INTRUSION DETECTION: AN APPLICATION OF EXPERT SYSTEMS TO COMPUTER SECURITY

D. S. Bauer
F. R. Eichelman, II
R. M. Herrera
A. E. Irgon

Bellcore
444 Hoes Lane, RRC 1K-227C
Piscataway, NJ 08854
USA

ABSTRACT

Audit trails have long been an important component of a comprehensive computer security program. Unfortunately, the collected audit data is often not regularly analyzed and, in many cases, never even reviewed unless computer abuse is suspected. Many computer intrusions could very likely be discovered quickly if computer system audit trails were inspected on a regular basis by experts trained to recognize intrusive behavior. *Intrusion Detection* is a new research area in computer security focusing on developing the technology to detect intruders on computer systems in near real-time through the use of software systems that automatically analyze computer system audit trails. This paper presents an overview of current intrusion detection research and technology. The Network Intrusion Detection Expert System (NIDX), a software system recently prototyped by Bellcore, is described as an example of an Intrusion Detection System (IDS).

INTRODUCTION

The increasing reliance on computer systems to support sensitive applications, whose compromise could have far-reaching ramifications, has led to the recent increase in the attention given to computer security. There are three general categories of computer security mechanisms that may be implemented as part of a comprehensive computer security program: *preventive, detective,* and *recovery*. Of all computer security mechanisms, preventive and recovery mechanisms are the most well developed and deployed. Such mechanisms include: passwords, authorization tables and software, data encryption, physical security, and administrative policies and practices. Only the detective component remains mostly an ad-hoc process.

The conventional process for detecting or analyzing computer intrusions typically involves the manual inspection of audit logs. This approach suffers from several drawbacks: first, inspecting the sheer volume of audit information generated requires a large effort; second, although the collected audit data will often include evidence of intrusive activities, intrusions are not discovered until days, weeks or months after they occur - usually because other indications of an intrusion initiated an exhaustive search through the audit data; and third, the success of the inspection depends, in many cases, on the skill of the inspector - novice inspectors may miss all but the most obvious attacks.

It is important to detect computer intrusions as quickly as possible to reduce the potential impact to corporate and client data and operations. One method to permit fast detection is to maintain a continuous surveillance of user activities on a computer system. This can be accomplished via continuous inspection of audit trials of user activity generated by the computer operating system or applications used. Since manual surveillance of audit trails is impractical, one extension is to provide automated analysis via a software system. A new area of computer security research, called *intrusion detection*, is an effort to develop techniques for recognizing intruders on computer systems by using technologies such as expert systems, neural networks, and statistics to analyze computer-generated audit data.

At Bellcore, we have developed a prototype expert system, called the Network Intrusion Detection Expert System (NIDX), for detecting intrusions on UNIX® systems in near real-time. An expert system or knowledge-based type of Intrusion Detection System (IDS) encodes the knowledge of security experts in a software system which is then used to examine computer audit trails to detect potentially intrusive behavior by users of that computer system. The goal of IDSs is to be able to detect intrusions quickly, before significant damage is done. In this paper, architectural, detection, and security aspects of knowledge-based detection systems are presented. Throughout the paper, NIDX is used as an example to demonstrate capabilities, limitations, and other issues of intrusion detection.

IDS SYSTEM ARCHITECTURE

We define seven main components within an IDS architecture. These are:

- *Subjects* - entities that initiate actions, e.g., users.
- *Objects* - computer system resources acted upon by subjects, e.g., files, devices, programs, data.
- *Audit Records* - log of actions taken or attempted by subjects on objects.
- *Target System* - The computer system monitored by the IDS.
- *Host Computer* - The external computer in which the IDS resides.
- *Audit Line* - Connection going from the target system to the host computer.
- *Containment Line* - Connection going from the host computer to the target system.

UNIX is a registered trademark of AT&T.

The target system sends its audit information through the audit line. This information, consisting of all user and system interactions, is called the *audit trail*. The audit trail contains all the data needed to perform intrusion detection. However, there might be more data than needed. Hence, the audit trail may go through a *filter stage* which removes all information which is not relevant to the intrusion detection process. This refined audit trail arrives at the host computer; the IDS reads this data and detects possible anomalies in user behavior. In the case of imminent danger, the IDS can contact the target system via the containment line. This way, the security administrator can send notifications to the target system, and deny access to potential intruders in extreme cases.

There are three reasons for having the IDS resident in an external computer environment: first, to offload the target system from expending resources analyzing the data; second, productive analysis may require that one IDS use data from several internetworked target systems; and third, to limit the possibility of attack on the expert system itself. The existing ways of accessing the IDS from the target system, the audit and containment lines, should be extremely secure. If communication is done over a shared network, encryption should be used to safeguard the data.

