. Probability update and propagation in a probabilistic network.

$$P(X_2 = y) = 0.56 + 0 = 0.56$$
$$P(X_2 = n) = 0.44 + 0 = 0.44$$

For another example, $P(X_4 \mid X_3, X_1)$ requires the updated joint probability table of X4 given the new evidences on X_1 and X_3. To determine $P(X_4 \mid X_3, X_1)$, we update the joint probability table of R_{234} using the updated evidence on X_2. The updated joint probability table of R_{234} is again updated using the new evidence on X_3. By marginalizing X_2 and X_3 out of the twice-updated joint probability table of R_{234}, we obtain the updated joint probability table of X_4 as shown in part (b) of Figure 3-2.

3.3. Probabilistic Networks For Cyber Attack Detection

As discussed in section 3.1, we obtain two sets of $(X_1, \ldots, X_{284})$ vectors from the training data set: one set for the count measurement, and another set for the existence measurement. The learning and inference algorithms of the probabilistic network apply to variables with a finite number of discrete values as states. In the set of the $(X_1, \ldots, X_{284})$ vectors with the existence measurement, each variable has two states: existence (with the value of 1) and non-existence (with the value of 0). In the set of the $(X_1, \ldots, X_{284})$ vectors with the count measurement, each variable takes a count number. Although a count is not exactly a continuous value, it is a discrete value with possibly no upper limit. For our attack detection problem, the upper limit of a count is equal to the size of the moving window. That is, the largest count for a certain type of audit event is equal to the total number of audit events in the moving window, when all audit events in the moving window are the same type. If we take each possible value of the count as one state of a variable, the variable may end up with possibly too many states. Since many of such states are just slightly different and do not always appear, it would be a waste of computer resources if we take each possible count as one state. Such a waste of computer resources is not desirable, especially when we deal with a large-scale problem.

Therefore, we want to transform each set of $(X_1, \ldots, X_{284})$ vectors with the count measurement so that variables take a reasonable number of discrete states. A variety of methods exist to discretize continuous values [40-44]. Those methods generally fall in two categories. One category of the methods determine a set of dividing points to yield the discrete segments of a continuous variable by evaluating whether this set of dividing points lead to a better fitted model to the training data. Another category of the methods determine a set of dividing points using a fixed formula, leading to either a linear segmentation or non-linear segmentation. In this study, we use the following non-linear segmentation formula to simplify the computation in learning:

$$\begin{aligned} State &= 1 + \log_2 Count \quad && \mathit{if}\ Count > 0 \\ &= 0 && \text{if Count} = 0 \end{aligned} \tag{15}$$

The two kinds of measurements (count and existence) produce two different sets of (X_1, …, X_{284}) vectors to train two different probabilistic networks, respectively. Each probabilistic network had 284 variable nodes. The parameter B for the maximum number of links in a probabilistic network was arbitrarily set to 70. The parameter D for the maximum number of links from a variable node was arbitrarily set to 3. The weighting factor for the entropy of the training data in a probabilistic network was arbitrarily set to 9. The weighting factor for the description length of a probabilistic network was arbitrarily set to 1.

The testing data contain the audit data for both attack activities and normal activities. When a vector of (X_1, …, X_{284}) from a window slice of the testing data is presented to a probabilistic network trained with the audit data of normal activities (the norm-based probabilistic network), the probability that this vector is supported by the norm-based probabilistic network is determined by formula (12). The larger the probability is, the more likely activities in the window slice are normal.

It is possible that a vector of (X_1, …, X_{284}) from the testing data presents a state of a variable and/or a state of a relation among several variables which are not encountered during the training and are thus not covered by the joint probability tables of the trained probabilistic network. While using formula (12) to infer the support probability of the trained probabilistic network to this vector of (X_1, …, X_{284}), we assign a state, which is not available in the trained probabilistic network, a probability of 0.00001 which is close to zero and is much smaller than any existing probabilities in the joint probability tables of the trained probabilistic network.

We do not assign the probability of zero to such states, because the probability of zero would make the final result from formula (12) become zero. This would make it impossible to distinguish normal activities with noises from attack activities, since noises in normal activities may introduce new states. The amount of new states from noises in normal activities is expected much less than the amount of new states from attack activities. By assigning a small probability rather than zero to the new states, the smaller effect of noises in normal activities on the final result of formula (12) becomes distinguishable from the larger effect of attack activities on the final result of formula (12).

3.4. Results And Discussions

During the testing, we compute the probabilities that the two trained probabilistic networks support the testing data. Table 3-1 summarizes the statistics of the testing results. For each probability network, we compute the minimum, maximum, average and standard deviation of the probabilities that are produced for each kind of the testing data (the testing data of normal activities – normal data, and the testing data of attack activities – attack data).

Table 3-1. The statistics of the testing results.

