IDES: The Enhanced Prototype
A Real-Time Intrusion-Detection Expert System

**Teresa F. Lunt, R. Jagannathan, Rosanna Lee,
Sherry Listgarten, David L. Edwards,
Peter G. Neumann, Harold S. Javitz,
and Al Valdes**

SRI-CSL-88-12, October 1988
SRI Project 4185-010

Computer Science Laboratory
SRI INTERNATIONAL
333 Ravenswood Ave.
Menlo Park, CA 94025-3493
(415) 859-5924

SRI International

IDES: The Enhanced Prototype
A Real-Time Intrusion-Detection Expert System

**Teresa F. Lunt, R. Jagannathan, Rosanna Lee,
Sherry Listgarten, David L. Edwards,
Peter G. Neumann, Harold S. Javitz,
and Al Valdes**

SRI-CSL-88-12, October 1988
SRI Project **4185-010**

**Computer Science Laboratory
SRI INTERNATIONAL**

Abstract

This report describes the design and implementation of a real-time intrusion-detection expert system (IDES) designed and developed by SRI International. IDES is an independent system that monitors the activities of different types of subjects, such as users and remote hosts, of a target system to detect security violations by both insiders and outsiders as they occur. IDES adaptively learns subjects'behavior patterns over time and detects behavior that deviates from these patterns. IDES also has an expert system component that can be used to encode information about known system vulnerabilities and intrusion scenarios.

This work was supported by the U.S. Navy, SPAWAR, which funded SRI through subcontract 9-X5H-4074J-1 with the Los Alamos National Laboratory.

Contents

List of Figures

Chapter 1

Introduction

Although a computer system's primary defense is its access controls, numerous newspaper accounts of break-ins and computerized thefts clearly demonstrate that access control mechanisms cannot be relied upon in most cases to safeguard against a penetration or insider attack. Most computer systems have security susceptibilities that leave them vulnerable to attack and abuse. Finding and fixing all of the flaws is not technically feasible, and building systems with no security vulnerabilities is extremely difficult, if not generally impossible. Moreover, even the most secure systems are vulnerable to abuse by insiders who misuse their privileges. Audit trails can establish accountability of users for their actions, and have been viewed as the final defense, not only because of their deterrent value, but because in theory they can be perused for suspicious events and they provide evidence to establish the guilt or innocence of suspected individuals. Moreover, analysis of audit trails may be the only means of detecting authorized but abusive user activity.

While many computer systems collect audit data, most do not have any capability for automated analysis of that data. Moreover, those systems that *do* collect audit data generally collect large volumes of data that are not necessarily security-relevant. Thus, for security analysis, a security officer must wade through stacks of printed output of audit data. Beyond the pure tedium of this task, the sheer volume of the data makes it impossible for a security officer to detect suspicious activity that does not conform to a handful of obvious intrusion scenarios. What is needed is the capability for automated security analysis of audit trails.

IDES, an Intrusion-Detection Expert System developed by SRI Interna-

tional for SPAWAR, addresses this problem. IDES is an independent system that runs on its own hardware and processes audit data characterizing user activity received from a target system [1,2].

IDES provides a system-independent mechanism for real-time detection of many types of security violations, whether they are initiated by outsiders attempting to break into a system or by insiders attempting to misuse the privileges of their accounts. The IDES approach is based on the hypothesis that any exploitation of a computer system's vulnerabilities entails abnormal use of the system; consequently, intrusions can be detected by observing unusual patterns of use. IDES is based on the intrusion-detection model developed by Denning [3,4]. This model is independent of any particular target system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection system using real-time analysis of audit data.

IDES processes audit data received from a target system via a network. It is capable of monitoring the behavior of users, hosts, and the target system as a whole. IDES determines whether a subject's behavior as reported in the audit data is normal with respect to the subject's past behavior and with respect to the behavior of subjects that are designated as being similar in some regard to the subject in question. The statistics that describe the historically observed behavior patterns for each subject are updated daily and aged so that the older audit data contributes less weight to the overall statistics; thus, IDES adaptively "learns" the subjects' behavior patterns. In addition to this statistical intrusion-detection capability, IDES has a built-in expert system shell that allows for the application of heuristics that have proven useful in detecting intruders. These two intrusion-detection techniques combine to give IDES the capability of detecting internal and external penetrators of various skill levels who are using a variety of techniques.

1.1 The Threats Addressed by IDES

Anderson [5] categorized the threats that could be addressed by audit trail analysis as

- External penetrators (who are not authorized the use of the computer).
- Internal penetrators (who are authorized use of the computer but are not authorized for the data, program, or resource accessed), including:

- Masqueraders (who operate under another user's id and password)
- Clandestine users (who evade auditing and access controls).

- Misfeasors (authorized users of the computer *and* resources accessed who misuse their privileges).

Anderson suggested that masqueraders can be detected by observing departures from established patterns of use for individual users. This is one approach taken by IDES; thus, we can expect that IDES will be potentially capable of detecting masqueraders.

Anderson suggested that external penetrators can be detected by auditing failed login attempts, and that some would-be internal penetrators can be detected by observing failed access attempts to files, programs, and other resources. This suggests an approach of characterizing intrusions, as opposed to characterizing normal user behavior. We are including expert system rules in IDES that would characterize certain types of intrusions. This will give IDES the potential to detect external and internal penetrators.

Anderson had little to offer toward detecting the legitimate user who abuses his or her privileges. IDES takes two approaches to detecting this kind of abuse. First, IDES compares a user's behavior with the norm established for the class of users (i.e. *group)* to which the user belongs, to detect abuse of privilege. We expect this approach to be especially useful when there is a very large number of users (in the thousands or tens of thousands) who may operate in distinct roles or job capacities. Secondly, we are planning to include *a priori* rules for "socially unacceptable" behavior, analogous to those that characterize intrusions, in IDES's expert system rule-base.

The clandestine user can evade auditing by misuse of system privilege or by operating at a level below which auditing occurs. The former could be detected by auditing all use of functions that turn off or suspend auditing, change the specific users being audited, or change other auditing parameters. The latter could be addressed by performing auditing at a low level, such as auditing system service or kernel calls. We will be adding the capability to use information obtained from auditing system calls when we begin to monitor networks of Sun workstations in the next phase of the IDES project.

Anderson's suggestion for detecting the clandestine user is to monitor certain system-wide parameters, such as CPU, memory, and disk activity, and compare these with what has been historically established as "usual"

or normal for that facility. IDES performs system-wide monitoring for several variables and keeps a profile for the target system as a whole. We do not claim that IDES will ever be effective for detecting clandestine users. However, although a skilled penetrator will be able to disable the audit mechanisms in order to work undetected, auditing and intrusion-detection mechanisms are still of value in detecting the less skilled penetrator, because they increase the difficulty of penetration. Moreover, Linde has suggested [6] that auditing and intrusion-detection mechanisms can make it so difficult for a penetrator to avoid detection that other methods, such as bribing a user or system personnel, will be more attractive.

1.2 The IDES Prototype

IDES is currently implemented on two Sun-3 workstations and uses the Oracle relational database system as well as a home-grown expert system shell. The IDES processes perform the following functions: communicate with the target systems; receive, decrypt, and validate audit records; perform the intrusion-detection analysis; and keep track of subjects' normal behavior patterns. They are implemented on a Sun-3/260 with a 560 megabyte disk. Because these activities must occur in real-time, they require their own workstation so that they are not impeded by the IDES security administrator interface activity.

The IDES security administrator interface is implemented on a Sun-3/60 workstation. It maintains a continuous display of various indicators of user behavior on the monitored systems and allows the security administrator to choose from a menu of built-in queries or to build ad-hoc queries against the audit data and profiles. The interface is on a separate workstation because it requires substantial computing both for display generation and for administrator-initiated query processing.

Implementing IDES on a machine separate from the target system has advantages with respect to performance, security, and integration. IDES does not noticeably degrade the response time of the system monitored or otherwise affect its behavior. In principle, IDES can be made tamper-resistant from would-be intruders whose activity is being monitored on the target system, so that any flaws that exist in the target system do not endanger the security of IDES. And IDES can be adapted to different environments and integrated with different types of host systems. In particular, IDES can be used to monitor any system that can transmit audit data to it

over a network according to the IDES communication protocol and generic audit record format.

Although IDES could preprocess the target system audit data into the IDES audit record format, this would significantly degrade IDES's performance. For large systems that collect vast amounts of audit data, we envisage using only a small fraction of the total amount of audit data generated. Thus, the volume of data processed by IDES is drastically reduced by performing the preprocessing on the target system rather than on the IDES workstation.

We developed a flexible format for the audit records that IDES expects to receive from the target system and a secure protocol for the transmission of audit records from the target system. The IDES protocol is system-independent; in principle, IDES can be used to monitor different systems without fundamental alteration (although the particular measures and parameters chosen will depend on the system and users being monitored).

The IDES prototype currently monitors a DEC-2065 machine at SRI running a locally customized version of the TOPS-20 operating system. We have modified the TOPS-20 operating system to collect audit data, transform the data into the IDES format, encrypt the formatted data, and transmit the records to IDES.

1.3 Summary of Enhancements

The IDES prototype described in this report contains significant improvements over the earlier system described previously [1]. The earlier system was intended as a proof-of-concept for IDES, whereas the system described here is intended as an exploration into the use of different measures, statistical methods, and security administrator interfaces in intrusion-detection systems.

The initial IDES prototype monitored three user measures, with the profiles containing only means and counts for each measure. Anomaly detection was done individually for each measure by comparison of observed activity with the historically observed mean for that measure, as stored in the user's profile. The interface consisted of a component to view each anomaly in detail, a component for viewing the status of IDES (for example, the number of audit records processed), and a mechanism for querying the IDES database.

The current IDES prototype is much more sophisticated in all respects.

IDES now supports three types of subjects: users, remote hosts, and target systems. In addition, subjects of the same type may be gathered into groups so that their individual behavior can be compared with that of the groups to which they belong. IDES monitors 36 different measures, divided among the three types of subjects. Instead of using only counters for the measures, as was done with the initial prototype, the current system distinguishes between counter-type measures (called *continuous* measures) and measures recording events that are nonnumerical in nature (called *categorical* measures). Within each measure, events that generate data for the measure are further divided into *working statuses* depending on *when* they occur, in order to classify events by the level of activity expected. For example, certain days may be classified as *working* days for users and other days as *nonworking* days. Anomaly detection and profile maintenance have changed accordingly to take into account the increased number of subject types, measure types, groups, and different working statuses. In addition, anomaly detection also takes into account the correlation between different measures. When an anomaly is detected, IDES also examines the individual measures to determine which ones contributed most to the anomaly. The current IDES system is also augmented with an expert system shell which allows non-profile-based vulnerabilities to be captured. The security administrator interface, in addition to providing the capabilities offered by the initial prototype, allows the security administrator, if desired, to customize IDES for each subject with respect to different intrusion-detection measures, different working days/hours, and membership in different groups. The security administrator is provided with the capability to roll back a subject's profile in order to exclude data containing real security violations (so as not to "contaminate" existing profiles). It also allows the security administrator to monitor any time segment of interest comprehensively, for example, by displaying in detail the audit records received for that time segment.

1.4 Report Overview

Chapter 2 of this report describes the audit data used by IDES. Chapter 3 gives an overview of the IDES design. Chapter 4 describes the main features of the statistical intrusion-detection procedures; Chapter 5 provides detail on the statistical algorithms employed by IDES. Chapter 6 describes the rule-based expert system shell built into IDES. Chapter 7 presents the IDES security administrator interface. Chapter 8 describes the target sys-

tem being monitored while Chapter 9 discusses details of the current implementation. Chapter 10 contains SRI's conclusions. Finally, Chapter 11 outlines our plans for future work. An example of IDES processing is given in the Appendix.

Chapter 2

The Audit Data

Although existing audit trails (i.e., those not designed specifically for security purposes) can be of some use in intrusion detection (as was demonstrated by Javitz et. al. [7]), specialized audit trails for security can be potentially much more powerful. Existing audit trails collect far too much data to be usefully analyzed for intrusions and do not collect much of the information that may be relevant to intrusion detection. The Sytek study, for example, had to construct its own audit-data-collection program in order to obtain relevant data to analyze [8]. For similar reasons, we designed and built special audit-data-collection software for the TOPS-20 target system. A description of the TOPS-20 target system audit-data-collection software can be found in our earlier report [1].

IDES monitors target system activity as it is recorded in audit records generated by the target system. The format of the audit records is system-independent to the extent possible. Each audit record consists of a variable number of fields and completely describes a single event. The *subject* field serves to uniquely identify the user and location of the event. The *action* field identifies what action the subject was attempting. The *object* field describes what the subject was performing the action upon. There is also an *errorcode* field, which indicates whether an error occurred for the event. A *resource-info* field quantifies the resources (e.g., connect time, CPU time) used by the subject since the current user-session began. The *time* field reports the date and time of the event.

There are two types of audit records: system audit records and *control* audit records. System audit records are those generated by the target system that describe user activity on the target system. The types of system audit

records are as follows.

- *login* - A login has been attempted.
- *logout* - A user-session has terminated.
- *location change* - A user has reconnected to a user-session, possibly from a different location. The audit record contains the name of the new location.
- *command* - One of the commands that is being monitored has been executed. The audit record contains the name of the command.
- *network activity* - Some processing involving the network has occurred on the target machine. The audit record contains the name of the foreign host involved and the type of network activity.
- *directory modification* - A user has modified a directory on the target system.
- *directory accessed* - A user has referenced a directory that requires password access. The audit record contains the name of the directory concerned.
- *system call* - A user has executed a system call on the target system.

This choice of system audit record types is not dictated by the IDES design or by the IDES audit record format. The specific selection of system audit record types to be monitored depends on the target system.

IDES cannot always infer the status of activity on the target system simply from observing the system audit records. For example, the lack of a *logout* record does not mean that a user-session has not terminated; the user-session may have terminated due to a system crash. For such events not easily reportable by the target system, we use *control* audit records. The types of control audit records are as follows.

- *user-session started* - A user-session has begun.
- *user-session ended* - A user-session has terminated normally.
- *user-session aborted* - A user-session has terminated abnormally.
- *target reloaded* - The target system has rebooted.

- *day ended* - The target system has started generating audit records for a new day.
- *hour ended* - The target system has started generating audit records for a new hour.

The user-session-related control audit records are used to define the start and end of a user-session The time-related control audit records are used to trigger regularly scheduled operations. For example, the *day ended* record causes the profiles to be updated.

