An Intrusion-Detection Model

DOROTHY E. DENNING

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-13, NO. 2, FEBRUARY 1987, 222-232.

Abstract-A model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse is described. The model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model includes profiles for representing the behavior of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behavior from audit records and for detecting anomalous behavior. The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system.

Index Terms-Abnormal behavior, auditing, intrusions, monitoring, profiles, security, statistical measures.

I. INTRODUCTION

This paper describes a model for a real-time intrusion-detection expert system that aims to detect a wide range of security violations ranging from attempted break-ins by outsiders to system penetrations and abuses by insiders. The development of a real-time intrusion-detection system is motivated by four factors: 1) most existing systems have security flaws that render them susceptible to intrusions, penetrations, and other forms of abuse; finding and fixing all these deficiencies is not feasible for technical and economic reasons; 2) existing systems with known flaws are not easily replaced by systems that are more secure-mainly because the systems have attractive features that are missing in the more-secure systems, or else they cannot be replaced for economic reasons; 3) developing systems that are absolutely secure is extremely difficult, if not generally impossible; and 4) even the most secure systems are vulnerable to abuses by insiders who misuse their privileges.

The model is based on the hypothesis that exploitation of a system's vulnerabilities involves abnormal use, of the system; therefore, security violations could be detected from abnormal patterns of system usage. The following examples illustrate:

- *Attempted break-in:* Someone attempting to break into a system might generate an abnormally high rate of password failures with respect to a single account or the system as a whole.
- *Masquerading or successful break-in:* Someone logging into a system through an unauthorized account and password might have a different login time, location, or connection type from that of the account's legitimate user. In addition, the penetrator’s behavior may differ considerably from that of the legitimate, user-, in particular, he might spend most of his time browsing through directories and executing system status commands, whereas the legitimate user might concentrate on editing or compiling and linking programs. Many break-ins have been discovered by security officers or other users on the system who have noticed the alleged user behaving strangely.
- *Penetration by legitimate user:* A user attempting to penetrate the security mechanisms in the operating system might execute different programs or trigger more protection violations from attempts to access unauthorized files or programs. If his attempt succeeds, he will have access to commands and files not normally permitted to him.
- *Leakage by legitimate user:* A user trying to leak sensitive documents might log into the system at unusual times or route data to remote printers not normally used.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
FEB 1987		**00-00-1987 to 00-00-1987**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
An Intrusion-Detection Model	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Center of Terrorism and Irregular Warfare,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A model of a real-time intrusion-detection expert system capable of detecting break-ins, penetrations, and other forms of computer abuse is described. The model is based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage. The model includes profiles for representing the behavior of subjects with respect to objects in terms of metrics and statistical models, and rules for acquiring knowledge about this behavior from audit records and for detecting anomalous behavior. The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**17**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

- *Inference by legitimate user:* A user attempting to obtain unauthorized data from a database through aggregation and inference might retrieve more records than usual.
- *Trojan horse:* The behavior of a Trojan horse planted in or substituted for a program may differ from the legitimate program in terms of its CPU time or 1/0 activity.
- *Virus:* A virus planted in a system might cause an increase in the frequency of executable files rewritten, storage used by executable files, or a particular program being executed as the virus spreads.
- *Denial-of-Service:* An intruder able to monopolize a resource (e.g., network) might have abnormally high activity with respect to the resource, while activity for all other users is abnormally low.

Of course, the above forms of aberrant usage can also be linked with actions unrelated to security. They could be a sign of a user changing work tasks, acquiring new skills, or making typing mistakes; software updates; or changing workload on the system. An important objective of our current research is to determine what activities and statistical measures provide the best discriminating power; that is, have a high rate of detection and a low rate of false alarms.

II. OVERVIEW OF MODEL

The model is independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection expert system, which we have called IDES. A more detailed description of the design and application of IDES is given in our final report [1].

The model has six main components:

- *Subjects:* Initiators of activity on a target system- normally users.
- *Objects:* Resources managed by the system-files, commands, devices, etc.
- *Audit records:* Generated by the target system in response to actions performed or attempted by subjects on objects-user login, command execution, file access, etc.
- *Profiles:* Structures that characterize the behavior of subjects with respect to objects in terms of statistical metrics and models of observed activity. Profiles are automatically generated and initialized from templates.
- *Anomaly records:* Generated when abnormal behavior is detected.
- *Activity rules:* Actions taken when some condition is satisfied, which update profiles, detect abnormal behavior, relate anomalies to suspected intrusions, and produce reports.

The model can be regarded as a rule-based pattern matching system. When an audit record is generated, it is matched against the profiles. Type information in the matching profiles then determines what rules to apply to update the profiles, check for abnormal behavior, and report anomalies detected. The security officer assists in establishing profile templates for the activities to monitor, but the rules and profile structures are largely system-independent.

The basic idea is to monitor the standard operations on a target system: logins, command and program execution's, file and device accesses, etc., looking only for deviations in usage. The model does not contain any special features for dealing with complex actions that exploit a known or suspected security flaw in the target system; indeed, it has no knowledge of the target system's security mechanisms or its deficiencies. Although a flaw-based detection mechanism may have some value, it would be considerably more complex and would be unable to cope with intrusions that exploit deficiencies that are not suspected or with personnel-related vulnerabilities. By detecting the intrusion, however, the security officer may be better able to locate vulnerabilities.

The remainder of this paper describes the components of the model in more detail.

III. SUBJECTS AND OBJECTS

Subjects are the initiators of actions in the target system. A subject is typically i terminal user, but might also be a process acting on behalf of users or groups of users, or might be the system itself. All activity arises through commands initiated by subjects. Subjects may be grouped into different classes (e.g., user groups) for the purpose of controlling access to objects in the system. User groups may overlap.

