A new model for monitoring intrusion based on Petri Nets

H. Arafat Ali
Information Systems Department, Faculty of Computers and Information System, Mansoura University, Mansoura, Egypt

Keywords
Computer networks, Security, Monitoring, Management, Model

Abstract
Computers and the information they process are critical to many organizations' ability to perform their mission and business functions. It therefore makes sense that executives view computer security as a management issue and seek to protect their data which are stored in these computers. Presents a main objective of introducing a modeling design and verification of the monitoring part of common intrusion detection framework (CIDF) using Petri Nets. To enhance the security of a system by monitoring system activity and detecting a typical behavior, statistical unusual behavior must be found in the observation of the system. Such a monitoring system will be capable of detecting intrusion that could not be detected by any other means. These systems that do collect audit data are the only way to build a real secure system which is the most important part of the network. Presents a proposed model of the monitoring part of the CIDF based on Petri Nets modeling technique. Tests the proposed model using the Petri Nets properties.

Information Management & Computer Security
9/4 [2001] 175–182

[ISSN 0968-5227]

Introduction

As the number of businesses and government agencies connecting to the Internet continues to increase, the demand for information protection – point of security guarding a private network from intrusion – has created a demand for reliable models which help to monitor them and then take the suitable action (Puketza *et al*., 1996; Compaq Computer Corporation, 1998). Protecting an information system is, in large measure, based on providing essential services while isolating the information system from risks that would damage the system or the information it stores, processes, or transmits. Should the isolation measures fail, and an event occur that could damage the system, a prudent security policy provides the means to detect the failure and report the event to an assigned person for resolution (Puketza *et al*., 1996; SAMS, 1998).

A key challenge presented in protecting interconnected networks, intranets and extranets, is to maintain isolation from individuals with no legitimate need to access the system: "outsiders". Internetworking, particularly with the Internet, has been interpreted as a challenge by a significant number of those individuals, motivating them to develop techniques for penetrating system security. Penetration techniques are conveniently grouped under the concept of "hacking" with the practitioners of hacking being "hackers". Unfortunately, insiders as well as outsiders can engage in hacking activities. In fact, the insiders who misuse their privileges are much more likely to be successful in hacking activities, because the insider is not hampered by the protection measures implemented at the network perimeter (e.g. firewalls) to keep the network isolated from the outsider (Lunt, 1992; Amaroso, 1998; Stallings, 1999). Hackers exploit security holes in the network operating system to gain access. Many network managers feel that their networks are relatively secure and that it is unlikely that a hacker could gain unauthorized access. This often results in a false sense of security. One of the major failings of a large percentage of the security policies is that they do not provide a means of detecting and reporting security events as they happen. A hacker could be active, but undetected, in the network. Therefore, in reality, the manager has no way of knowing if a hacker is active in the network or not. In virtually all environments today, the security measures in place are not designed to alert management when an active attack is in progress (Phillips, 1992; SAMS, 1998; Karnik, 1998). Therefore, the only time the network manger knows a hacker is active is if the hacker is careless or destructive.

If the operational risk associated with a network is to be managed, the ability to recognize and respond to an attack while it is happening, is required. Otherwise, the potential of the attacker to inflict damage cannot be minimized. An attack monitoring and response capability integrated into the security mechanisms of an environment is required to address this issue. Typically implemented in software, an attack monitoring and response system continually monitors network traffic, looking for a known pattern. When it detects an unauthorized activity, the software responds automatically with predetermined action. It may report the attack, log the event, or terminate the unauthorized connection. Attack monitoring and response software operates in concert with other security mechanisms to address the risk associated with hacking (Lunt, 1992; Karnik, 1998; Stallings, 2000).

This paper introduces a design, a model and verification of the monitoring part of common intrusion detection framework (CIDF) using Petri Nets. The tested model which will be useful to help the designer to build a good monitoring part based on a correct model, is introduced. This paper consists of four parts. The first part is an introduction. The threat of the network is maintained and the Internet related risk are discussed. The second part is Petri Nets. Why I use Petri Nets as a modeling technique is answered, also basic information about this topic is presented. The third part is monitoring of the common intrusion detection. In section four, a framework of the proposed model using Petri Nets, and a description of the proposed model presented. Collected information and encapsulated DB structure are discussed in the penultimate section. The last part is the conclusion, which contains brief information about this paper and the major requirements of the monitoring part.