Audit datagrams are used to communicate audit records from the target system to the IDES monitor. Each datagram consists of a plain-text header and an encrypted audit record. The IDES datagram header contains four decimal fields for use in processing the datagram. The first field describes the source of the datagram. It is a number, assigned by IDES, that uniquely identifies the target system; thus, the IDES protocol and audit record format can also be used when IDES monitors multiple target systems simultaneously. The second field contains the encryption format identifier that identifies the format of the encrypted datagram. The third field identifies the format of the decrypted datagram. The fourth field is the audit trail event sequence number. The sequence number is used to indicate the order of events and to encrypt the datagram.

Chapter 3

The IDES Design

IDES maintains profiles for *subjects*. (Here we are using the term *subject* slightly differently from how we use it in describing the audit records.) A *profile* is a description of a subject's normal (expected) behavior with respect to a set of intrusion-detection measures. The subjects profiled by IDES are users, groups, remote hosts, and overall target systems. *Groups* can be groups of users, of hosts, or of target systems. Profiling groups enables IDES to detect when the behavior of an individual member of a group deviates from the overall average behavior of the group. For example, although an individual clerk's behavior may be normal with respect to his or her past behavior, it may not conform to the behavior of an average clerk. Profiling a target system enables IDES to detect system-wide deviations in behavior, when such deviations cannot be attributed to a single user. For example, the number of login attempts system-wide may be indicative of an intrusion, although they might not all be attributable to a single user-id.

The profiles are periodically updated based on the observed behavior of the subjects of the target system. Thus, IDES adaptively learns subjects' behavior patterns; as subjects alter their behavior, the profiles will change. This capability makes IDES a flexible system, in that it does not have to be given "rules" determined by a human "expert" in order to learn what constitutes suspicious behavior; rather, this statistical component of IDES derives its own rules. Thus, IDES is potentially sensitive to abnormalities that human experts may not have considered.

Profile updating is performed once daily at a time when system usage is low. The procedures require a minimum amount of storage for historical data. Rather than storing all historical audit data, the profiles keep only

statistics such as counts, means, and covariances. When new audit data is used to update the profiles, the counts, means, and covariances stored in the profile are first aged by multiplying them by an exponential decay factor. This has the effect of creating a moving time window for the profile data, so that the expected behavior is influenced most strongly by the most recently observed behavior.

IDES also includes an expert system component. The expert system operates in parallel with the component of IDES that performs statistical anomaly detection. Although at present this component of IDES is not fully implemented, we intend that IDES's expert system component will contain rules that describe suspicious behavior that are based on knowledge of past intrusions, known system vulnerabilities, or installation-specific security policies. Thus, IDES will also be sensitive to known or posited intrusion scenarios that may not be easily detectable as deviations from past behavior.

IDES is driven by the arrival of audit records. IDES examines the audit records as they are received from the target system and ascertains whether the observed activity is abnormal with respect to the profiles. IDES flags as anomalous behavior that which deviates from the normal behavior by some amount; this amount is scalable by the IDES security administrator. IDES will also flag as anomalous behavior that triggers one or more of its expert system rules. When an anomaly is detected, an anomaly record is generated and recorded in the database.

One of the strengths of the current IDES prototype is its window-based administrator interface, which provides the IDES security administrator with a powerful and comprehensive view of the target system being monitored. The IDES security administrator interface is structured so that the IDES security administrator can selectively examine different types of anomalies graphically or by using a query interface. The IDES security administrator interface provides time-varying graphical displays of target system activity, as well as the ability to "zoom" on abnormal behavior by selecting from among several built-in database queries. When the IDES display shows that abnormal activity is occurring, the IDES security administrator can quickly and easily determine which user at what location is generating the anomaly. The IDES security administrator interface will also make use of the explanation facility of the expert system.

Figure 3.1 shows the structure of IDES. The IDES database consists of the following data:

- The *audit data* consist of a table of valid decrypted and unprocessed

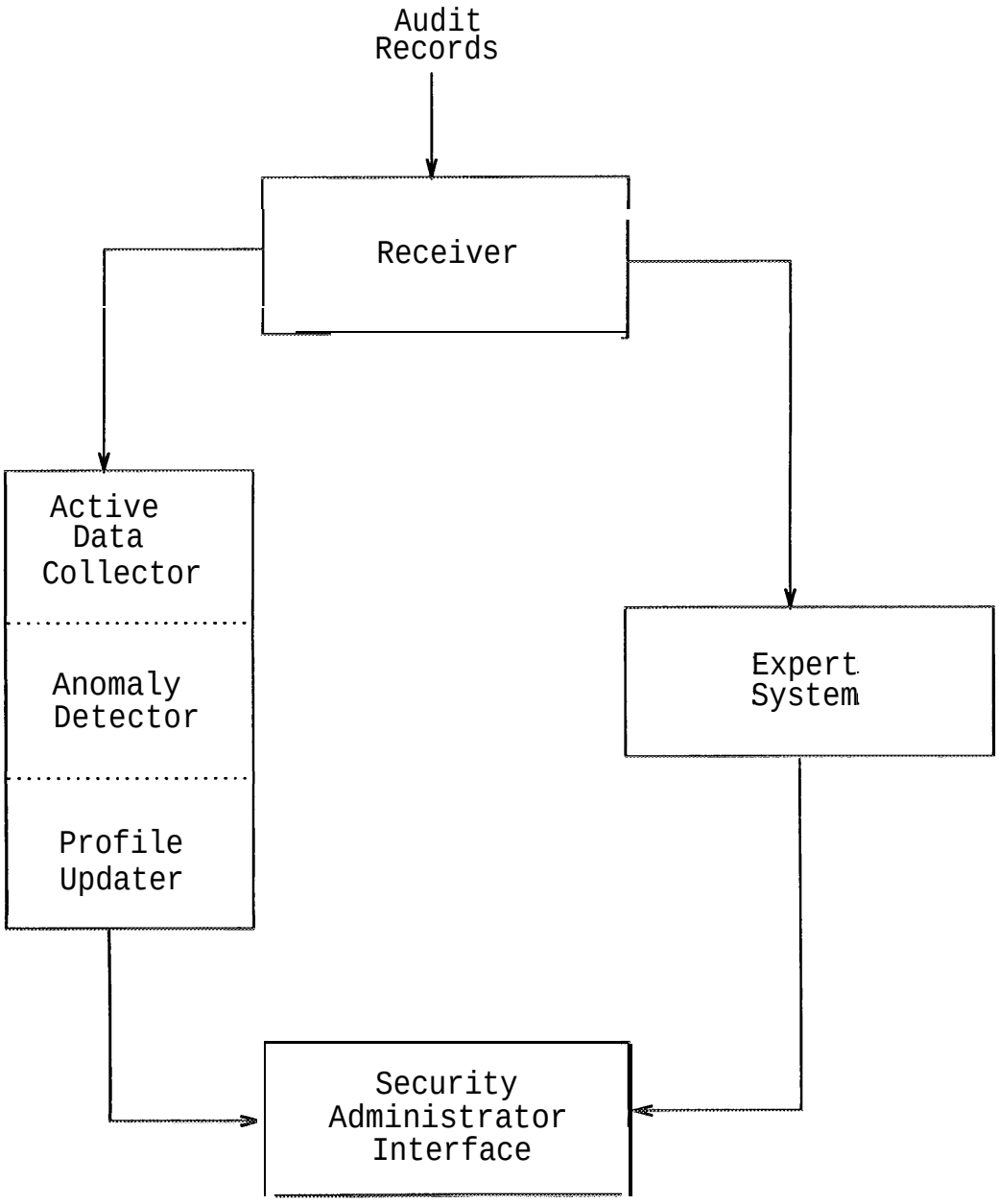


Figure 3.1: Structure of the IDES Prototype

audit records received from the target system.

- The *active data* record the accumulated activity for each user, group, remote host, and the system as a whole since the last profile update. These data are used to periodically update the profiles.

- The *archive data* consist of a table of processed audit records and a set of files. The audit records in the table are periodically removed to a new file, and the files are regularly backed up to tape.

- The *profile data* consist of several tables that define normal behavior based on past behavior for each subject. The profiles are updated daily using the most recently observed behavior (from the *active data.*)

- The anomaly *data* consist of a set of tables containing anomaly records. An anomaly record is generated when the observed current behavior deviates abnormally from the normal behavior as specified in the profiles, or when the rules specified in the expert system component have been violated.

IDES is implemented using components that communicate with each other through the IDES database.

- The *receiver* decrypts, parses, and validates each audit record as it is received, and inserts valid audit records into the *audit data.*

- The *active data collector* retrieves audit records from the *audit data* and uses them to update the *active data.* After an audit record has been processed, the *active data collector* marks it as having been processed.

- The *anomaly detector* compares incoming audit records with the profiles; if the observed behavior is considered anomalous, then an anomaly record is generated and inserted into the *anomaly data.*

- The *expert system* reads records from the *audit data* and applies each audit record in turn to its rule-base. If the observed behavior triggers one or more of the expert system rules, an anomaly record is generated and inserted into the *anomaly data.* After an audit record is processed, the *expert system* marks it as having been processed.

- The *archiver* removes records from the audit data that have been marked as processed by both the active data collector and the expert system and inserts these records into the *archive* data, and periodically backs up processed audit records from the *archive* data to an external file.

- The *profile* updater updates the profiles from the *active* data, and also recomputes the probability distributions (after appropriate decaying) for the profiles. It also provides the capability to roll back the profile to exclude data for a day containing an anomalous time segment in which a security violation did indeed occur.

- The *security administrator* interface allows the IDES security administrator to view target system activity, investigate anomalous activity, and control how IDES monitors the target system.

Chapter 4

Profile-Based Intrusion Detection

The main goal of IDES is to detect and report unusual behavior of subjects associated with a target system. This gives rise to several questions: What or who are these subjects of interest? Mow is their behavior characterized and measured? When is their behavior considered unusual? We address all of these questions in this section, and Chapter 5 answers the third question in detail.

4.1 Subjects

There are several types of subjects that an intrusion-detection system can monitor. Examples of subject types are users, remote hosts, printers, terminals, and target systems. IDES monitors the subject types that are most relevant to intrusion detection: users, remote hosts, and target systems. IDES monitors many users and remote hosts. In the current implementation, IDES monitors only one target system subject, namely, the SRI DEC-2065/TOPS-20 target system. In the next phase of our research, IDES will be able to monitor several target systems simultaneously (see Chapter 11).

4.2 The Intrusion-Detection Measures

IDES determines whether a subject's observed behavior as reported in the audit data is normal with respect to that subject's past behavior as characterized by specific intrusion-detection measures. A *measure* is an aspect of

behavior on the target system. In Section 4.2.1, we discuss the unit of time used for analysis of the intrusion detection measures. In Section 4.2.2, we introduce different types of intrusion detection measures. In Section 4.2.3, we describe particular measures that are applied to users, remote hosts, and target systems.

4.2.1 Time Segments

The unit of time during which behavior is measured is called a *time segment*. A time segment is a slice of a subject's behavior that IDES judges to be normal or anomalous. For example, one might speak of an abnormal day, hour, session, or process for a user. The length of a time segment is chosen for the subject type and intrusion-detection measure so that comparisons between time segments are meaningful. For example, if the segment is too small, say one minute, subject behavior may vary widely between successive time segments, making it difficult to establish what is "normal" for a segment. On the other hand, if the segment is too long, say one day, short bursts of abnormal activity could be masked by being grouped together with much longer intervals of normal activity. Thus, the appropriate length for a time segment depends on the intrusion-detection measures and the rates at which these are expected to vary. Taking this into account, IDES uses a session (the time period between login and logout, or between login and abnormal session termination) as the time segment for users. IDES uses an hour as the time segment for host and system subjects. These definitions can be modified easily for different choices of measures or for different target systems. For example, it might be convenient to define a time segment for users in a Sun environment as the time period between calls to the program `lockscreen`, as users might rarely log in and log out.

4.2.2 Types of Measures

IDES implements two types of intrusion-detection measures.

- A *categorical measure* is a function of some aspect of observed behavior whose range is the set of all combinations of a finite set of categories. An example of a categorical measure is the commands invoked by a user, where the range is all combinations of file names. Another example is the hour of activity by a user, where the range is all combinations of the 24 hours of a day.

- A *continuous measure* is a function of some aspect of observed behavior whose range is the set of real numbers. An example of a continuous measure is the length of a user-session. Another example is the number of lines printed by a user during a session.

IDES uses 25 intrusion-detection measures to describe the behavior of users, 6 for the behavior of hosts, and 5 for the target system. For a continuous measure, IDES records only one value for each time segment for each subject. For example, for the user-measure *errors,* IDES records the number of errors that occur during a user's session. For any user-session, the measure's value is a single number, such as 0 or 3. In contrast, for a categorical measure, IDES can record *several* values for each time segment for a subject. For example, for the user-measure *command usage,* IDES records for each user the names of the commands the user invokes during a session and the number of times each was invoked. Thus, if a user logs in, executes the `finger` command twice and the `mail` command three times, and then logs out, IDES will record two numbers in two categories for that user-session----category `finger` will have value 2 and category `mail` will have value 3. Thus, for example, the *command usage* measure could detect if a user is using the C compiler much more frequently than usual.

A special case of the categorical measures are the *binary categorical measures,* for which each category has value 0 or 1. An example of a binary categorical measure is the user-measure *command usage: binary*. For this measure, IDES records *whether* a command is invoked by a user during a session, as opposed to *how frequently* a command is used. IDES records the value 1 for commands invoked by the user during the session and the value 0 for other commands. Whereas the *command usage* measure will detect anomalies in the relative frequencies of the commands used, the *command usage: binary* measure is expected to be much more sensitive to use of infrequently used commands. Thus, for example, the *command usage: binary* measure could detect if a user invokes a command that he or she has never used, or has not used recently.

4.2.3 User Measures

The 25 measures listed below are computed for each user for each user-session.

- **CPU usage.** This continuous measure indicates the CPU usage during the session. The value recorded is the natural log of the number

of CPU seconds consumed.

- **Input/output usage.** This continuous measure indicates the cumulative input/output usage during the session. The value recorded is the natural log of the amount of I/O activity. For TOPS-20, the amount of I/O activity is measured by the number of characters typed at the terminal.

- **Connect time.** This continuous measure indicates the elapsed time of the session. The value recorded is the natural log of the session length, in seconds.

- **Audit records generated.** This continuous measure indicates the amount of activity that occurred during the session. The value recorded is the number of audit records received during the session.

- **Shift of login.** This categorical measure records the *shift* during which the user logged in. Saturdays and Sundays are considered the *weekend* shift; 6 am to 6pm on weekdays are considered the *day* shift; and the rest of the hours are considered the *night* shift.

- **Location of use.** This categorical measure records the number of times a user attempted to connect to the target system from different locations during the session.

- **Location change count.** This continuous measure records how many times a user attempted to change location during the session.