Training	Measurement	Testing	Minimum	Maximum	Average	Standard Deviatio
Normal data	Existence	Normal data	4.89E-05	2.64E-01	1.29E-01	1.08E-01
Normal data	Existence	Attack data	3.46E-113	1.48E-21	7.90E-24	1.08E-22
Normal data	Count	Normal data	6.04E-18	1.26E-02	1.96E-03	2.24E-03
Normal data	Count	Attack data	4.28E-127	1.09E-29	8.74E-32	8.44E-31
Attack data	Existence	Normal data	9.51E-47	3.00E-33	5.28E-34	1.14E-33
Attack data	Existence	Attack data	4.84E-82	4.78E-05	1.01E-07	2.02E-06
Attack data	Count	Normal data	8.14E-46	7.13E-26	2.83E-28	4.03E-27
Attack data	Count	Attack data	1.56E-82	6.60E-08	4.80E-10	4.72E-09

The two norm-based probabilistic networks, which are trained with the normal data (audit data of normal activities) for two kinds of measurements respectively, produce much larger probability values for the normal data than the attack data during the testing. As shown in formula 2, the probabilities for event numbers 1125-2431 (the normal data) are much larger than the probabilities for event numbers 1-1124 (the attack data). A larger probability means more support of the norm-based probabilistic networks to the data. For the existence measurement, there exists a huge gap between the minimum probability for the normal testing data (4.89E-05) and the maximum probability for the attack testing data (1.48E-21). For the count measurement, there also exists a huge gap between the minimum probability for the normal testing data (6.04E-18) and the maximum probability for the attack testing data (1.09E-29).

These results indicate that for both kinds of measurement we are able to clearly distinguish the normal activities from the attack activities during the testing, using any probability value in the gaps as the decision threshold. If the probability of the testing data from a moving window is greater than the decision threshold, the activities in the moving window are classified as normal. Otherwise, the activities in the moving window are classified as attack.

In overall, the probability for the count measurement on the testing data within a moving window is smaller than the probability for the existence measurement on the same testing data for two reasons. First, we use the small probability value of 0.00001 for a new state that appeared in the testing but did not show in the training, when we use formula (12) to calculate the final probability of a vector from the moving window. Second, the count measurement create more of such new states.

In summary, the norm-based probabilistic networks demonstrate the promising performance in detecting attack activities during the testing, regardless of which measurement method is used. The results from this study encourage the further investigation of the probabilistic network technique and its application to attack detection.

Note that the results of this study are obtained using an arbitrary set of parameters for the probabilistic networks, such as parameter B for the maximum number of links in a probabilistic network, parameter D for the maximum number of links from a variable node, the weighting factor for the entropy of the training data in a probabilistic network, and the weighting factor for the description length of a probabilistic network. In general, the larger parameters B and D are, the better a probabilistic network can fit the training data. However, a better fit does not necessarily lead to a better testing performance due to the over-fitting problem. Just as many training parameters in an artificial neural network must be determined empirically for a particular application [40], these parameters for a probabilistic network can be determined empirically using some training and testing data.

References

[1] Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction.* New York, NY: North-Holland.

[2] Ye, N. (1996). A hierarchy of system-oriented knowledge for diagnosis of manufacturing system faults. *Information and System Engineering,* 2(2), 79-103.

[3] T. Escamilla. *Intrusion Detection: Network Security beyond the Firewall.* New York: John Wiley & Sons, 1998.

[4] H. Debar, M. Dacier, and A. Wespi. "Towards a taxonomy of intrusion-detection systems," Computer Networks, 31, pp. 805-822, 1999.

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

[6] T. Bass. "Intrusion detection systems and multi-sensor data fusion," Communications of the ACM, 43(4), pp. 99-105, April 2000.

[7] U. Lindqvist, and P. A. Porras. "Detecting computer and network misuse through the production-based expert system toolset (P-BEST)," In Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE, Oakland, CA, May 1999.

[8] P. A. Porras, and P. G. Neumann. "EMERALD: Event monitoring enabling responses to anomalous live disturbances," In Proceedings of NISSC, October 1997.

[9] P. G. Neumann, and P. A. Porras. "Experience with EMERALD to date," In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, April 1999, pp. 73-80.

[10] W. Lee, and S. J. Stolfo. "Data mining approaches for intrusion detection," In Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas, January 1998.

[11] W. Lee, S. J. Stolfo, and K. W. Mok. "A data mining framework for building intrusion detection models," In Proceedings of the 1999 IEEE Symposium on Security & Privacy, May 1999.

[12] W. Lee, S. J. Stolfo, and K. W. Mok. "Mining audit data to build intrusion detection models," In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, NY, August 1998.

[13] W. Lee, S. J. Stolfo, and K. W. Mok. "Mining in a data-flow environment: Experience in network intrusion detection," In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99), San Diego, August 1999.

[14] G. Vigna, and R. Kemmerer. "NetStat: A network-based intrusion detection appoach." In Proceedings of the 14th Annual Computer Security Applications Conference, Scottsdale, Arizona, December 1998, http://www.cs.ucsb.edu/~kemm/netstat.html/.

[15] G. Vigna, S. T. Eckmann, and R. A. Kemmerer. "The STAT tool suit," In *Proceedings of the DARPA Information Survivability Conference and Exposition.* Los Alamitos, CA: IEE Computer Society, pp. 46-55, January, 2000.

[16] S. Kumar. Classification and Detection of Computer Intrusions. Ph.D. Dissertation, Department of Computer Science, Purdue University, West Lafayette, Indiana, 1995.