Intrusion's monitoring

Before examining some of the details of security management, it will be useful to characterize the security intrusion; the definition of the intrusion detection is (Amaroso, 1998) "Intrusion detection is the process of identifying and responding to the malicious activity targeted at computing and networking resources". Its security threats, which include the following (Stallings, 1998).

Interruption

An asset of the system is destroyed or becomes unavailable or unusable. This is a threat to availability (e.g. cutting a communication line, the disabling of the file management system, etc.) (see Figure 1).

Interception

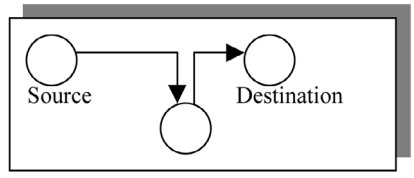
An unauthorized party gains access to an asset, it could be person, program, or computer (e.g. wiretapping, illicit copying, etc.) (see Figure 2).

Modification

An unauthorized party not only gains access, but also tampers with an asset which is a threat to integrity (e.g. changing value in a data file, altering a program so that it performs differently, modifying the content of a message being transmitted, etc.) (see Figure 3).

Figure 1
Cutting a communication line

Figure 2
An unauthorized party gains access to an asset



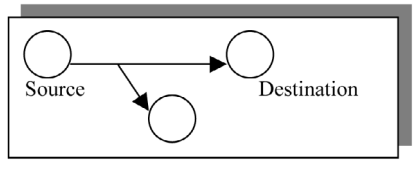
Fabrication

An unauthorized party inserts counterfeit objects into the system that is threat to integrity (e.g. addition of records to a file by insertion of spurious message in a network, etc.).

What is an *intrusion detection system* supposed to be doing? Let us think about how the system monitors the intrusion? Attack's monitoring and response software typically detects attacks using either rule-based or statistical anomaly approaches:

1. *Rule-based*. This approach draws from a library of known attack patterns or unauthorized activity, and watches for those specific types of attack. This is similar to the technique used in virus detection. The attack pattern library is updated continually as new types of attacks are discovered. Database updates are provided either by downloading new copies of the database automatically, or in new software releases. In this paper a proposed model of updating a database is presented.
2. *Statistical anomaly*. This approach operates on the assumption that users and networks always exhibit a predictable pattern of behavior, and do not depart from this pattern over short periods of time. A deviation is considered an attack. An example of that approach is repetition detection technique. Repetition detection is a powerful concept because it comes close to being able to detect intrusions

Figure 3
An unauthorized party tampers with an asset with a threat to its integrity



without knowing their details. For instance, if some intruder dreams up an intrusion that involves trying something over and over until it succeeds, then this may be detected as an intrusion indicator without even knowing the rationale for the repeated behavior. Some technical issues related to the detection of intrusions by repetitive behavior include the following:

- *Repetition threshold*. In the detection of repetitive behavior as indicators of intrusion, some numeric threshold must be identified to differentiate normal repetition from suspicious repetition. For example, if some user is trying to gain access to a system via an authentication response to a challenge, then a couple of unsuccessful tries might be considered normal. On the other hand, 20 or 30 unsuccessful attempts at acquiring a response to a challenge may be an indicator of an intrusion. Fine-tuning of the initial threshold is required.
- *Time repeat instances*. In the detection of repetitive behavior, the more time between repeat instance, the more difficult it becomes to recognize the pattern. From an intrusion detection system design perspective, if the first instance of some potential suspicious behavior occurs, then this would correspond to the expected time between potential instances of the behavior. For example, suppose that repeated attempts to guess passwords at a server are identified as a suspicious pattern, and that knowledge of one guessing instance is detected, then it is likely that the intrusion detection system will recognize the pattern. If, on the other hand, several days pass between instances of password guessing, then this might be more difficult to detect, as the buffers would likely be cleared.
- *Repetitive pattern*. If the behavior being repeated by an intruder is more complex than a simple service request (i.e. a password guess), then detection may be more difficult. For example, if an intruder is trying to gain access to some target asset by connecting through a series of intermediate systems, and performing a series of set-up operations that may or may not be suspicious, then the detection of the repetition cycle may not be easy. Similarly, if some attribute of the intruder changes (e.g. different Internet service providers are used for subsequent steps of an intrusion), then detection becomes more difficult.