- **Command usage.** This categorical measure indicates the number of times each command was used during the session.

- **Command usage (binary).** This categorical measure records whether a command was used during the session. This measure is similar to the *command usage* measure above except the count for each command is at most 1.

- **Mailer usage.** This categorical measure records the number of times various mailers were used during the session. The security administrator can specify which commands are to be considered mailers. The defaults for the target system are `mm`, `hermes`, and *mail*.

- **Editor usage.** This categorical measure records the number of times various editors were used during the session. The security administrator can specify which commands are to be considered editors. The defaults for the target system are `edit` and `emacs`.

- **Compiler usage.** This categorical measure records the number of times various compilers were used during the session. The security administrator can specify which commands are to be considered compilers. The defaults for the target system are `pcc, bliss, fortran, macro,` and `link`.

- **Directory modification.** This continuous measure records the number of times that a user modified a directory during the session.

- **Directories accessed.** This categorical measure indicates the number of times that directories requiring passwords were accessed during the session.

- **Directories accessed (binary).** This categorical measure records which directories requiring passwords were accessed during the session. This measure is similar to the *directories accessed* measure above except the count for each directory is at most 1.

- **Errors.** This continuous measure records the number of errors that occurred during the session.

- **Errors by type.** This categorical measure records the number of times each type of error occurred during the session.

- **Hourly use.** This categorical measure records the number of audit records received for each hour spanned by the session. This measure is similar to the *audit records generated* measure except that it has hours that the session spans as categories.

- **Hour of use (binary).** This categorical measure indicates the hours spanned by the session. It is similar to the *hourly use* measure, except the count for each hour is at most 1.

- **Day of use (binary).** This categorical measure records the days of the week spanned by the session.

- **Network activity.** This continuous measure records the number of *network activity* audit records received during a session.

- **Network activity by host.** This categorical measure records the amount of network activity that originated from different hosts during the session. This measure is similar to the *network activity* measure, except that it is categorized by the remote hosts.

- **Network activity by type.** This categorical measure records the number of different types of network activity that occurred during the session. This measure is similar to the *network activity* measure, except that it is categorized by the types of network activity.

- **Hourly network activity.** This categorical measure records the amount of network activity that occurred in each hour of the session. This measure is similar to the *network activity* measure, except that it is categorized by the hours of a session.

- **Hourly network activity by host by type.** This categorical measure records the hourly amount of network activity of each type originating from different hosts during a session. This measure is similar to *network activity* measure, except that it is categorized by the hours of a session, originating host, and type of activity.

4.2.4 Host Measures

The six measures listed below are computed for each target system for each remote host for each hour of the day.

- **Host users.** This categorical measure records the number of times each user accessed the time system from the host during the hour.

- **Activity types.** This categorical measure records the number of different types of network activity (as designated by the port number) that originated from the host during the hour.

- **Hourly use.** This categorical measure records the number of network activity audit records received each hour from the host.

- **Hourly use by type.** This categorical measure records the number of network activity audit records of each type received each hour from the host.

- **Bad login attempts.** This continuous measure records the number of bad login attempts made from the host during each host time segment (which is an hour).

- **Hourly bad login attempts.** This categorical measure records the number of bad login attempts made each hour from the host.

4.2.5 System Measures

The five measures listed below are computed for each hour of the day.

- **Bad login attempts.** This continuous measure records the number of bad login attempts made on the target system during each system target segment (which is an hour).
- **Hourly bad login attempts.** This categorical measure records the number of bad login attempts made on the target system during each hour.
- **System errors.** This continuous measure records the number of errors for the target system during each hour.
- **System errors by type.** This categorical measure records the number of errors of different types made on the target system during each hour.
- **Hourly system errors.** This categorical measure records the number of errors, categorized by hour and type, that occurred on the target-system during each hour.

4.3 Identifying Anomalous Activity

We have described the measures IDES uses to determine anomalous behavior. We now consider how IDES uses these measures to detect anomalies.

IDES maintains a profile for each subject that describes certain characteristics of the subject's typical time segment. Whenever an audit record for a subject is received, or when a subject's time segment is completed, IDES compares the observed activity for the subject during that time segment with the subject's profile description of a typical time segment for that subject. IDES uses the statistical procedures described in Chapter 5 to determine if an anomaly has occurred.

We describe three extensions to this general approach. Section 4.3.1 discusses the notion of "working status" that IDES uses to develop separate subject profiles for "on" periods and "off" periods (for example, for weekdays

and weekends). Section 4.3.2 discusses how IDES can detect an anomaly even before a time segment has been completed; thus IDES does not have to wait for a user to log out before it can determine that his or her user-session is anomalous. Section 4.3.3 discusses how IDES profiles *groups* of subjects; thus IDES can detect subject behavior that deviates from the typical behavior for the other subjects in any group to which it belongs. Sections 4.3.4 and 4.3.5 discuss when anomaly detection is performed and how new subjects are monitored.

4.3.1 Working Status

It may not be desirable to consider all of a subject's time segments to be comparable to one another. For example, a host may show lots of activity between 7 am and 7 pm and very little at night, and a user may show one type of behavior on weekdays and another typical behavior pattern on weekends and holidays. Using the observed activity from all of a subject's time segments to update the same profile would therefore make anomaly detection much more difficult, because the subject's behavior described by the profile would be multimodal. IDES therefore allows the security administrator to partition each subject's time segments into separate *working statuses,* each of which are profiled separately. Each time segment is identified with exactly one working status, determined by the termination time of the time segment. For example, a user's session is either a weekday session or a weekend session, depending on the time during which the session occurred. When performing anomaly detection, IDES compares the observed data for the subject's current time segment to the subject's profile with the corresponding working status. Thus, for example, IDES compares a user's weekend activity to that user's weekend profile, and not the weekday profile. We can view the working status partition as splitting a subject into multiple subjects, one for each applicable working status.

For each user, the IDES security administrator can assign each day of the week a working status, and the working status for a user's session is simply the working status of the day on which the session occurred. For example, a user might be assigned working status 0 on Mondays through Fridays and working status 1 on Saturday and Sunday, which means that IDES keeps separate profiles for workdays and weekends for the user. As another example, if a user normally works only on Mondays and Tuesdays, then the security administrator could assign that user working status 0 for Mondays and Tuesdays and working status 1 for all other days. In addition,

the security administrator can also specify the working status for each user on "holidays" -the holidays are defined by the security administrator and are treated uniformly across all users. The holiday working status takes precedence over the weekday working status. So, for example, the working status for a session on Thanksgiving would be the status recorded for that subject under holidays, and not that under Thursdays. If no working statuses are explicitly assigned by the security administrator for a user, IDES assumes that the user has working status 0 on Mondays through Fridays, and working status 1 on Saturdays, Sundays, and holidays. The default holidays are New Years' Day, Presidents' Day, Memorial Day, Independence Day, Labor Day, Thanksgiving Day, and Christmas Day. (The dates for these need to be supplied by the IDES security administrator.)

For each host, and for the target system, the IDES security administrator can assign each of the 24 hours of the day a working status. If these are not specified, IDES assumes that hours 7 am to 7 pm have working status 0 and hours 7 pm to 7 am have working status 1.

Some discretion should be used in assigning working statuses to subjects. For hosts, for example, it is possible to assign the same status to every hour and have all hours treated uniformly, or, at the other extreme, to assign each hour of the day a different working status. Some intermediate strategy is usually preferable. In general, if there are too few working statuses (i.e., there are regular expected variations in the time segment data that are not made known to IDES), then anomaly detection will be impeded since the variance of the measures will be large. Similarly, anomaly detection will be impeded if too many working statuses are identified, as the reduced amount of data contributing to each profile will increase the length of time it takes for a profile to become stable enough for reliable anomaly detection.

4.3.2 Extrapolation of Time Segments

If a complete description of a time segment, for example, a user-session, is needed for comparison with the profile values, then anomaly detection cannot occur until a time segment has been completed. This is not sufficient, for example, when an intruder initiates a five-hour session during which sensitive data are compromised or valuable information assets are damaged early in the session. IDES has been designed so that anomaly detection can occur after any and every action a subject takes. IDES performs anomaly detection upon receipt of *every* audit record. So, for example, the intruders' behavior in the above scenario would be analyzed after every command, and

the anomalous activity could be detected early in the session.

IDES accomplishes this real-time anomaly detection by extrapolating the statistics for incomplete time segments into statistics for complete time segments and then performing the statistical anomaly detection tests on the resultant extrapolated time-segment data. Chapter 5 provides details of the extrapolation algorithm that is used.

4.3.3 Subject Groups

Another extension that IDES has made to profile-based anomaly detection is to allow subjects to be identified as belonging to one or more *groups*. Subjects can be assigned to groups by the IDES security administrator. Normally, subjects would be assigned to groups because they are expected to behave similarly to each other. For example, the user population could be divided into groups by job, by expertise with the computer, by department within a company, by geographic office location, or by security clearance. The user population could be assigned to groups along several dimensions. For example, a user could be assigned to the group of system programmers, to the group of first-line managers, and also to the group who work the night shift. Hosts might be grouped as file servers, gateways, UUCP hosts, Arpanet hosts, local to the organization, external to the organization, etc. If a subject's behavior deviates from the norm established by *any* of the groups to which it belongs, that subject's behavior will be flagged as anomalous. One can think of the group profile as enforcing a degree of "socially acceptable" conformance to group norms.

The group concept has several motivations. First, a subject's activity may be consistently abusive, making it impossible to detect the fact that the subject's behavior is suspicious by simply comparing it to the subject's past behavior. Secondly, a subject's behavior may be so erratic that, taken individually, its variance in behavior is too high for IDES to be able to detect an anomaly. Third, a subject may be active too infrequently for IDES to be able to develop a reliable profile for the subject. Finally, for some installations, there may be far too many users (perhaps in the tens of thousands) to feasibly keep profiles for all of them individually. All of these problems can be addressed by comparing a subject's activity to the profile for the group(s) to which it belongs.

IDES does not require that every subject belong to a group. In addition, when the user population is extremely large, IDES allows the security administrator to designate that some or all subjects are to be profiled *only*

as group members, and not individually.

Group profiles are also useful when IDES encounters a subject for which no individual profile exists, for example, a newly registered user. The IDES security administrator can select a group profile for a subject's initial default profile (see Section 4.3.5 for a discussion of initial default profiles).

IDES compares each subject's activity not only to the subject's profile, but also to the profile of every group to which the subject belongs. A group is a collection of subjects of the same type (e.g., a group of users or a group of hosts). IDES updates a group's profile using the observed behavior of all of the subjects belonging to that group. Thus, we can think of a group as being a subject whose activity is the sum of the activity of all of the subjects belonging to that group. For example, a group might contain all users, in which case comparison against it would be akin to comparing a user with the average user.

Groups, like subjects, can have separate profiles corresponding to different working statuses. The security administrator can identify the working statuses for each group-host groups have working statuses for each hour, and user groups for each weekday and for holidays. The default working statuses for groups are the same as for subjects of the same type. When a subject's time segment is compared with a group profile, the group profile selected for the comparison has working status that matches that of the time segment according to the subject status information, rather than to the group status information. For example, if a user's Monday session is being compared to a group profile, and the user has working status 0 on Mondays while the group has working status 1 on Mondays, then the group profile for working status 0 will be used for comparison with the session data.

4.3.4 Frequency of Anomaly Detection

Anomaly detection is done for all subjects as each audit record arrives. When IDES receives an audit record, it performs anomaly detection for the corresponding user, using its session id, subject id, working status, and observed values for the measures; the observed values are compared against the user's profile with the appropriate working status. Then, for each group to which the user belongs, anomaly detection is done using the group id, the user's working status, and the user's observed values. (Groups are discussed in Section 4.3.3.) IDES compares the user's observed values with the group's profile for the appropriate working status.

Anomalies related to hosts are checked if the audit record belongs to a

session originating from a foreign host. If the host belongs to groups, then anomaly detection is also performed for each group as described above for users.

Anomaly detection is performed for system measures if the audit record affects any of the system measures. Although a system can also belong to groups, because there is only one instance of target system in the current implementation, no group anomaly detection is done.

4.3.5 Initial Default Profiles

Default profiles are used for new subjects for whom IDES has no observed history of activity. These default profiles are used during an initial "training" period (as IDES is "training" itself on the subject) until there is sufficient observed activity for the subject's profile to be meaningful. The length of this initial period depends on the frequency of the subject's activity and can be set by the IDES security administrator. Default profiles are discussed further in Chapter 5.

4.4 Deactivating and Reactivating Measures

Different intrusion-detection measures may be appropriate to different classes of subject. For example, for users whose computer usage is almost always during normal business hours, an appropriate measure might simply track whether activity is during normal hours or off hours. However, other users might frequently log in during the evenings as well, yet still have a distinctive pattern of use (e.g., logging in between 7 and 9pm but rarely after 9 or between 5 and 9); for such users, an intrusion-detection measure that tracks for each hour whether the user is likely to be logged in during that hour would be more appropriate. For still others for whom "normal" could be any time of day, a time-of-use intrusion-detection measure may not be meaningful at all. IDES allows the security administrator to activate or inactivate specific measures for particular subjects.

4.5 Section Summary

In this section, we described how IDES detects subjects' departures from historically established behavior patterns. IDES monitors three types of subjects: users, remote hosts, and target systems. In addition, IDES profiles

groups of subjects, so that it can detect when a subject's behavior deviates from the pattern established by the group(s) to which it belongs. For users, the time segment of analysis is a session, whereas for remote hosts, target systems, and groups it is an hour. IDES implements 25 intrusion detection measures for users, 6 measures for remote hosts, and 5 measures for target systems. Some measures are continuous, with values that increase monotonically throughout a time segment, and some measures are categorical, with values taken from a finite, discrete set. IDES has knowledge of the working status of each subject, and for every subject IDES keeps separate profiles for each working status. IDES detects anomalous behavior by comparing the observed subject activity with the subject profiles. IDES uses extrapolation of time segments to detect anomalies as they occur without having to wait for the completion of a time segment. IDES allows specific intrusion-detection measures to be deactivated and reactivated for particular subjects.

Chapter 5

Statistical Procedures

In this section we describe the statistical procedures that are used to detect anomalies and maintain profiles. First we give a descriptive overview of how IDES applies its statistical tests. Second, we describe the structure of active data which is collected for a subject on a per measure basis. Third, we describe the structure of the subject profiles for continuous and categorical measures. Fourth, we discuss when and how subject profiles are aged and updated. Lastly, we describe the statistical tests that are used to detect anomalous behavior of subjects.