Objects are the receptors of actions and typically include such entities as files, programs, messages, records, terminals, printers, and user- or program-created structures. When subjects can be recipients of actions (e.g., electronic mail), then those subjects are also considered to be objects in the model. Objects are grouped into classes by type (program, textfile, etc.). Additional structure may also be imposed, e.g., records may be grouped into files or database relations; files may be grouped into directories. Different environments may require different object granularity; e.g., for some database applications, granularity at the record level may be desired, whereas for most applications, granularity at the file or directory level may suffice.

IV. AUDIT RECORDS

Audit Records are 6-tuples representing actions performed by subjects on objects:

<Subject, Action, Object, Exception-Condition, Resource-Usage, Time-stamp>

where

- *Action:* Operation performed by the subject on or with the object, e.g., login, logout, read, execute.
- *Exception-Condition:* Denotes which, if any, exception condition is raised on the return. This should be the actual exception condition raised by the system, not just the apparent exception condition returned to the subject.
- *Resource-Usage:* List of quantitative elements, where each element gives the amount used of some resource, e.g., number of lines or pages printed, number of records read or written, CPU time or 1/0 units used, session elapsed time.
- *Time-stamp:* Unique time/date stamp identifying when the action took place.

We assume that each field is self-identifying, either implicitly or explicitly, e.g., the action field either implies the type of the expected object field or else the object field itself specifies its type. If audit records are collected for multiple systems, then an additional field is needed for a system identifier.

Since each audit record specifies a subject and object, it is conceptually associated with some cell in an "audit matrix" whose rows correspond to subjects and columns to objects. The audit matrix is analogous to the "access-matrix" protection model, which specifies the rights of subjects to access objects; that is, the actions that each subject is authorized to perform on each object. Our intrusion-detection model differs from the access-'matrix model by substituting the concept of "action performed" (as evidenced by an audit record associated with a cell in the matrix) for "action authorized" (as specified by an access right in the matrix cell). Indeed, since activity is observed without regard for authorization, there is an implicit assumption that the access controls in the system permitted an action to occur. The task of intrusion detection is to determine whether activity is unusual enough to suspect an intrusion. Every statistical measure used for this purpose is computed from audit records associated with one or more cells in the matrix.

Most operations on a system involve multiple objects. For example, file copying involves the copy program, the original file, and the copy. Compiling involves the compiler, a source program file, an object

program file, and possibly intermediate files and additional source files referenced through "include" statements. Sending an electronic mail message involves the mail program, possibly multiple destinations in the "To" and "cc" fields, and possibly "include" files.
Our model decomposes all activity into single-object actions so that each audit record references only one object. File copying, for example, is decomposed into an execute operation on the copy command, a read operation on the source file, and a write operation on the destination file. The following illustrates the audit records generated in response to a command

COPY GAME.EXE TO <Library>GAME.EXE

issued by user Smith- to copy an executable GAME file into the <Library> directory; the copy is aborted because Smith does not have write permission to < Library >:

(Smith, execute, <Library>COPY.EXE, 0, CPU=00002, 11058521678)
(Smith, read, <Smith>GAME.EXE, 0, RECORDS=O, 11058521679)
(Smith, write, <Library> GAME.EXE, write-viol, RECORDS=O, 11058521680)

Decomposing complex actions has three advantages. First, since objects are the protectable entities of a system, the decomposition is consistent with the protection mechanisms of systems. Thus, IDES can potentially discover both attempted subversions of the access controls (by noting an abnormality in the number of exception conditions returned) and successful subversions by noting an abnormality in the set of objects accessible to the subject). Second, single-object audit records greatly simplify the model and its application. Third, the audit records produced by existing systems generally contain a single object, although some systems provide a way of linking together the audit records associated with a "job step" (e.g., copy or compile) so that all files accessed during execution of a program can be identified.

The target system is responsible for auditing and for transmitting audit records to the intrusion-detection system for analysis (it may also keep an independent audit trail). The time at which audit records are generated determines what type of data is available. If the audit record for some action is generated at the time an action is requested, it is possible to measure both successful and unsuccessful attempts to perform the activity, even if the action should abort (e.g., because of a protection violation) or cause a system crash. If it is generated when the action completes, it is possible to measure the resources consumed by the action and exception conditions that may cause the action to terminate abnormally (e.g., because of resource overflow). Thus, auditing an activity after it completes has the advantage of providing more information, but the disadvantage of not allowing immediate detection of abnormalities, especially those related to break-ins and system crashes. Thus, activities such as login, execution of high risk commands (e.g., to acquire special “superuser" privileges), or access to sensitive data should be audited when they are attempted so that penetrations can be detected immediately; if resource-usage data are also desired, additional auditing can be performed on completion as well. For example, access to a database containing highly sensitive data may be monitored when the access is attempted and then again when it completes to report the number of records retrieved or updated. Most existing audit systems monitor session activity at both initiation (login), when the time and location of login are recorded, and termination (logout), when the resources consumed during the session are recorded. They do not, however, monitor both the start and finish of command and program execution or file accesses. IBM's System Management Facilities (SMF) [2], for example, audit only the completion of these activities.

Although the auditing mechanisms of existing systems approximate the model, they are typically deficient in terms of the activities monitored and record structures generated. For example, Berkeley 4.2 UNIX [3] monitors command usage but not file accesses or file protection violations. Some systems do not record all login failures. Programs, including system programs, invoked below the command level are not explicitly monitored (their activity is included in that for the main program). The level at which auditing

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.