Intrusion monitoring in security policy

Intrusion detection technology is an affordable and prudent measure that will significantly improve an enterprise's ability to manage operational to hacker activity and conduct safe computing in an interconnected network environment (Karnik, 1998; Stallings, 2000). It is recommended that intrusion technology be required by enterprises with network connectivity that spans enterprise borders; for example, public network connectivity or connections to other enterprise's networks. Intrusion detection technology should be used in conjunction with perimeter defenses and other appropriate technology, to provide robust and durable layered protection for the enterprise's information assets.

Advanced architecture

The security system consists of three integrated components running on central server:

1 *Recognition engine*. The recognition engine monitors the network in real-time, detecting and reporting attacks. It reports events to the administrator's module.
2 *Response engine*. The response engine reacts automatically to attack events as they are recognized, triggering pre-specified actions ranging from logging the attack and alerting the administrator to terminating offending connections.
3 *Administrator's module*. The administrator's module provides management of the recognition and response engines from a single GUI simplifying network management. The administrator can configure the recognition and response engines from the administrator's module to implement specific security policies (Phillips, 1992; Compaq Computer Corporation, 1998).

Response ways

Attack monitoring and response software can be configured to react automatically to an attack in a variety of ways, including:

- Log the event along with associated information.
- Alert the administration in real time through console message, e-mail, or pagers.
- Terminate the offending connection.
- Call a user-defined script or program.
- Perform a combination of these actions.

Monitoring model using state diagram and the problem formulation

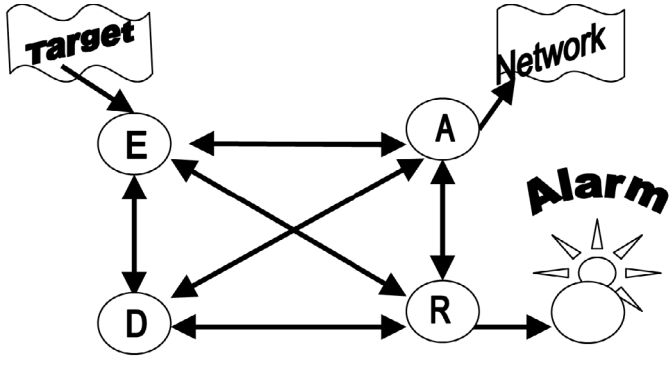
Figure 4 illustrates a simple representation of the monitoring and detecting model. This model consists of input state (target), four states ("E" events, "A" analysis, "D" database, and "R" response), and two output states (alarm and network connection). There are many arcs that connect these states with each other. The only way to analyze this model is to imagine and write down what you think about. There is no kind of verification and you might miss some important points since there is no rule to use. At this point we need another technique that allows us to verify our model based on specific rules.

Petri Nets modeling technique

Petri Nets form a powerful technique for the formal description of systems, particularly complex systems, which contain many interacting components, and concurrent and parallel activities (Billington, 1982; Konber, 1985; Baldwin, 1987). In fact, the greater the complexity the greater the benefit obtained from Petri Net modeling. Petri Nets have many fields of applications such as queuing theory, communication systems, computer networks, electronics, conflicts, political systems, traffic, etc.

The question might be asked, "Why use Petri Nets?" The answer lies in the inherent properties of Petri Nets, the major advantage lies in the fact that an analysis of Petri Nets reveals the criteria applicable to the model it represents and clearly demonstrates whether there are defects or not and whether the requirements are satisfied by the models or not. The implementation will therefore be based on the model which has been tested and proved correct before the implementation. In addition to this inherent capability, that gives Petri Nets an advantage over existing techniques, tools also exist to assist in analysis models based on it (Diaz, 1982; Konber, 1985; Phillips, 1992). Concurrency of activities creates many problems and requires synchronization. The procedure for obtaining a correct model is shown in Figure 5.