5.1 Overview

Before describing the IDES statistical procedures in detail, we first present an overview of what these procedures are and how they are applied. Each time segment is described by a vector of values, one for each intrusion-detection measure. The statistical procedures operate on this vector to detect anomalies for a time segment. Because some of the intrusion-detection measures that are used by IDES are categorical (for example, *location of use* is a categorical measure, whereas *connect time* is a continuous measure), a transformation is used to convert categorical measures into continuous measures. This transformation is necessary so that IDES can treat both categorical and continuous intrusion-detection measures uniformly when applying statistical tests. For a given subject and categorical measure, each category has a probability associated with it. This probability reflects the frequency of occurrence of that category, when compared with the other categories for the particular categorical measure (for the given subject). The value of

the continuous counterpart of a categorical measure is calculated in such a way that the contribution of a category is inversely proportional to its relative probability of occurrence for that measure, for a given subject. For each audit record, IDES compares the vector of values of intrusion-detection measures for the subject's time segment (that the audit record denotes) with the corresponding vector of profile values.

The intrusion-detection algorithm works as follows. If the point in *N*-space defined by the vector of values of intrusion-detection measures is sufficiently far from the point defined by the expected values stored in the profiles, with respect to the historical covariances for the measures stored in the profiles, then the record is considered anomalous. Thus, the statistical procedures pay attention not only to whether the observed value for a particular measure deviates too much from its expected value, but also to whether any observed value deviates abnormally relative to the observed values for other measures. Thus, IDES evaluates the total pattern of usage, not just how the subject behaves with respect to each measure considered independently.

To be useful, IDES must maximize the true *positive rate* (percentage of intrusive behavior correctly identified as abnormal) and minimize the *false positive rate* (percentage of normal behavior incorrectly identified as abnormal). Although the false positive rate can be reduced by raising the threshold of the statistical test (so that fewer events are considered abnormal), this also lowers the true positive rate. The acceptable false positive rate depends on the number of subjects (users) and on their rate of activity. We plan to make this parameter adjustable by the IDES security administrator (see Chapter 7). The appendix contains an example illustrating how IDES applies the statistical procedures we describe below.

5.2 Structure of the Active Data

A subject's active data is characterized by the following three items.

- The *count* vector (Count).
- The *sum* vector (Sum).
- The *cross-product* matrix (Cprod).

If there are *m* intrusion-detection measures (categorical and continuous) associated with a given subject type, then the count vector and the sum

vector for each subject are each of length m, and the cross-product matrix for each subject has dimension $m \times m$.

Each element of the count vector for a given subject contains the number of complete time segments since the corresponding profile was last updated. The count associated with measure x since the last profile update is denoted by Count(x).

Each element of the sum vector for a given subject contains the sum of continuous values (for the measure) for all complete time segments since the corresponding profile was last updated. (For categorical measures, we use the corresponding transformed continuous measures, as defined below.) We define Sum(z), the sum of values for measure x observed since the last profile update, as follows, where $v_i(x)$ denotes the value of measure x in the ith time segment since the last profile update:

$$Sum(x) = \sum_{\forall \text{ time segment } i} v_i(x).$$

For categorical measures, the value for the ith time segment since the last profile update is computed as follows. Given a categorical measure x, for time segment i, a list of numbers of occurrences by category is recorded in the active data. Let $Occur_i(k(x))$ denote the number of occurrences for category k associated with measure x for time segment i, and let Metric($k(x)$) (which is described in Section 5.3) denote a value associated with measure x and category k. The value $v_i(x)$ is derived using the following formula:

$$v_i(x) = \sum_{\forall \text{ categories } k(x)} Occur_i(k(x))\, Metric(k(x)).$$

Each element of the cross-product matrix for a given subject contains the sum of products of continuous values of two measures (identified by the row and column of the matrix) for all complete time segments since the corresponding profile was last updated. Given measures x and y such that $v_i(x)$ and $v_i(y)$ refer to the respective values of the ith time segment since the last profile update, the cross-product matrix entry Cprod(x, y) is defined as follows:

$$Cprod(x, y) = \sum_{\forall \text{ time segment } i} v_i(x) v_i(y).$$

5.3 Profile Structure

The profile of a subject is characterized by the following four items.

- The *effective count vector* (Effn).
- The *mean vector* (Mean).
- The *covariance matrix* (Cov).
- The *inverse of the covariance matrix* (InvCov).

If there are m intrusion-detection measures (categorical and continuous) associated with a given subject type, then the effective count vector and the mean vector for each subject are each of length m, and the covariance matrix and inverse covariance matrix for each subject are each of dimension $m \times m$.

Each element of the effective count vector for a given subject contains the number of complete time segments observed for a given measure since the measure has been used for the subject. The element associated with measure x is denoted by Effn*(x)*.

Each element of the mean vector for a given subject contains the historical mean value for a given measure. For measure x, the mean is denoted by *Mean(x)*.

The covariance matrix reflects the relationships between the measures. If the values for measure x and measure y in a time segment tend to be directly proportional, then the *(x, y)* entry in the covariance matrix will be a large positive value. On the other hand, the entry will be a large negative value if the measure values tend to be inversely proportional. If the measures are unrelated, the covariance value will be close to zero. Formally, if two measures x and y have values $v_i(x)$ and $v_i(y)$ for the *ith* time segment, then the covariance Cov*(x,y)* between the two measures after n time segments where n = min *(Effn(x), Effn(y))* is

$$\mathrm{Cov}(x, y) = \frac{1}{n}\left(\sum_{i=1}^{n} v_i(x)v_i(y)\right) - Mean(x)\ \ Mean(y).$$

The inverse covariance matrix is the inverse of this matrix.

The formulas above for calculating the mean vector and covariance matrix are implemented with modifications for profile aging, which we discuss in Section 5.5.1.

Additional profile items are necessary to aid transformation of each categorical measure to its continuous counterpart. The two profile items that have to do with categorical measures are

- *Total occurrences.*
- *Metric.*

Both of these are recorded for each category for each categorical measure.

The total occurrences associated with a specific measure and category are the sum of the values for that category, where the sum includes all time segments since the subject has been profiled for the measure. For example, the total occurrences for the *editor usage* measure and category `emacs` would be the total number of times the subject in question has ever used the `emacs` editor.

The metric is a number that in some sense reflects how rare a category is. The metric is used in two ways.

1. The conversion of time-segment information for categorical measures to corresponding information associated with equivalent continuous measures. (This was discussed in Section 5.2.)

2. The conversion of profile information for categorical measures to corresponding information associated with equivalent continuous measures. (This is described in Section 5.4.)

The metric for a category of a measure (for a subject) is computed as follows. Let Metric*(k(x))* denote the metric associated with category *k* of measure *x*. Let *TotOccur(k(x))* denote the total number of occurrences of category *k* for measure *x* (i.e., *k(x))* since the measure has been profiled for the subject:

$$Metric(k(x)) = -\log_e \left(\frac{TotOccur(k(x))}{\sum_{\forall \text{ categories } c(x)} TotOccur(c(x))} \right).$$

Note that the expression inside the parentheses gives the relative probability of a category occurrence. Thus, the metric can be thought of as the *log* transformation of the probability distribution of categories. This value will generally be high for unusual categories (e.g., an editor that is rarely used by the subject), and near zero for more common categories. It will never be negative.

5.4 Conversion of Categorical to Continuous Profiles

In order to apply the statistical tests described in Section 5.7, it is necessary to convert the profiles of categorical measures to profiles of equivalent continuous measures. The goal of the conversion is to compute, for a subject, a single value for a categorical measure that somehow reflects the distribution of the values over the categories. This can be done by multiplying the total number of occurrences of each category by the profiled metric for that category and then summing all these products together. Thus, if there are many values in categories with high metrics, the resulting single value will be high, whereas many values in low-metric categories will generally result in a low single value. The single value $v(x)$ for categorical measure x is derived as follows:

$$v(x) = \sum_{\forall \text{ categories } k(x)} Occur(k(x))\ Metric(k(x)).$$

If the metric for a category is not available in the profile, then a default metric of 6.901 is used. This corresponds to a relative probability of 0.001 for the category, where the relative probability of a category $c(x)$ is

$$\frac{TotOccur(c(x))}{\sum_{\forall \text{ categories } k(x)} TotOccur(k(x))}.$$

5.5 Profile Update

The profiles for all subjects are updated every day. This involves two procedures—aging the old profile data and incorporating the active data, that is, data that have been accumulated since the last time the profile was updated.

5.5.1 Profile Aging

A *decay factor* is applied to the profile data (the profiles are "aged") to give them a half-life of 50 days. So that the profiles reflect a moving time window of behavior and are influenced most strongly by the subject's most recent behavior. Specifically, the effective counts are multiplied by a decay factor of 0.9862 every day; the value 0.9862 was chosen so that the data will have a half-life of 50 days. In other words, after 50 days the audit data values

will contribute only half as much to the profile values as do the new audit data. (We plan to allow this half-life to be adjustable by the IDES security administrator.) A subject may have many profiles (for different working statuses), so only those corresponding to working statuses that occurred on the day of the update are aged. Thus, for example, weekend profiles are aged only on weekends, and weekday profiles are aged only on weekdays.

The following items are aged by a decay factor of $\gamma = 0.9862$ using the formulas below for all subjects and each measure *x*.

- The effective count is aged as follows:

$$Effn(x) = \gamma Effn(x).$$

- If *x* is a categorical measure, the total number of occurrences per category are aged as follows:

$$\forall \text{ categories } k,\ TotOccur(k(x)) = \gamma TotOccur(k(x)).$$

5.5.2 Incorporating Active Data

After the profiles are aged, the active data collected since the last profile update are incorporated into the profile data.

- The new effective count for measure *x* is computed as follows. Note that Effn*(x)* is the current effective count, which has been aged.

$$newEffn(x) = Effn(x) + Count(x).$$

- For each categorical measure *x* and category *k(x),* the new total number of occurrences per category is computed as follows. Note that TotOccur*(k(x))* is the current number of occurrences, which has been aged:

$$newTotOccur(k(x)) = TotOccur(k(x)) + Occur(k(x)).$$

- The new mean for measure *x* is computed as follows:

$$newMean(x) = \frac{Mean(x)\ Effn(x) + Sum(x)}{Effn(x) + Count(x)}$$

- The new covariance entry for measures x and y is computed as follows. Let *mincount* be the minimum of Count*(x)* and Count*(y)* and let *mineffn* be the minimum of Effn*(x)* and Effn*(y):*

$$newCov(x,y) = \frac{Cov(x,y)mineffn}{Effn(x) + Count(x)} + \frac{Cprod(x,y) - (Sum(x)Sum(y)/mincount)}{mineffn + mincount}$$

The computations are done efficiently in terms of both the time and memory they use.

5.6 Restoring Past Profiles

Not all activity that IDES flags as anomalous actually represents an intrusion or abuse. Some of the activity flagged by IDES as anomalous will actually be false alarms. IDES does not assume that anomalous behavior represents a violation, and thus the behavior IDES flags as anomalous is used, as is the normal data, to update subject profiles. When IDES flags an anomaly, it inserts an anomaly record in the anomaly table. It is up to the IDES security administrator to determine whether the anomalies flagged by IDES are actually intrusions. This is not expected to be a real-time activity; however, it is reasonable to expect that a determination would be made within a day of when the anomaly has been flagged. By the time the security administrator does determine that an anomaly is an intrusion, it may already have been used to update the subjects' profile. Thus, a capability is needed to restore the (immediate) past profile of the same working status, thereby removing the effect of the anomalous activity. Because an intruder may have altered data pertaining to several different subjects, IDES offers the security administrator the capability to restore *all* the profiles to their previous state. This restoration process is initiated by the security administrator.

5.7 Tests for Anomaly Detection

Finally, we describe the tests that identify anomalous behavior. Anomaly detection has two phases. First, a *composite* test is applied to determine whether the combination of observed values for all the measures is anomalous. The second phase, which occurs only if the first phase detects an

anomaly, involves determining which of the intrusion-detection measures contributed most to the anomaly.

If the subject has no profile (for the relevant working status), the audit record is flagged as anomalous.

5.7.1 The Composite Test

The test that IDES critically depends upon to detect anomalies is the so-called *t-test*. This test uses the profiled means and covariances to determine if the observed activity for a given time segment i (i.e., $\forall x, v_i(x)$) is abnormal. (See Section 5.2 for description of $v_i(x)$.) By using the covariance data, the t-test takes into account the degree to which the values of the measures vary, on their own and with respect to the other measures. The test works by computing a statistic, called *t2,* that reflects the deviation of the time-segment vector from the mean vector. If we let X be the continuous measure vector for a certain time segment, let $\overline{X}$ be the mean vector from the corresponding subject's profile, and let C^{-1} be the inverse covariance matrix, then the test statistic *t2* that is computed for the time segment is (using matrix multiplication)

$$t2 = (X - \overline{X})\ C^{-1}(X - \overline{X})^{T}.$$

This statistic has a known distribution, which means, for example, that one can determine a range within which *t2* will lie with 95 % probability. IDES uses this information to detect when a time segment is abnormal- IDES signals an anomaly when the statistic is outside the 95 % probability range. We plan for this threshold to be adjustable by the IDES security administrator.

5.7.2 Finding the Cause of the Anomaly

The test just described is called a composite test because it takes all the measures and their relationships with one another into account. If a composite anomaly arose out of comparison with data in the existing profile (as opposed to arising because no profile existed), then certain individual measures may also be anomalous. So that the IDES security administrator can assess the cause of the anomaly, the measures that contributed most to the composite anomaly need to be determined and reported to the security administrator.

An individual test on a measure is conducted by looking at the standardized value for that measure. The standardized value, also called the *z-value,* is the distance of the value from the mean, divided by the standard deviation (the square root of the variance) for the measure. That is, if x is the value for the measure for the time segment being tested, $\bar{x}$ is the mean value found in the profile, and c is the covariance of the measure with itself (these diagonal entries in the covariance matrix are also called *variances),* then the standardized value z is

$$z = \frac{|x - \bar{x}|}{\sqrt{c}}.$$

Currently, the top five measures (if any) whose normalized observed values exceed 2.0 are recorded as having contributed to the anomaly.

5.7.3 Extrapolating Incomplete Time Segments

So that IDES can detect anomalous time segments with each audit record, rather than having to wait until the time segment is completed, IDES extrapolates the statistics for incomplete time segments into projected statistics for complete time segments and then performs its anomaly-detection tests on the resultant extrapolated time-segment data. The extrapolation algorithm substitutes the historical mean values for (as recorded in the subject's profile) measures whose values are below their mean. For measures whose values are above their recorded mean, the current observed values are used in the anomaly-detection analysis. The effect of this extrapolation procedure is to allow unusual activity that results in one or more unusually high measure values to be detected before a time segment is completed. Replacing the low values with their means is necessary because otherwise most time segments would be anomalous until roughly their midpoint. For example, if a user's mean connect time is historically 3 hours, then for the first few hours of each session, that user's connect time would be anomalously low; however, this is not necessarily cause for suspicion provided the user continues to be logged in. However, if the session terminates with an abnormally short connect time, IDES reports this as an anomaly, because it is unusual for that user.