This architecture is relatively generic. Its basic requirement is the presence of an secure audit trail generated by the target system. Hence, this architecture allows the implementation of intrusion detection mechanisms on any system or application which provides such an audit trail.

DETECTION PRINCIPLES

Intrusion detection technology is based on the premise that, either in the attempt to break into a system, or once having broken into a system, an intruder will exhibit some form of abnormal behavior when attempting to access and control the system's resources. This abnormal behavior can then be detected by analysis of an activity audit trail, allowing appropriate alerting and recommendations for action to the system security administrator.

In order to develop an automated detection capability, the characteristics of computer intrusions and intruders must be defined. Detailed discussion of intruder *modus operandi* is beyond the scope of this paper. Another paper in these proceedings [1] describes many of the attributes of intrusions and intruders. Based on case studies [2] and observations of typical computer users' behavior [3], intrusion detection researchers created two models to represent intruder behavior that may be employed by intrusion detection systems.

The first is a *behavioral* model that is based on the assumption that an intruder who *masquerades*[1] as a valid user will exhibit significantly different behavior from the valid user. This model attempts to address the situation where an intruder breaks into a computer system by discovering the login and password of an authorized user of that computer system. Detection is accomplished by first initializing and updating profiles based on the observed behavior of normal users and then monitoring events on the system looking for anomalous behavior. Anomalous behavior is discovered by comparing a users' current actions to their profiled history and determining by statistical or other methods that a person logged in as user x is exhibiting behavior not consistent with x's usual behavior. For example, if a user performs actions (e.g. file accesses) the number or type of which deviates substantially from the mean of that activity as shown by his/her profile, then such behavior is deemed anomalous or suspicious. In practice, many actions are monitored and reflected in each users' profile, e.g., files read, programs run, disk space used, commands executed, and CPU used. The Intrusion Detection Expert System (IDES) [4], is a good example of an IDS that implements the behavioral model.

Another model is the *scenario* model which is based on using the knowledge experts have concerning the methods employed by intruders when computer systems are attacked. Detection of intrusions is accomplished by comparing current user activities to predetermined patterns or scenarios of unacceptable or intrusive behavior. A suspicious event or security violation is noted when user actions match one or more of the scenarios encoded in the IDS. In this model, the a-priori definition of unacceptable or intrusive behavior is crucial and is generally accomplished via collaboration with security subject matter experts; although security literature also provides excellent case studies. The knowledge base of the IDS expert system contains all of the information necessary to match user activities to scenarios.

Both the behavioral and scenario models have advantages and disadvantages. For example, the behavioral model can be system independent because it is based on detecting deviations in various measures, e.g., the amount of CPU time used, without semantic interpretation. However, it also can be insensitive to a small changes in user behavior over time which can ultimately lead to a serious security violation. Scenario-based systems require that all potential intrusion scenarios be determined and encoded beforehand - a basically intractable task. However, in our experience, intruder activity that matches a scenario tend to be detected very quickly. Finally, the type of audit data available and the kinds of intrusions that need to be detected affect the choice of model that may be employed. Behavioral models work well if specific intrusions cannot be identified, i.e., all user activities are legal and the only concern is that an unauthorized person may be performing them. The scenario model works very well if there is adequate expertise available to describe common intruder goals and methods. In practice, a hybrid approach is usually best. As such, we have chosen to implement both methods in NIDX, as described below.

NIDX DETECTION STRATEGY

The NIDX IDS prototype [5] was developed to detect command or shell level intrusions on UNIX systems: the unauthorized browsing or modification of system objects and the unauthorized achievement of *root*[1] status. No attempt was made to detect intrusions into higher level applications

1. When an intruder assumes the identity of a valid user, it is known as a masquerade intrusion.

1. *Root* privilege in UNIX allows system administrators to perform delicate system operations. A *root* user has an effective user-id of 0.

that may reside on top of UNIX.

NIDX uses two major strategies to detect the above intrusions and other anomalous behavior. The first of these strategies is a *behavior-oriented* approach. This approach is based on the premise that users tend to access a limited number of objects within the target system while performing their work. This collection of normal user accesses is what we refer to as the user's *working set*. The user's working set is created by recording the access permission list for the system objects which the user can legally access. The access permission list for a system object consists of a series of user id numbers associated with that object. By adding the user number to the access list, NIDX will not generate an alarm if that user subsequently accesses that object. A training mechanism has been developed that permits NIDX to automatically learn the set of normal object accesses initiated by each user. This training, of course, must be carefully supervised so that unauthorized activities are not recorded as "normal".