[17] C. Ko, G. Fink, and K. Levitt. "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, 1997.

[18] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. "GrIDS – A graph-based intrusion detection system for large networks," In Proceedings of the 19th National Information Systems Security Conference, October 1996.

[19] T. Bowen, D. Chee, M. Segal, R. Sekar, T. Shanbhag, and P. Uppuluri. "Building survivable systems: An integrated approach based on intrusion detection and damage containment," *In Proceedings of the DARPA Information Survivability Conference and Exposition, Volume II.* Los Alamitos, CA: IEE Computer Society, pp. 84-99, January, 2000.

[20] D. E. Denning. "An intrusion-detection model," IEEE Transactions on Software Engineering, SE-13(2), pp. 222-232, February 1987.

[21] D. Anderson, T. Frivold, and A. Valdes. *Next-generation Intrusion Detection Expert System (NIDES): A Summary.* Technical Report SRI-CSL-97-07. Menlo Park, CA: SRI International, May, 1995.

[22] H. S. Javitz, and A. Valdes. "The SRI statistical anomaly detector." In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy.* May 1991.

[23] H. S. Javitz, and A. Valdes. *The NIDES Statistical Component Description of Justification.* Technical Report A010. Menlo Park, CA: SRI International, March, 1994.

[24] Y. Jou, F. Gong, C. Sargor, X. Wu, S. Wu, H. Chang, and F. Wang. "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.* Los Alamitos, CA: IEE Computer Society, pp. 69-83, 2000.

[25] W. DuMouchel, M. Schonlau. "A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities," In Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics.

[26] H. Debar, M. Becker, D. Siboni. "A neural network component for an intrusion detection system," In Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May 1992, pp. 240-250.

[27] A. K. Ghosh, A. Schwatzbard, and M. Shatz. "Learning program behavior profiles for intrusion detection." In *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring,* Santa Clara, California, April, 1999, http://www.rstcorp.com/~anup/.

[28] S. Forrest, S. A. Hofmeyr, and A. Somayaji. "Computer immunology." *Communications of the ACM,* 40(10), pp. 88-96, October, 1997.

[29] H. Debar, M. Dacier, M. Nassehi, and A. Wespi. "Fixed vs. variable-length patterns for detecting suspicious process behavior," In Proceedings of the 5th European

Symposium on research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18, 1998, pp. 1-15.

[30] C. Warrender, S. Forrest, and B. Pearlmutter. "Detecting intrusions using system calls: Alternative data models," In Proceedings of the 1999 IEEE Symposium on Security and Privacy, pp. 133-145.

[31] Ye, N., Hosmer, C., Giordano, J., and Feldman, J. (1998). Critical information infrastructure protection through process modeling and model-based information fusion. In *Proceedings of the Information Survivability Workshop 1998,* pp. 197-201.

[32] http://www.csl.sri.com/intrusion.html.

[33] http://seclab.cs.ucdavis.edu.

[34] Forrest, S., Hofmeyr, S. A., and Somayaji, A. (1997). Computer immunology. *Communications of the ACM,* 40(10), October, 88-96.

[35] Ye, N., Giordano, J., Feldman, J., and Zhong, Q. (1998). "Information fusion techniques for network intrusion detection". In *Proceedings of the 1998 Information Technology Conference*, pp. 117-120.

[36] F. V. Jensen. *The Instruction to Bayesian Networks*. New York: Springer, 1996.

[37] J. Suzuki. "Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique." *IEICE Transactions on Information and Systems,* Vol. E82-D, No. 2, pp. 356-367, February, 1999.

[38] J. Liu, and M. C. Desmarais. "A method of learning implication network from empirical data: Algorithm and Monte-Carlo simulation-based validation." *IEEE*

Transactions on Knowledge and Data Engineering, 9(6), November/December, pp. 990-1004, 1997.

[39] W. L. Buntine. "Operations for learning with graphical models." *J. of Artificial intelligence Research,* 2, pp. 159-225, 1994. Http://www.cs.washington.edu/research/jair/home.html.

[40] T. M. Mitchell. *Machine learning.* Boston: McGraw-Hill, 1997.

[41] J. Cheng, D. Bell, and W. Liu. "Learning Bayesian networks from data: An efficient approach based on information theory." Http://www.cs.ualberta.ca/~jcheng/lab.htm.

[42] R. Hofmann, and V. Tresp. "Discovering structure in continuous variables using Bayesian networks". In *Advances in Neural Information Processing System* 8, Cambridge MA: MIT Press, 1996. Http://www7.informatik.tu-muenchen.de/~hofmannr.

[43] U. M. Fayyad, and K. B. Irani. "Multi-interval Discretization of continous-valued attributions for classification learning." In R. Bajcsy (Ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence.* Morgan-Kaufmann, pp. 1022-1027.

[44] U. M. Fayyad, and K. B. Irani. "On the handling of continuous-Valued attributes in decision tree generation" *Machine learning,* 8(1), pp. 87-102, January, 1992.

☆U.S. GOVERNMENT PRINTING OFFICE: 2001-710-038-10184

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.