An alternative extrapolation algorithm, linear extrapolation, would allow both projected low *and* high values to be flagged before the end of a time segment. Linear extrapolation assumes that the values for a measure grow linearly throughout a time segment, and thus values for completed time segments can be estimated from values for incomplete time segments

if the length of the time segment is known. This procedure was rejected because (1) there is no reason to believe that the measures grow linearly, (2) it is not clear whether to measure time-segment length in terms of elapsed time or the number of audit records received, and (3) the error introduced in estimating the length of time segments is likely to be large.

5.8 Section Summary

In this section, we have described the statistical procedures used by IDES to detect anomalous subject behavior. We described the structure of the active data and profiles for subjects. For all measures (continuous and categorical), the profile consists of historical counts, historical means, and entries in the covariance and inverse covariance matrix. For categorical measures, the profile consists of the historical number of occurrences for each category and associated metrics. The profiles for each subject and each measure are aged and updated at the end of each day to ensure that the profiles are most strongly influenced by the most recently observed behavior. If the IDES security administrator determines that an anomaly flagged by IDES is in fact an intrusion, then he or she can restore the subject's profile or profiles of all subjects to that of the previous day. We described the composite anomaly-detection test and the individual anomaly-detection test. IDES uses the composite test to determine if observed activity in a given time segment for a subject is anomalous. IDES uses the individual test to determine which aspects of the subject's behavior are anomalous when the composite test detects anomalous subject behavior. Finally, we describe an extrapolation algorithm that enables IDES to perform anomaly detection on incomplete time segments.

Chapter 6

Rule-Based Intrusion Detection

There are obvious difficulties with attempting to detect intrusions solely on the basis of departures from observed norms for individual users. Although some users may have well-established patterns of behavior, logging on and off at close to the same times every day and having a characteristic level and type of activity, others may have erratic work hours, may differ radically from day to day in the amount and type of their activity, and may use the computer from several different locations and different time zones (in the office, at home, and on travel). Thus, for the latter type of user, almost anything is "normal," and a masquerader might easily go undetected, Thus, the ability to discriminate between a user's normal behavior and suspicious behavior depends on how widely that user's behavior fluctuates and on the range of "normal" behavior encompassed by that user. And although the "departure from norm" approach might be successful for penetrators and masqueraders, it may not have the same success with legitimate users who abuse their privileges, especially if such abuse is "normal" for those users. Moreover, the approach is vulnerable to defeat by an insider who knows that his or her behavior is being compared with his or her previously established behavior pattern-insiders who slowly vary their behavior over time, until they have established a new behavior pattern within which they can safely mount an attack. Trend analysis on user behavior patterns, that is, observing how fast and in which direction a user's normal behavior changes over time, may be useful in detecting such attacks. In addition, trends in variances in user behavior can be used to measure the rate at which a

user's normal range "spreads" over time. We are currently developing some "second order" measures to detect such trends in user behavior.

An obvious second line of defense against these weaknesses is to enforce rules that describe suspicious behavior that is independent of whether a user is deviating from past behavior patterns. We are adding to IDES the capability to characterize intrusions using expert system rules. We plan to encode information about known system vulnerabilities and reported attack scenarios, as well as intuition about suspicious behavior, in IDES's expert system rule-base. The rules are fixed in that they do not depend on past user or system behavior. An example of such a rule might be that more than 3 consecutive unsuccessful login attempts for the same user id within 5 minutes is a penetration attempt. Other examples include a login attempt for a locked account, and a status inquiry command on an account (such as finger) followed by a login attempt. Audit data from the monitored system is matched against these rules to determine whether the behavior is suspicious. A limitation of this approach, if used in isolation, is that one is looking for known vulnerabilities and attacks, and the greatest threat may be the vulnerabilities we do not yet know about and the attacks that have not yet been tried; one is in a position of playing "catch-up" with the intruders. Such an approach, used in conjunction with detecting departures from established user norms, addresses the vulnerabilities and has the strengths of both approaches.

We plan to structure the IDES rule-base into two parts. One part will be a generic rule-base that includes intrusion-detection rules that can apply to many different types of target systems and installations. One example might be that two or more login attempts within half an hour from geographically separated areas using the same user id is suspicious. The other part of the IDES rule-base will consist of a set of rules that are either operating-system-dependent or installation-specific. For example, we expect that the intrusion scenarios for a Unix system will differ from those for an MVS system. As another example, some installations may process highly sensitive data and may consider browsing among other users' directories to be a security violation; in this case, rules that are triggered by browsing would be contained in the installation-specific portion of the IDES rule-base.

Chapter 7

Security Administrator Interface

The security administrator interface consists of the following subcomponents:

- *Status monitor*
- *Query monitor*
- *Anomaly monitor*
- *Session monitor*
- *Message monitor*
- *Customization monitor*

The status monitor provides a concise summary of what IDES has been doing during a certain time interval. The message monitor is intended to provide the security administrator with a more detailed view of what happened by displaying the messages generated by various components of IDES. The session and anomaly monitors provide information at a yet more detailed level, displaying activity at the user-session level and individual anomaly level, respectively. The query monitor allows interrogation and manipulation of the IDES database at the lowest level, by accepting both ad hoc and built-in SQL queries. The customization monitor allows the security administrator to set various parameters at the system-wide or subject level to control how IDES monitor subjects.

Details regarding the status monitor, query monitor and anomaly monitor can be found in our previous report [1]. In this section, we concentrate on the capabilities offered by the session monitor, show how the security administrator can view messages generated by IDES, and describe how the security administrator can set customizable parameters.

7.1 Session Monitor

The session monitor allows the security administrator to examine user sessions.[1]

Figure 7.1 illustrates the session monitor's main window as displayed on a Sun workstation screen. The window is divided into two parts: the top part contains a menu and type-in areas that the security administrator can use to specify what he or she wants to examine; the bottom part is a display area for information returned by IDES that conform to the specification.

The top line of the specification area contains a general selection criteria: `Active Only` means display only active sessions; `Remote Only` means display only sessions that were initiated from remote hosts; `With Errors` means display only sessions that raised errors. The next line, `Limits,` allows the security administrator to supply more specific criteria by specifying which user he or she is interested in, sessions with identifiers greater than a limit (session identifiers are monotonically increasing across all users), and sessions with run time or CPU time greater than certain limits. When the *DISPLAY* button is selected, sessions fitting the selection criteria and limits are displayed in the display area, as shown in Figure 7.2. Figure 7.2a displays all active sessions, while Figure 7.2b shows all sessions of user 1200 with session id greater than 15000.

To examine a particular session in greater detail, enter the session id in the `Detail Session ID` field and select the *DETAIL* button. Figure 7.3 illustrates what the Detail window looks like. The window is split into three parts: the top part contains statistics of the session, the middle part contains a log of all the audit records generated for that session, and the bottom part contains a log of all the messages generated for that session.

[1] We plan to extend the session monitor to handle host and system time segments as well.

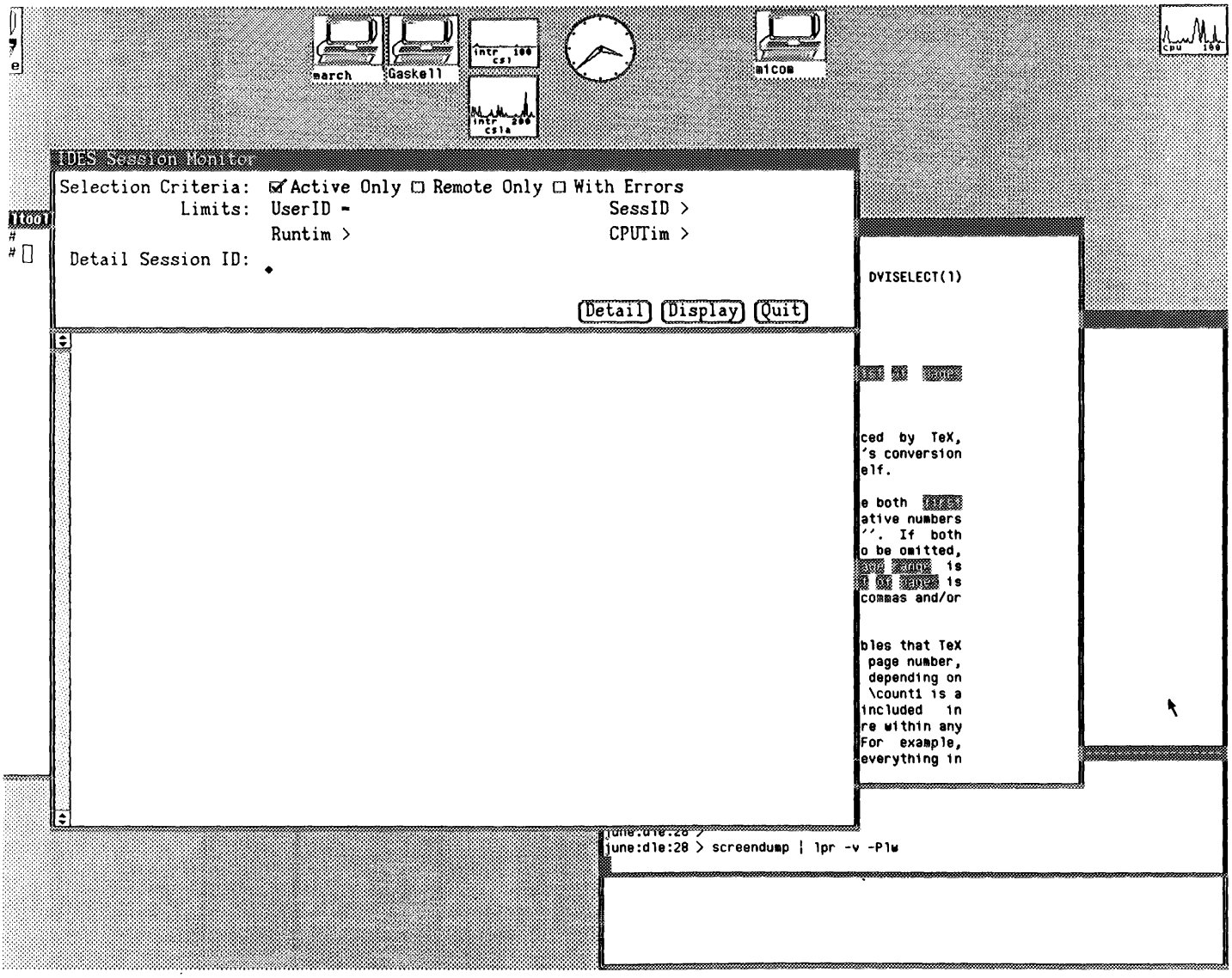
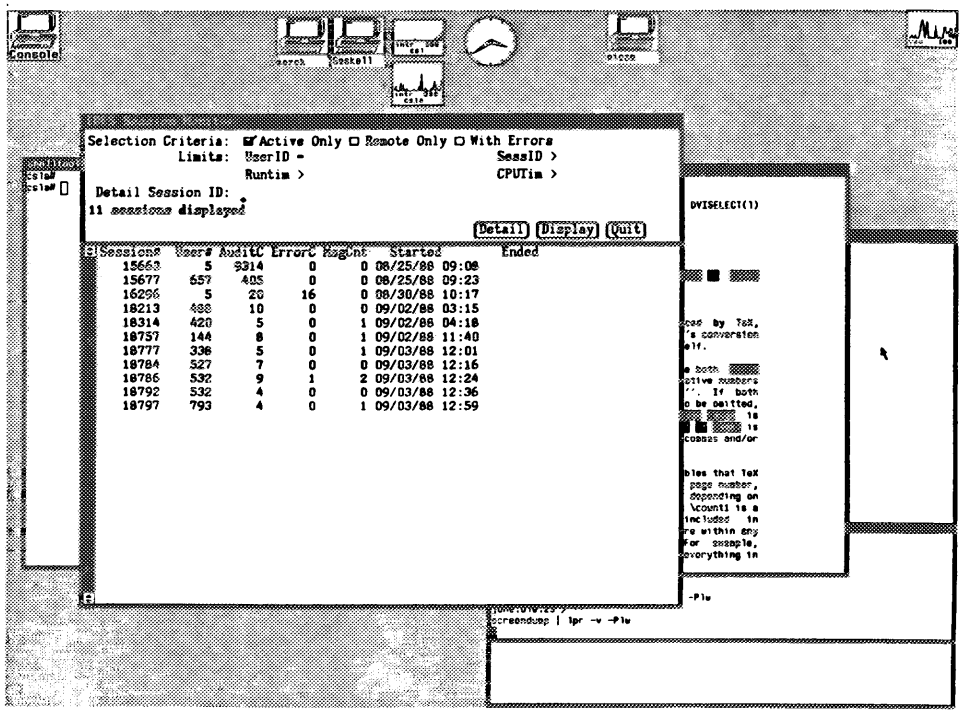
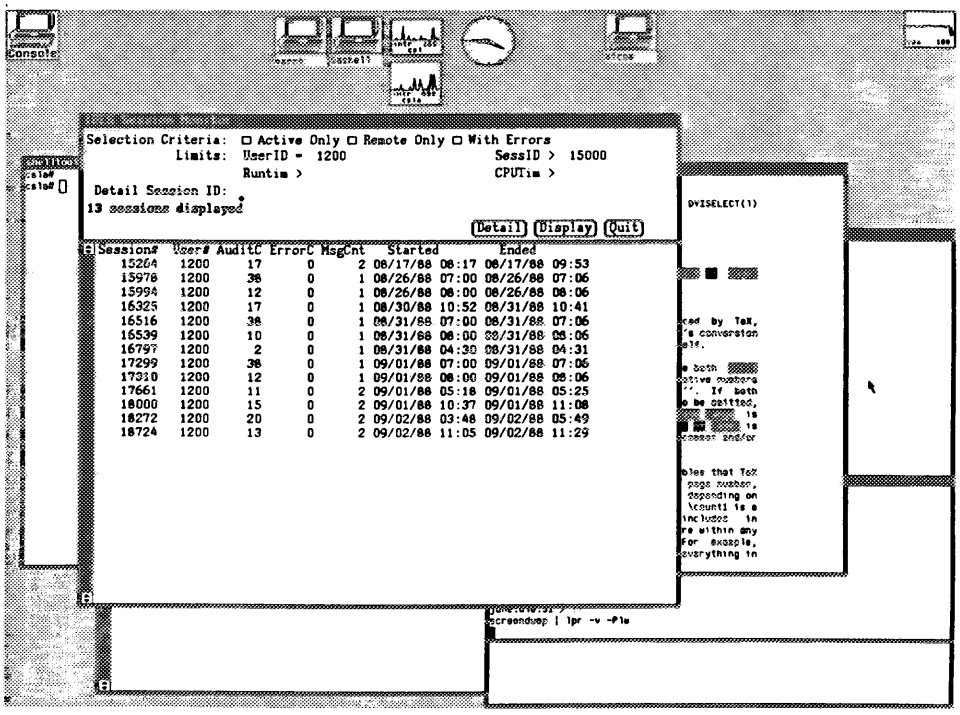