Figure 4
State diagram model



Petri Net structure

Petri Net is composed of a four parts: a set of places, a set of transitions, an input function, and an output function (Bochman, 1978; Diaz, 1982; Billington, 1982). The input functions relate transition and place. The input function is a mapping from a transition to a collection of places, known as the input places of transition. The output function maps a transition "T" to a collection of the places, known as the output places of the transition, Figure 6 illustrates this. For more information, the reader will find, in the reference part at the end of this paper, many resources on the subject of Petri Nets. We denote that the transition "T" is firing (i.e. the system can go to another state) if the pre-condition was satisfied, resulting in removing the tokens from its input places and adding tokens to its output places according to the arcs connecting them. (This operation is called marking transformation.)

Figure 5
Procedure for obtaining a correct model

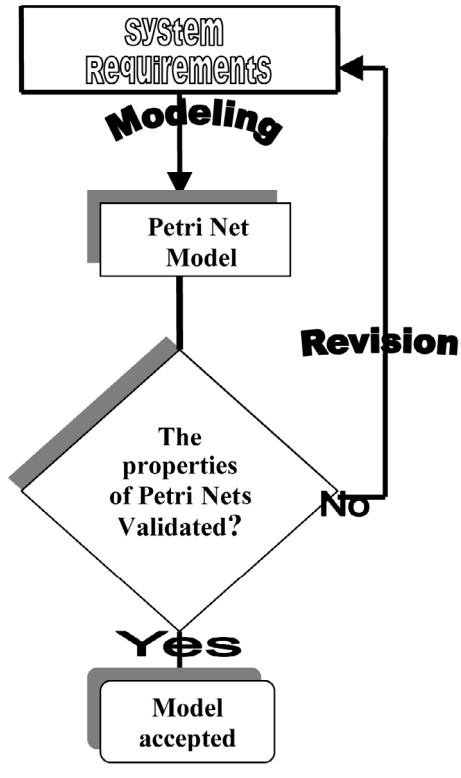
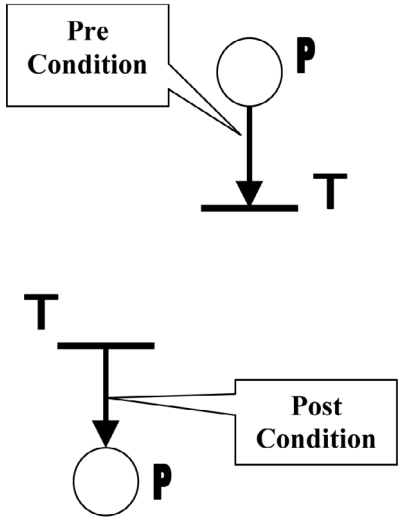


Figure 6
Petri Nets structure



Properties of Petri Nets

The main properties of Petri Nets related to correctness are:

- *Boundedness*. A Petri Net is said to be bounded if all places in the net are bounded for all the reachable marking. It is a very important property since it indicates that the system modeled by the Petri Nets can be implemented.
- *Liveness*. A Petri Net is live for a given marking if, and only if, any transition can be fired through a certain sequence of the system process.
- *Reinitialisable or cyclic*. A Petri Net is cyclic for an initial state if, and only if, there are sequences where the system can restart its operation after the process is done.

Plan of solution

This section discusses the plan of solution and the description of the proposed model of the monitoring part of CIDF. The model given in Figure 7 shows full details of the proposed model. Risk has many different components: assets, threats, vulnerabilities, safeguards, consequences, and likelihood. This examination normally includes gathering data about the threatened area and synthesizing and analyzing the information to make it useful. Audit trails can be used to review what occurred after an event, for periodic reviews and for real-time analysis. Reviewers should know what to look for to be effective in spotting unusual activity. They need to understand what normal activity looks like. Audit trail review can be easier if the audit trail function can be queried by user ID, terminal ID, application name, date and time, or some other set of parameters to run reports of selected information.

Proposed model description

Initially, let us suppose some target tries to connect to the network. Then the place P1 will have three tokens which will allow the transition T1, representing the monitoring system, to fire (i.e. the recognition system will start its investigation). Now we will go through this investigation, T1 fires, then P5, which represents pre-evaluation of the user, will receive a token, this token will allow T5 to fire. Since T5 fires, a token will go to P4, which represents requesting of some information from the database of the security system through the firing of T4.

T8 represents a counter which is used to count sequences of the related activities by marking P7, when P7 is marked with a certain threshold that the administrator adjusted, then we have two ways. The first one is the event does not meet the threshold point of the intrusion, then T3 will fire to reset the investigation as a forgiveness state and correlate these data in report through P2 and T2. The second way is the event does meet the threshold point, then T10 will fire removing a token from P7 and adding a token into P9 (P9 represents an event of attack).