Once all objects are updated, NIDX can determine whether current user interactions are legal or not by simply checking if the user number appears in the access permit list of the object being handled. Legal interactions are ignored and logged in an external file. However, any illegal access committed by the user is stored in the user profile, and a suspicious event is generated. Finally, the penalty weight corresponding to the object is added to an intrusion counter in the user profile. If this counter exceeds a threshold specified in the user's profile, then the access will be considered an alarm, instead of a suspicious event. All resulting suspicious events and alarms are displayed by NIDX, and logged for future reference.

Our second intrusion detection strategy is a scenario-based approach. Scenarios are either a single event or a sequence of events that are known to be suspicious by security experts. Scenarios are represented as heuristic (rule thumb) *rules*. Rules are composed of *conditions* and *actions*. The conditions of a rule represent the sequence of events, indicated by audit records, that correspond to a particular scenario while the actions of a rule represent the containment or alerting measures to pursue whenever that scenario is detected.

A simple example of a rule in the UNIX domain is:

RULE check-change-user-id
IF *a user's real user-id is not* 0
AND *a user's effective user-id becomes* 0
AND *the object used for this change is unknown*
THEN
{*cause alarm* - POTENTIAL UNAUTHORIZED ROOT ACCESS}

This rule describes a scenario in which a user obtains *root* user privilege by executing a program not known to the system for such purpose. Whenever this scenario occurs, the conditions for the rule will be fulfilled and an alarm will be generated. Other actions, such as sending a notification to the user or even logging the user off the target system remotely, could be taken at the administrator's judgement.

This kind of approach allows the security administrator to catch a user whose behavior matches a specific scenario. Currently, scenarios are encoded by AI specialists working in conjunction with security specialists. Scenarios could be locally specified by the security administrator. The administrator would have to code these rules into NIDX using the proper format.

The union of these approaches provides a strong combination against both intrusions and misuse. Even if potentially dangerous interactions were included in the working set of a highly privileged user, the scenario mechanism would be able to catch such interactions if the user executes these. On the other hand, an intruder can steal a password for a authorized user and "legally" access the system. No scenario can detect such an intrusion. However, once on the system, the intruder must only access objects within the original user's working set to remain undetected. In our experience, we have found that intruders immediately begin exploring a system upon penetration - this activity can usually be quickly detected. In addition, the intruder would have to avoid any other scenarios the system knows about.

CONCLUSION

Conventional security tools provide various levels of protection against potential intruders. However, this protection is given only as long as the preventive mechanisms are not compromised. Once these mechanisms are defeated by a sufficiently determined intruder, the underlying system is again left open to tampering. Moreover, this tampering is often untraceable by conventional means.

A complementary mechanism to intrusion prevention is detection. Conventionally, this is achieved by providing detailed audit data that allows security officers to detect intrusions, assess damage and repair security holes. The approaches described in this paper in which the audit trail is analyzed in real-time using expert system technology is a substantial improvement over batch mode audit trail inspection. A key objective of real-time analysis is that it may allow real-time containment of an intruder's activity, thereby limiting the damage that can be done.

Our NIDX prototype showed that near-real-time detection could be accomplished for intrusions at the UNIX system command level. Command level intrusions are common among intruders - especially from exploratory intruders who are trying to find a use for a newly discovered system. Intrusions into higher level applications, such as those that control communications networks for example, require a different level of detection. In this case, detection must be done in terms of higher-level application-oriented objects, e.g., communications network elements as opposed to operating system objects. We are now expanding our work to the analysis of application audit data in order to develop IDSs that can detect attacks aimed at large transaction oriented software systems.

REFERENCES

1. Kluepfel, H.M., Foiling the Wiley Hacker: More than Analysis and Containment, *Proceedings of the 1989 International Carnahan Conference on Security Technology,* October 1989.
2. C. Stoll, What Do You Feed A Trojan Horse?, *Proceedings of the 10th National Computer Security Conference,* September 1987, pp 231.
3. T.F. Lunt, Automated Audit Trail Analysis and Intrusion Detection: A Survey, *Proceedings of the 11th National Computer Security Conference,* October 1988.
4. T.F. Lunt and R. Jagannathan, A Prototype Real-Time Intrusion-Detection Expert System, *Proceedings of the 1988 IEEE Symposium on Security and Privacy,* April 1988.
5. D.S. Bauer and M.E. Koblentz, NIDX - A Real-time Intrusion Detection Expert System for UNIX Systems, *Proceedings of the 1988 Summer USENIX Conference,* June, 1988.