Figure 7.1: Instance of the Session Monitor



(a)



(b)

Figure 7.2: Sample Uses of the Session Monitor

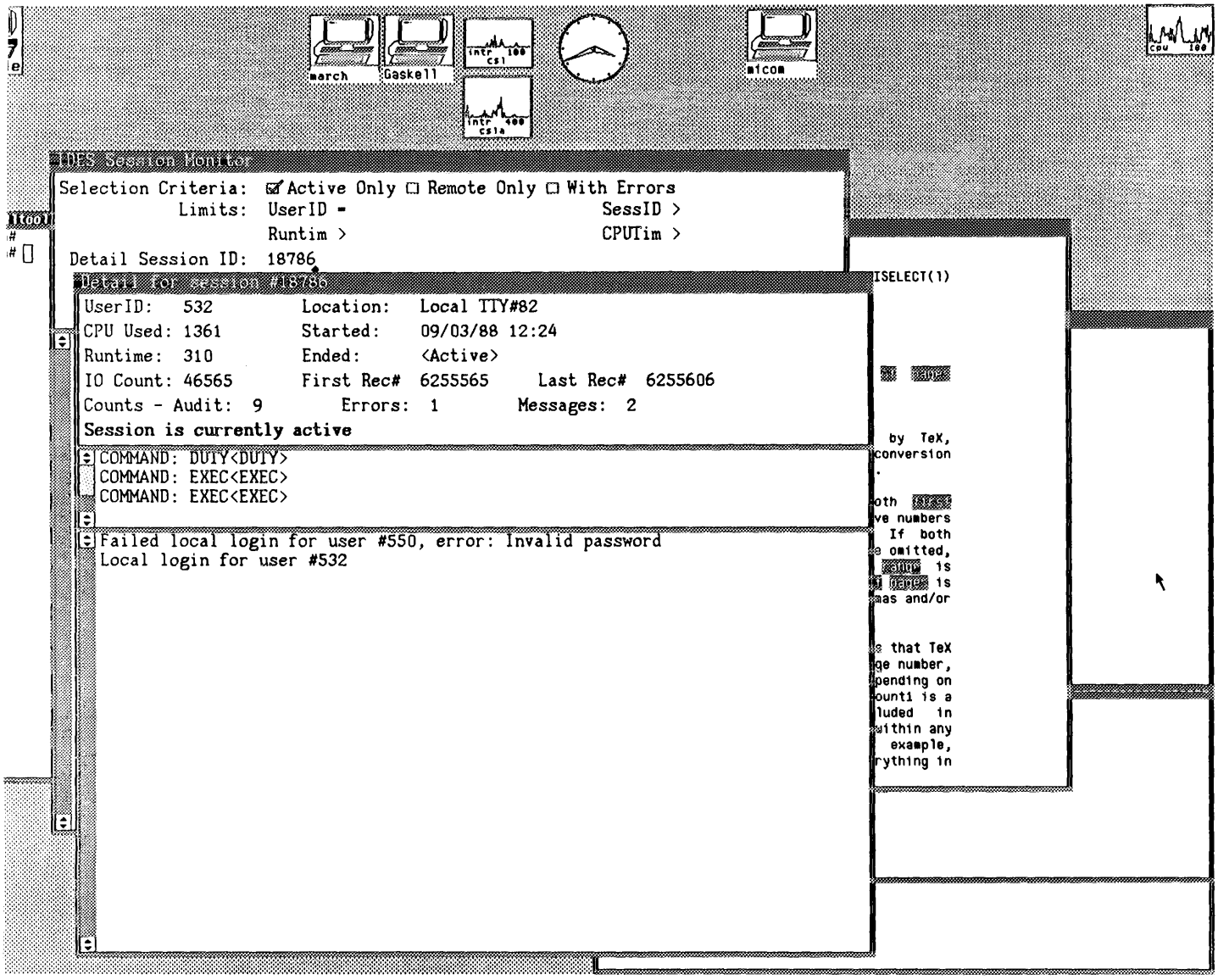


Figure 7.3: Instance of the Detail Window

7.2 Monitoring Messages

The message monitor is intended to provide the security administrator with a global view of IDES. Different components of IDES insert messages into the IDES database and the message monitor displays the messages to the security administrator. These include events such as profile update started or completed, target system reloaded, failed logins, anomalies detected, and so on. The security administrator should be able to tell what each component is doing by browsing through the messages.

We plan to build a graphical interface for the message monitor. Currently, the interface is through the query monitor, as shown in Figure 7.4.

7.3 Customizations

The parameters currently available for customization include: holidays, command classification, group membership, working hours and days, and activation and deactivation of specific intrusion-detection measures. Holidays and command classification are system-wide parameters; the rest are subject-specific. Values for these parameters are stored in tables in the IDES database and are modified and maintained by the security administrator. An example of command classification is the list of commands that invoke editors.

For the holidays table, the security administrator enters dates that are considered to be holidays for users of the target system. These dates are used to determine the working status of subsequent audit records and time segments.

IDES currently monitors three command classes: mailers, editors, and compilers. The security administrator can add or delete commands from these classes. These updated lists are used to classify commands of subsequent audit records.

The security administrator can assign subjects of the same type to *groups* by adding entries to the *GROUP* table. Selection of appropriate group identifiers that do not conflict with subject identifiers of that subject type is currently the task of the security administrator. (We plan for IDES to provide assistance with this in the future.) The updated tables are automatically used to process subsequent audit records. Profiles for new groups are created automatically during the next update period and anomaly detection based on new group profiles will commence after that update,

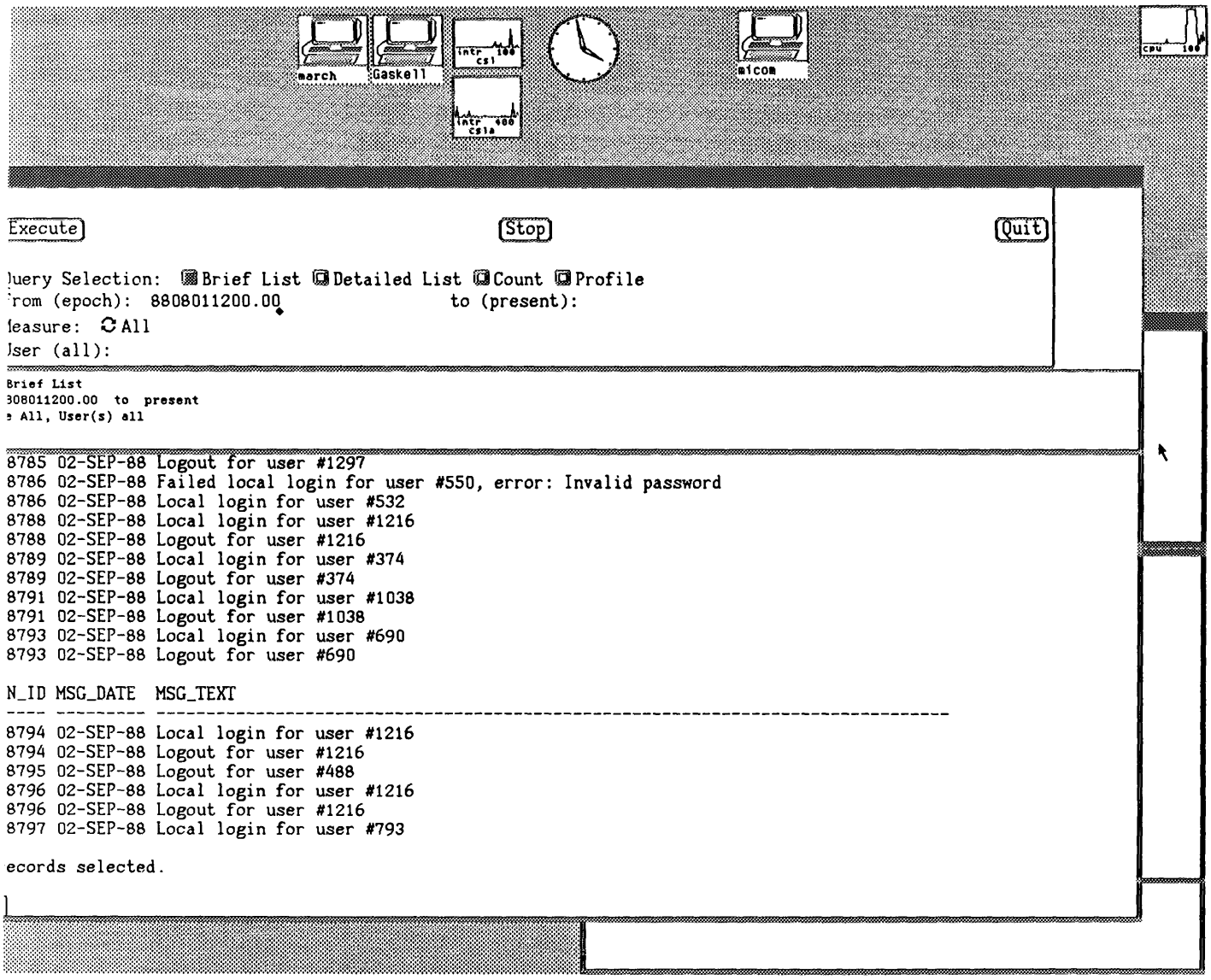


Figure 7.4: Use of Query Monitor in Monitoring Messages

The security administrator can specify for a user which days of the week are working days and whether holidays are working days. For hosts and systems, the security administrator can specify the working status on a per-hour basis. For example, the security administrator may say that user A works only during holidays while user *B* works only Monday through Thursday; similarly, the security administrator may specify that host *X* is active from 7 am to 3 pm, and 5 pm to 12 pm, (perhaps as a result of two shifts). If a subject does not have an entry in these tables, then the defaults are used (see Section 9.1).

For each subject, the security administrator can specify that specific intrusion-detection measures do not apply. By default, all intrusion-detection measures apply to a subject and these measures are weighted equally in determining whether an anomaly has occurred. With this capability, the security administrator can concentrate on measures which he or she believes are most relevant for a specific subject. For example, if the security administrator is interested in monitoring a certain user's use of commands and believes that other measures are not of relevance, he or she can turn-off monitoring of the other measures. When an anomaly is raised for that user, it would then be obvious that it was due to an abnormal use of commands; anomalies that would have arisen due to excessive CPU usage or other factors would be ignored. Monitoring for a subject for a particular measure should probably be turned off if the subject's variance for the measure is extremely high, for example. Note that although measures specified in this way are ignored for anomaly detection, their values are still retained for the subjects' profiles so that they will be available when the security administrator subsequently reactivates the measure.

We plan to supply graphical interfaces through which the IDES security administrator can set these parameters. Currently, the interface is through the query monitor, as shown in Figure 7.5.

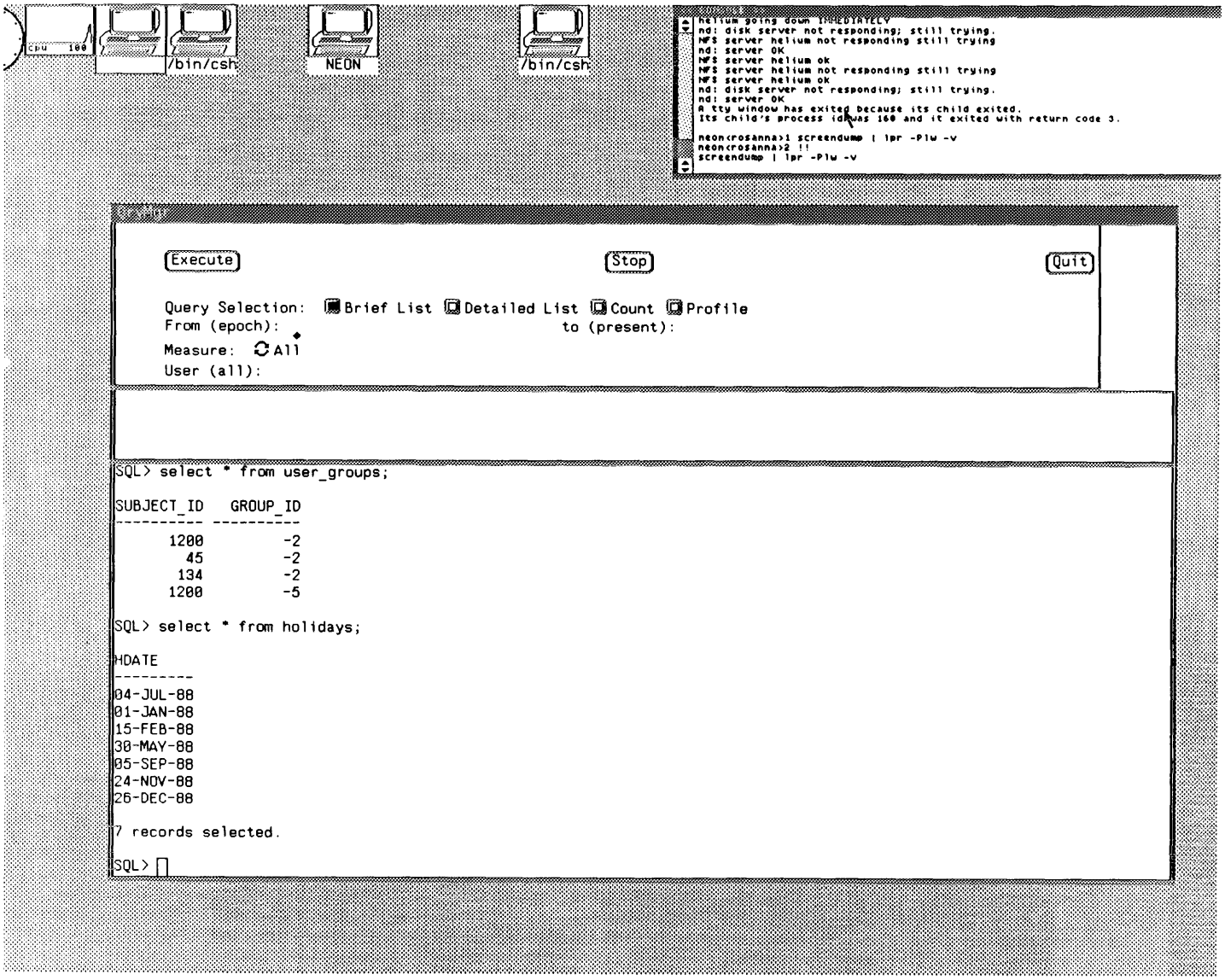


Figure 7.5: Use of Query Monitor in Customizing IDES

Chapter 8

The Target System

The target system that IDES is currently monitoring is a DEC-2065 running a locally customized version of the TOPS-20 operating system. This system "knows" about IDES to the extent that it processes the audit data into audit records, encrypts the records, and transmits them to the IDES workstations. The customizations done to the TOPS-20 system are described in detail in our earlier report [1].