P9 will check some information that was prepared on P8 (P8 represents a fast cache) through the firing of T11, and if the event satisfies the intrusion conditions then T14 fires. Otherwise T11 will keep firing until P9 receives all information that is needed. When T14 is fired, P11 will be marked (P11 represents the analysis state), P11 might need more information to complete its analysis, and this will be through the firing of T16.

T17, which represents the response action, is the last action of these sequences of firing. Two investigations will be made, the first one is the user is intruder (bad guy) then T19 will fire, making “alarm”, and prepare a complete report, the second one is that; the evidence is not intrusion, then T18 will fire making P15 and P13 marked.

P15, which represents the generation of the output report that will be kept as an archive database, will enable T6 to fire if it was marked. This report will be part of the expert system of the security knowledge base.

Database

There are two databases (DB, Cash DB) encapsulated within the model as shown in

Figure 7
Common intrusion detection framework

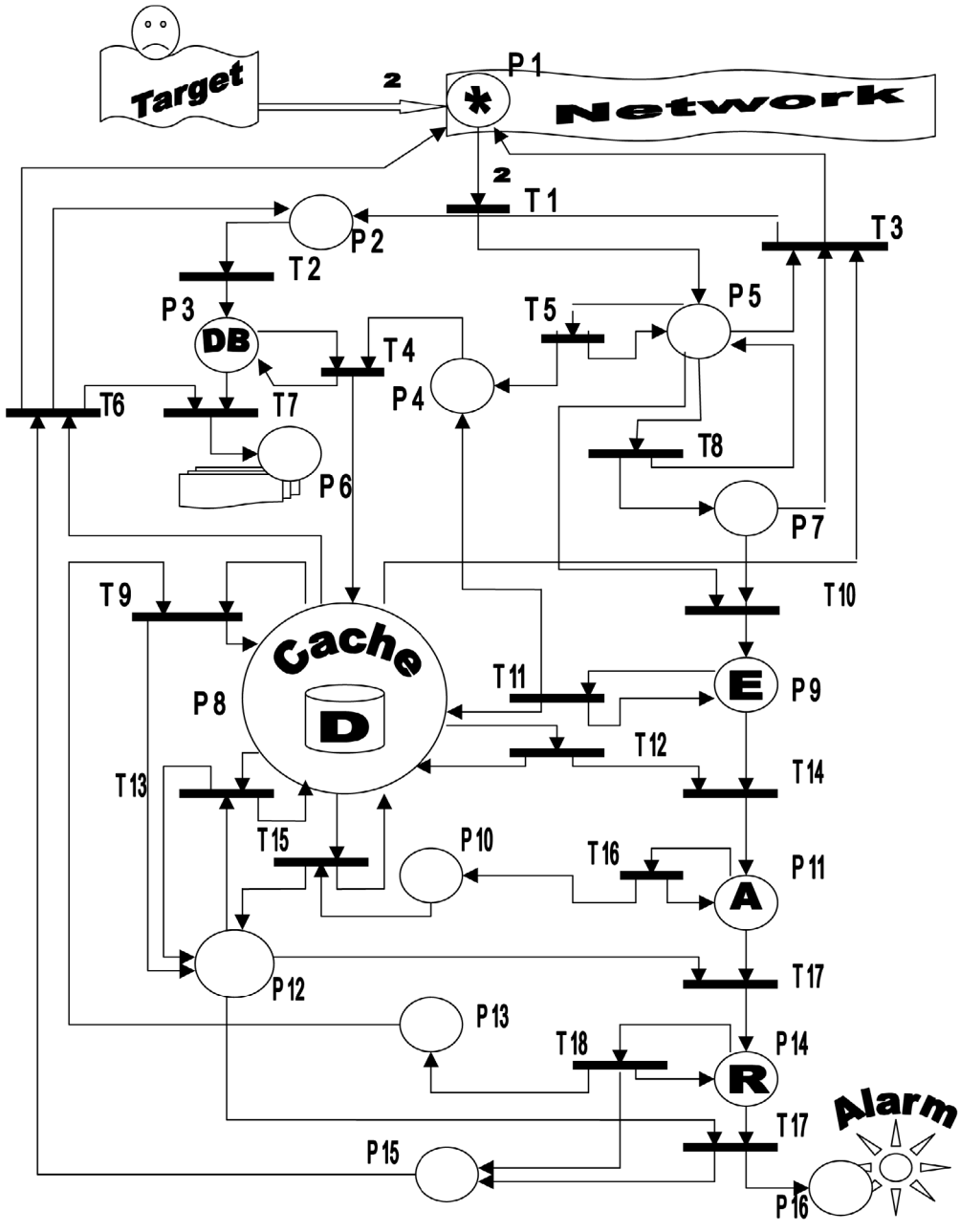


Figure 7. The first (DB) contains the initial data that may be predefined by the system administrator. Other data resulting from monitoring the network may be appended to these data. Figure 8 shows the structure of the schema of that database, it includes five tables. The first and the second include the data of the external and internal users such as: No_of_Flog: Number of failed login; Vact: Vacation day; Copy: Copying Process; Delete: Deleting; Process;Format: Formatting Process; Execute: Executing Process. The remaining three tables can be categorized into three classes (this categorization can be redefined according to the system administrator's demands), the first of these tables includes the most important data and access control list for each, the second and the third include the less important data.

The second database (Cash DB) contains a snapshot of the record that may be needed at this time. Table I shows a sample of the snapshot record. The illegal log in at any time that causes an alarm will represent a record which must be added to the first database. Table I shows the structure of the

database (note that this structure can be modified according to the administrator's requirements).

Conclusion

Attack monitoring and response software must meet a number of requirements to provide truly effective protection against attacks. The major requirements include:

- *Real-time reporting*. The attack monitoring and response software must be capable of detecting, reporting, and reacting to suspicious activity in real-time. Software that merely logs events is ineffective. After-the-fact detection is like a burglar alarm that goes off long after the burglar has fled. In addition, many attackers erase logs during the break-in, so the intrusion cannot be detected by merely scanning an event log.
- *Update capability*. Just as there is a progression of new computer viruses, hackers continually find new ways to break into computer systems. Therefore, attack monitoring and response software must be capable of continually adding to its knowledge base of known break-in patterns and unauthorized activity. The update function should be performed frequently and relatively easily. If the attack signatures are integral to the intrusion detection system, software releases should be issued when appropriate, and automatically downloaded. If the signatures are contained in table or database, the file should be automatically downloaded when appropriate.
- *Run on and support popular network operating system*. The software must support existing network infrastructure, including network operating systems.
- *Ease of configuration*. Configuration should be easy, without sacrificing effectiveness. The attack monitoring and response software should provide a default configuration so that administration can deploy it quickly and optimize it over time information accumulates. In addition, the software should provide sample configuration to guide administrators in setting up the system.
- *Flexibility*. To provide maximum flexibility, all control options should be configurable
- *Manageability*. Rapidly rising network management costs present a significant problem for organizations. Attack monitoring and response software must be easy to manage so that it does not contribute to this problem. Management of the software over the network, from a central location, is essential. This requires compliance with network management standards such as SNMPxx.
- *Adaptability*. Organizations are continually changing, driven by many factors including reorganizations, mergers, and acquisitions. Therefore, security policies are also in flux. To remain effective, attack monitoring and response software should be easy to adopt to changing security policies. This ensures that these policies can be implemented in fact, as well as on paper.