Chapter 9

Implementation

9.1 The IDES Database

Figure 9.1 contains a summary of the tables in the IDES database and their use by various IDES processes.

9.1.1 Audit Data

The audit data consist of the tables *audit records, rejected audit records,* and *audit record log.* We describe the data contained in these tables, below.

Audit records

seq no	time-seg	subject	action	object	errorcode	resources	time

Seq no is a unique identifying number for the audit record. *Time-seg* is a number, assigned by the receiver, which uniquely identifies the user session to which this audit record belongs. The *subject* field is a triple *(job number, user id, location)* where *job number* is a job number assigned to the session by the target system, *user id* is a unique number identifying a user, and *location* is either a specification of a terminal line or a remote host. *Action* is the type of event that occurred on the target system that triggered generation of this audit record, *Object* is a description of the object to which the action was applied. It can be any of the following: a user identified by the three-tuple (*job number, user id, location*), a string identifying the name of a directory accessed, a string identifying an account on the target system, or

Table	*Updated by*	*Referenced by*
Audit records	receiver	data collector, interface, archiver
Rejected audit records	receiver	interface
Audit record log	receiver	data collector, expert system, interface
Continuous active data	data collector, updater	data collector, updater anomaly detector
Categorical active data	data collector, updater	data collector, updater
User segment log	receiver	data collector, interface, updater
Host segment log	data collector	data collector, interface, updater
System segment log	data collector	data collector, interface, updater
Archive record	archiver	archiver
Variance profile	updater	updater, anomaly detector
Mean profile	updater	updater, anomaly detector
Categorical probabilities	updater	updater, data collector
Anomalies	anomaly detector	interface
Command classes	interface	data collector
Workdays, Workhours	interface	data collector, updater, anomaly detector
Holidays	interface	data collector, updater, anomaly detector
Groups	interface	data collector, updater, anomaly detector
Inactive anomalies	interface	anomaly detector
Message log	archiver, interface data collector, receiver, updater, anomaly detector	interface
Status log	receiver	interface
Error log	receiver	interface

Figure 9.1: IDES Processes and the IDES Database Tables

the keyword *SYSTEM,* in which case the object is the target system itself. *Resources* contains cumulative counts of the resources used for the current user-session to which this audit record belongs. It includes the CPU usage, input/output activity, and connect time. *Time* indicates when the record was generated by the target system.

Rejected audit records

host id	enc id	version no	seq no	time	reason	contents

This table stores audit records that IDES received but which did not conform to the audit record format expected for this version of IDES. *Host id* identifies which host the record came from. *Enc id* identifies the algorithm used for encrypting the record, while *version no* refers to the version of IDES expected by the audit record.

Audit record log

seq no	time stamp	time received	time processed	time exsys

This tables stores the times at which audit records are generated by the target system *(time stamp),* when they are received by IDES *(time received),* and when they are processed by the active data collector *(time processed)* and expert system *(time exsys).* This information is separated from the *audit record* table so that the *audit record* table, which is a much larger table, does not have to be locked when the table is updated.

9.1.2 Active Data

The active data consist of values maintained for continuous and categorical variables for time segments monitored since the last profile update. In addition, logs (per subject type) are kept of time segments introduced since the last profile update to identify to which subject the time segments belong. For example, the user segment log is used to identify to which user a particular session belongs. This mapping is necessary for anomaly detection and profile update.

Continuous active data

completed	time segment	measure	value

There are three tables of this kind, one for each type of subject. This table maintains the values of continuous variables (and converted categorical variables) on a per-time-segment basis. Each entry records whether the time segment has been completed.

Categorical active data

completed	time segment	measure	category	count

There are three tables of this kind, one for each subject type. This table records the frequency of occurrence of items in categorical measures on a per-time-segment basis. Each entry records whether the time segment has been completed.

User segment log

time-seg	job no	user id	seq nos	resources	counts	flags	times

This table maps *time-seg* to *user ids*. The other information are recorded for perusal by the security administrator. *Counts* contain the number of errors occurred, the number of messages generated, and the number of anomalies generated during this time segment. *Flags* contain bits indicating whether the audit records of the time segment have been archived, whether the time segment is complete, and whether the time segment is the result of a remote login. *Seq nos* refer to sequence numbers of the first and last audit records received for the time segment. *Times* contain the timestamps of the first and last audit records received for the time segment.

Host segment log

host number	time slot	time segment

This table is used to look up to which host a host time segment belongs. *Host number* is the internet host number assigned to a host. *Time slot* is the hour in which the time segment began.

System segment log

time slot	counter

This table is used to record when a system time segment began. *Counter* records how many host time segments have occurred during that system time segment. This value is used internally for generation of host time-segment ids.

9.1.3 Archive Data

The archive data consist of the table *archive record*. Its format is identical to that of *audit record*.

9.1.4 Profile Data

Profile data for continuous (and converted categorical) variables are kept in the *variance* and *mean* tables. Profile data for categorical variables are kept in the *categorical probabilities* table.

Variance profile

subject	work stat	sq-measure	oldvar	oldinvvar	variance	invvar

There are three tables of this kind, one for each subject type. This table stores the flattened covariance *(variance)* and inverse covariance *(invvar)* matrices for the continuous measures. Copies *(oldvar* and *oldinvvar)* of the values before the previous profile update are kept so that they can be later restored if so desired. *Subject* identifies the subject to which this entry belongs. *Work stat* describes the working status of this entry. *Sq-measure* is the index of the entry in the matrices.

Mean profile

subject	work stat	measure	oldmean	oldeffn	mean	effn

There are three tables of this kind, one for each subject type. This table stores the mean value *(mean)* and the number of time segments that contributed to the mean *(effn)* associated with each continuous measure. It also stores the previous days' *mean* and the previous days' *effn* so that they can later be restored if so desired. *Subject* identifies the subject to which the entry belongs. *Work stat* describes the working status of this entry.

Categorical probabilities

subject	work stat	measure	category	oldct	oldmet	count	metric

There are three tables of this kind, one for each subject type. This table stores the count *(count)* and metric *(metric)* associated with each category *(category)* of the categorical measures. The previous days count *(oldct)* and metric *(metric)* are stored so that they can later be restored if so desired. *Subject* identifies the subject to which the entry belongs. *Work stat* describes the working status of this entry.

9.1.5 Anomaly Data

The anomaly data consist of the single *anomalies* table.

Anomalies

subject	object	action	info	use	limit	time	time-seg	seq no

There are three tables of this kind, one for each subject type. *Subject* refers to the identifier of the subject that raised the anomaly. *Action* is the type of action that produced the audit record which raised the anomaly. *Object* is the identifier of the object that the action acted upon. *Info* contains a textual description of the anomaly. *Use* and *limit* indicate the amount of activity incurred and expected, respectively. *Time* records when the anomaly occurred. *Time-seg* identifies the time segment in which the anomaly occurred. *Seq no* identifies the audit record that produced the anomaly.

9.1.6 Interface Data

The tables described here are used either for customizing IDES or for displaying information about the status of IDES.

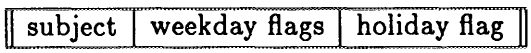
Command classes

command name	command class

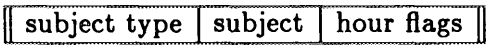
This table is used for classifying special commands. Currently, there are three classes of commands: mailers, editors, and compilers. The mailers are `mm`, `hermes`, and `mail`; the editors are `emacs` and `edit`; and the compilers are `pcc`, `bliss`, `fortran`, `macro`, and `link`.

Workdays and Workhours

Time segments are classified into different levels of activity expected depending on when the time segments occurred.

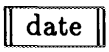


For users, time segments are classified by the day of the week in which time segments occur. By default, Monday to Friday are working days; Saturday, Sunday, and holidays are nonworking days. The security administrator can specify for each user (and user group) the working status of each weekday and holidays in general by setting the flags in the *workdays* table appropriately.



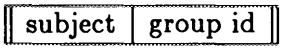
For systems and hosts, time segments are classified by the hour. By default, 7 am to 7 pm are working hours; the rest are off hours. The security administrator can specify for each host or system the working status of each hour by setting the flags in the *workhours* table appropriately. There are two tables of this kind, one for hosts and one for systems.

Holidays



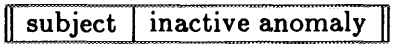
This table stores the dates that are considered holidays.

Groups



There are three tables of this kind, one for each subject type. This table stores the groups to which subjects belong.

Inactive anomalies



There are three tables of this kind, one for each subject type. By default, all measures are used in anomaly detection. This table can be used to turn off certain measures for a subject so that they do not contribute to the

anomaly-detection test. Note that although a measure specified here will be ignored for anomaly detection, data for the measure are still recorded in the active data and profile data so that should the measure later be reactivated, there would be a basis for comparison.

Message log

time segment	message type	seq no	priority	id	ack	time text

This table records the messages generated by different components of IDES. *Time segment* refers to the user time segment for which the message was generated. Messages are given *types* depending on their nature; for example, two message types are status messages and warning messages. *Seq no* identifies the audit record that caused this message to be generated. *Priority* indicates the importance of this message; for example, a message indicating occurrence of an anomaly has higher priority than one that simply reports of the completion of the daily profile update. Messages are assigned identifiers *(id)* so that they can be uniquely identified and referenced. *Ack* indicates whether the message should be acknowledged by the security administrator. How this field is used depends on the message monitor interface. Messages that do not require acknowledgment are simply displayed in a log which can be reviewed by scrolling backwards. Messages that require acknowledgment are displayed in a special area reserved for messages that have not been acknowledged by the security administrator; when the security administrator acknowledges that he or she has seen a message, the *ack* field is reset and the message is moved to the log of messages that do not require acknowledgment. *Time* shows when the message was generated. *Text* contains the text of the message.

Status log

sequence numbers	message number	uptime	oracle error	times

This table contains information about the status of IDES. *Sequence numbers* contains the last sequence number of the audit record seen by the receiver and the next sequence number expected. *Counts* records the number of reloads performed by the target system, the number of audit records processed by the active data collector, the number of audit records received by the receiver but not yet processed, and the number of control audit records generated by the receiver. *Oracle error* refers to the last error raised by the Oracle database system.

9.2 The IDES Architecture

IDES is divided into seven modules: receiver, active data collector, anomaly detector, expert system, profile updater, security administrator interface, and data archiver. All these components, except for the expert system shell, are written in C with embedded SQL[1] statements (using Oracle Pro*C). The SQL statements are used to query the Oracle relational database system that IDES uses to store the various types of data it processes. The expert system shell is written in Sun Common Lisp and communicates with the other processes by querying and asserting statements in the Oracle database.

We discuss the functionality and connectivity of each of the modules in this section.

9.2.1 Overview

Figure 9.2 is a schematic depicting the modules and the types of data from the IDES database that they share with one another. The *audit data* is introduced to IDES by the receiver—these are the audit records that came from the target system. The *active data* represents the statistics that have been collected for time segments since the profiles were last updated. These are generated by the active data collector from the audit data. The *anomaly data* are records generated by the anomaly detector when an anomaly is detected. They indicate the type of action that generated the anomaly, the measures IDES found to be anomalous, and other details including the time of the anomaly, and the time-segment id. The expert system examines each audit record from the *audit data,* uses its rules to determine whether the record is anomalous, and if so, writes a record into the *anomaly data.* The *profile data* are the subject profiles discussed in Section 5.3. This information is created and maintained by the profile updater. Finally, the archiver generates and stores *archive data,* which is simply a copy of the audit data brought in by the receiver.

Figure 9.2 shows the relationship between the IDES processes and the IDES database.