Figure 8
Structure of the database DB

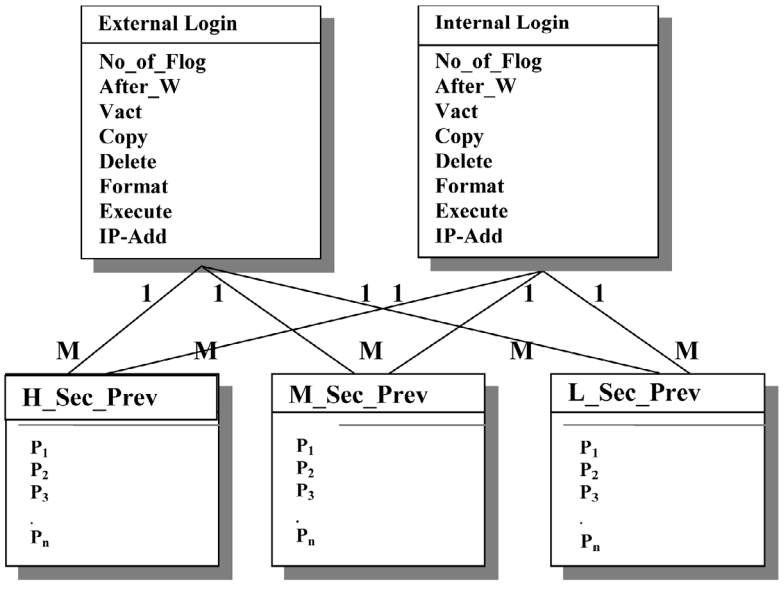


Table I
Sample of snapshot record

No of-Flog	Time	Term id	User id	Vact	IP-add	External	Copy	Delete	Run	P
3	Y	xxxx	xxxx	Y	99.99.99.99	Y	N	N	N	1
7	Y	xxxx	xxxx	Y	99.99.99.99	N	N	N	Y	2
5	Y	xxxx	xxxx	Y	99.99.99.99	N	N	N	Y	3
2	Y	xxxx	xxxx	Y	99.99.99.99	Y	N	N	Y	3
5	N	xxxx	xxxx	N	99.99.99.99	Y	N	N	Y	2
3	Y	xxxx	xxxx	Y	99.99.99.99	N	N	N	Y	3

- *Reporting features*. Reporting features should be easy to use and configure. A graphical user interface should be available to support reporting.
- *Secure*. The vendor should provide information related to the secure configuration of the operating environment and to protect the application. The product should also provide integrity controls to ensure that it cannot be easily modified or infected with malicious code (e.g. viruses).
- *Performance*. Because the application is analyzing packet traffic, it must be able to process large numbers of small packets at high speed. The vendor should address throughput constraints and performance degradation based on network traffic code.
- *Robust*. The intrusion's monitoring should monitor and detect all known attacks and many constraints should be required on that system.

References

Amoroso, Ed. (1998), "Intrusion detection, an introduction to Internet surveillance, processing, correlation, traps, trace back, and response," AT&T.

Baldwin, R.W. (1987), *Rule Based Analysis of Computer Security*, Massachusetts Institute of Technology, Cambridge, MA, June.

Billington, J. (1982), "Specification of the transport service using numerical Petri Nets", *Protocol Specification, Testing, and Verification Conference*, Sunshine Ed., North Holland Publishing, IFIP, Amsterdam.

Bochmann, G. (1978), "Specification and verification of computer communication protocols", Department d'Informatique et de Recherche Operationnelle, Universit de Montreal, December.

Compaq Computer Corporation (1998), "Intrusion detection in the enterprise network, managing hacker-related risk", March.

Diaz, M. (1982), "Modeling and analysis of communication and cooperation protocols using Petri Nets based models", *Computer Network*, Vol. 6, December, pp. 419-41.

Karnik, N. (1998), "Security in mobile agent systems", PhD dissertation, Department of Computer Science and Engineering, University of Minnesota, MN.

Konber, H.A. (1985), "High level Petri Nets to describe procedures for the signaling connection part of signaling system CCITT No. 7", Research report CIT-ALCATEL (Lannion-France), December.

Lunt, T. (1992), "Automated audit trail analysis for intrusion detection", *Computer Audit Update*, April, pp. 2-8.

Phillips, P.W. (1992), "New approach identifies malicious system activity," *Signal*, Vol. 46 No. 7, pp. 65-6.

Puketza, N.F., Zhang, K., Chung, M., Mukherjee, B. and Olsson, R.A. (1996), "A methodology for testing intrusion detection systems", *IEEE Transactions on Software Engineering*, Vol. 22, October, pp. 719-29.

SAMS (1998), *Maximum Security, a Hacker's Guide to Protecting Your Internet Site and Network*, 2nd ed., Sams Publishing.

Stallings, W. (1999), *SNMP, SNMPv2, SNMPv3, and RMON 1&2*, 3rd ed., Addison-Wesley, Reading, MA.

Stallings, W. (2000), *Network Security Essentials: Applications and Standards*, Prentice-Hall, Englewood Cliffs, NJ.