[1] *Structured Query Language* (SQL) is a *de facto* standard query language for relational database management systems. Oracle's version of SQL is called *SQL *Plus.*

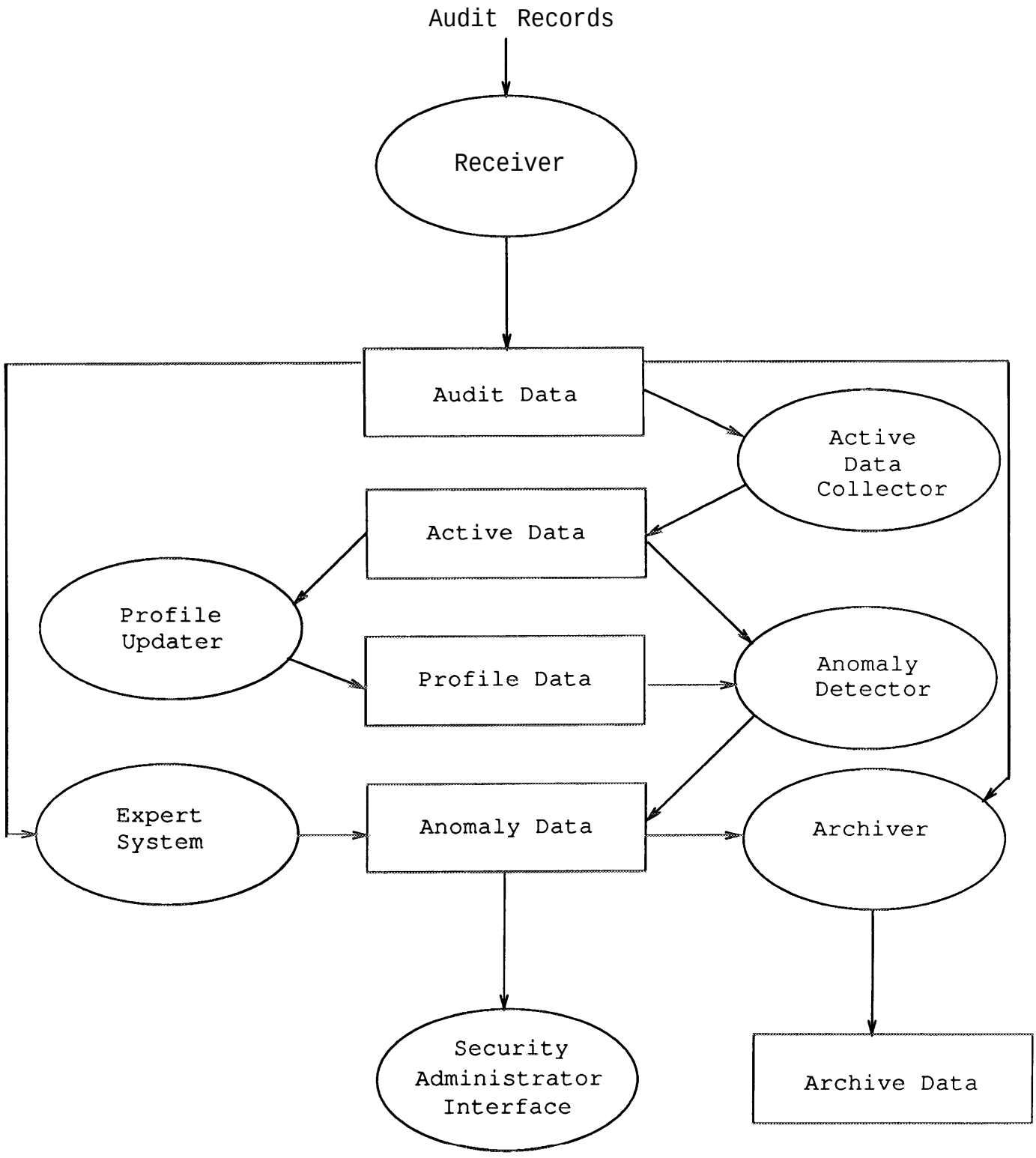


Figure 9.2: IDES Processes and Database Structure

9.2.2 The Receiver

The *receiver* implements the IDES protocol to communicate with the target system and generates control audit records in response to observed behavior on the target system. *Hello packets* are sent (as datagrams) to the target system at frequent intervals (every five seconds) until the target system responds with audit records during periods of inactivity. The *hello packet* identifies the version of the audit data format to be used and is directed to a pre-determined port of the target system. If a failure occurs during transmission of a message (usually the result of transient network errors) the receiver sleeps briefly and re-sends the message; the length of the sleep period is determined by the type of error.

Each encrypted audit record is received and processed in real-time. Each record is first decrypted and then parsed using a grammar that specifies the format of the audit data. Changes in audit data format can be handled readily by changing the grammar itself. If the record is valid, it is inserted into the table *audit record* of the *audit data.* The table *audit record* is locked in exclusive mode during the insertion process, which is typically less than 200 milliseconds in our prototype. Upon successful insertion into the *audit data,* a positive acknowledgment is sent to the target system. Upon the receipt of a duplicate record, an acknowledgment is sent to the target system without inserting the record into the *audit record* table.

An audit record can be rejected for a number of reasons:

- The audit record is improperly encrypted.
- The audit record has an invalid format.
- The audit record (as identified by its *sequence number) is* out of contiguous order.

The rejected record is negatively acknowledged to the target system and inserted into the *rejected audit record* table for later perusal by the security administrator.

In addition to processing incoming audit records, the receiver also detects hour or date changes and generates appropriate control audit records for these events. The receiver also detects and generates appropriate control audit records for user session starts and terminations by monitoring *login* and *logout* records, target system reloads, changes in ownership of different jobs on the target system, and other indicators. It also perform checks to determine to which user session an audit record belongs; these checks are

highly target machine dependent and will have to be adapted to each target computer.

9.23 The Active Data Collector

The active data collector generates active data from the audit data. This involves looking at each audit record, determining which subjects and time segments it affects, and computing and incorporating into the active data values for the measures using the data from the audit record fields. The anomaly detector is invoked on the active data of each affected time segment, extrapolated to full segments (if they are not complete). The audit record is then marked as processed.

Upon receipt of a *new day* audit record, the active data collector invokes the updater to update the profiles.

Note that IDES uses the target system's clock in determining the time at which the audit records are generated, when an anomaly occurred, and when a new day or hour begins. This distinction is important, for example, in the case of a target system crash that results in a backlog of audit data.

9.2.4 The Anomaly Detector

The anomaly detector performs a composite statistical test using values for all the measures relevant to the subject for the time segment. (Recall that different intrusion-detection measures apply to different types of subjects.) If the test indicates unusual activity, measures that contributed to the test results are noted-more specifically, the top five measures with a deviation greater than 2.0 (if any) are selected. The test is performed once using the subject's own profile as the basis of comparison, and repeated using group profiles for the groups of which the subject is a member.

An initial default profile is used when no profile is available for the test. This initial profile has time-segment counts of 0.1, means of 0, and identity matrices for the covariance matrix and its inverse. (The anomaly detector is interested only in the continuous values for measures. Continuous values for categorical measures are derived using metrics and formulas described in Section 5.4; the initial default for the metrics is 6.901.)

If the composite test detects unusual activity, the anomaly detector creates an anomaly record containing the subject, action, object, and time fields from the audit record that initiated the anomaly detection, plus some additional information, including a textual description of the anomaly, the values

of the top five individually anomalous measures that signalled the intrusion, and an indication of the degree to which these values were anomalous. This record may then be used by the security administrator interface.

9.2.5 The Profile Updater

The profile updater is run once a day by the active data collector. It ages the profiles, incorporates the days' active data into the profiles, and removes the active data from the database. The updater updates both subject and group profiles— the active data incorporated into a groups' profile are the active data for every subject belonging to the group.

The profile updater provides the capability to *roll back* or restore a previous days' profile at the discretion of the security administrator so that profiles will not be skewed by data representing actual intrusions or security violations. For example, restoration may be initiated by the security administrator after confirming that an anomaly was in fact an occurrence of a security violation. Restoration involves simply replacing the profile of every affected subject and working status with the previous days' profile. The profile updater saves profile data for one day in order to make this possible.

9.2.6 The Expert System

We have started to integrate an expert system shell into the IDES prototype using an expert system shell-originally built by SRI's Alan Whitehurst for Symbolics workstations, called Production-Based Expert System Tool (P-BEST). We have sorted out the logistics of the integration and are designing the structure of the knowledge base and creating the rules to be contained in the knowledge base. The P-BEST shell is capable of communicating with IDES by reading audit records, marking them as processed, and inserting anomaly records when appropriate. We plan to build up the knowledge base and design an interface for the security administrator to insert new rules.

9.2.7 The Security Administrator Interface

The security administrator interface is implemented as several processes, as described in Chapter 7. The status monitor, query monitor, and anomaly monitor are described in our earlier report [1].

The session monitor is implemented using the SunView software package and can be manipulated using the mouse and the keyboard. The Session Monitor window is made up of two subwindows: the top window is a "form"

which the security administrator can manipulate using the mouse and keyboard, the bottom window is purely a display window.

The Detail Window is made up of three information subwindows (i.e., no input allowed): the top subwindow contains a summary of the statistics collected for the session; the middle subwindow is a scrollable window containing a log of the audit records received for that session; the bottom window is a scrollable window containing a log of the messages received for that session.

Figure 7.1 contains a picture of the Session Monitor window, while Figure 7.3 contains a picture of the Detail Window.

9.2.8 The Archiver

The archiver is implemented as a background process that "wakes up" once every week. It then locks the table *archive_record* exclusively, and removes all records in the table to a file named "archive.xxx," where "xxx" is the date and time (exact to the nearest second) of the latest processed record in *archive_record*. (The archive-file-naming convention facilitates later retrieval of specific records.) The empty table *archive_record* is unlocked and available for insertion of processed audit records.

Chapter 10

Conclusions

Because the statistical profile and expert system approaches to intrusion-detection each address a different threat, a successful intrusion-detection system should incorporate both approaches. Because IDES combines a statistical user profile approach with a rule-based expert system that characterizes intrusions, it has the potential to become a strong intrusion-detection system. The IDES prototype is capable of detecting anomalous behavior as evidenced by preliminary experiments. The prototype is capable of detecting and reporting anomalies in real time.

We have made major improvements to IDES in the last year. IDES monitors a wide variety of event types from the target system and implements 25 intrusion-detection measures. IDES keeps profiles for three subject types: users, remote hosts, and target systems. IDES also profiles groups of subjects. IDES keeps separate profiles for each working status for each subject. IDES uses highly sophisticated and efficient statistical algorithms for performing its anomaly detection. The profiles are aged so that audit data have a half-life of 50 days; thus the most recent audit data contribute most to the profiles. We have also incorporated an expert system shell into IDES, so that IDES can now make use of rules characterizing known system vulnerabilities and intrusion scenarios. IDES also provides a mechanism for backing up to an earlier set of profiles if an anomaly detected by IDES is determined to be an actual intrusion by the IDES security administrator, so that the data representing the intrusive behavior will not be reflected in the profiles. IDES now uses a much faster hardware base and uses a separate workstation for the security administrator interface. IDES now has a greatly improved security administrator interface.

Chapter 11

Future Work

In our planned follow-on work, we intend to further enhance IDES in the following ways.

- Experimentation to determine which intrusion-detection measures are most useful for detecting anomalies.

  We will enact staged intrusions in an effort to determine which measures are the most effective in detecting intrusions; that is, which have the highest true positive rate while maintaining an acceptable false-positive rate. Based on this analysis, we may also implement additional intrusion-detection measures.

- Parameterizable statistical tests, so that

  - Different subjects can have different thresholds.

    Our plans are to allow the IDES security administrator to adjust the false-positive rates individually for each subject. For example, the security administrator might raise a user's false-positive rate (which simultaneously raises the true-positive rate) if he or she has reason to doubt a user's integrity or if the user's current assignment or security level requires closer scrutiny. The security administrator might lower a user's false-positive rate if he or she knows that a user's current assignment is changing or that the user has other legitimate reasons for changing his or her behavior.

  - Different subjects can have different profile update periods.

  - Different subjects can have different initial default profiles.

- Trend tests.

  We plan to develop and implement *second-order* intrusion-detection measures; that is, methods for conducting trend tests (to detect how fast a user's mean and variance change). We believe that we can use these trend tests to detect planned attacks in which a user gradually moves his or her behavior to a new mean, from which to safely mount an attack, or gradually increases the spread of the behavior considered normal.

- Add more rules to the Expert System shell.

  We intend to implement a complete set of rules to the expert system shell that characterize known intrusion scenarios and target system vulnerabilities.

- Monitoring a network of Sun workstations.

  We are planning a three-year follow-on effort to modify IDES so that it, can monitor in real time a distributed Sun facility, each of whose workstations may have multiple and dynamically changing windows running Unix. The modified IDES will also retain its capability to monitor in a centralized environment.

  The patterns of use for users of Sun workstations are different from patterns of use for users of a centralized computing resource using computer terminals. Sun users will typically have several windows open simultaneously, many of which may be running Unix independent of the other windows. Moreover, some of the windows may have an associated tty, whereas others (depending on the type of window) may not. The concept of user time segment is vague, because users may open or close Unix shell or other windows without logging in or out, and may leave these windows for extended periods of time (days or weeks) without ever logging out of the workstation. A user will typically be authorized "superuser" privileges on his or her own workstation, but typically not on other workstations. Users in a networked Sun facility will have more remote logins, remote procedure calls, and file transfers than users of a centralized computer. Patterns of abuse may differ significantly also, because users typically will have to remotely login to another workstation or use remote procedure calls to access sensitive resources, which are more likely to be associated with a machine other than the user's own. The challenge is to discover suit-

able measures of behavior for users of a networked Sun facility that capture any given user's distinctive behavior pattern and that are also useful in discriminating between normal and intrusive behavior. This question is made more difficult by the need for IDES to be able to integrate data from numerous sources for any given user, for example, if a user is remotely accessing other workstations or moves from workstation to workstation.

- Feasibility of IDES to monitor other mainframes.

  In a separate project, we have completed a one-year study on the feasibility of using IDES to audit user behavior for a large database management application that runs on IBM mainframe systems (implemented using MVS, TOP SECRET, ADABAS, and application code). The follow-on of that project (which will run for three years, concurrently with the IDES follow-on) will culminate in the implementation of appropriate modifications to the IBM system, the transmission of audit data to additional instances of IDES dedicated to those applications, and the use of IDES for detection of misuses. However, any extensions to IDES itself that are necessary to provide suitable generality to accommodate both the Sun network and the IBM mainframe applications will be a part of the IDES follow-on effort.

- Correlation of audit data with other available data.

  In addition to the raw audit data themselves, the following additional data are helpful in distinguishing suspicious activity from normal activity:

  - Information about changes in user status, new users, terminated users, users on vacations, changed job assignments, user locations, etc.
  - Information about files, directories, devices, and authorizations.

  Correlation of audit data with other available data may help in detecting intrusion attempts. Information about users, such as vacation and travel schedules, job assignments (e.g., clerical, application user, programmer, systems programmer, etc.), and unusual terminal locations can be helpful to the IDES security administrator in judging whether observed behavior is suspicious.

- Monitoring audit data at more than one system level.

Audit data can be collected at more than one system level. For example, the events audited could be low-level system calls or the command names and arguments typed at a terminal by a user. Each level has its advantages and disadvantages with respect to the types of intrusions it is possible to detect, the complexity and volume of the data, and the ability to appeal to an intuitive understanding of what is happening when an anomaly is detected.

Gathering audit data at the lowest possible levels makes it harder to circumvent auditing. Because user commands and programs can be aliased, it is difficult to ascertain what is really happening if auditing is performed at the command line level. In addition, as Anderson points out, intruders may operate at a level of control that bypasses the auditing and access controls [5]. To detect intruders operating at such a low level, auditing should be performed at the lowest level possible.

However, auditing at the command-line level makes it easier to define rules that characterize intrusive behavior, because our intuition of an intrusion scenario is also at this level. When an anomaly is detected, command-line auditing also allows the system to provide an explanation of what was considered suspicious or abnormal in what the user was doing. In addition, with this level of auditing, a security officer can scan a user's audit records to get a "feel" for what has happened.

It seems then that the most effective auditing approach is to audit at a very low level, so as to be able to detect clandestine users, as well as at the command line or application level, so as to be able to formulate expert system rules that characterize intrusions, and also to be able to determine what happened by scanning the audit records for a user's time segment.

Sun Unix will have auditing at the system call level. We also plan to implement our own auditing functions at the command line level. The challenge is for IDES to integrate these two levels of information to perform meaningful intrusion detection.

Appendix A

An Example of IDES Profile Updating and Anomaly Detection

Here we present an example of how IDES applies its statistical procedures. The example considers a single user (whom we have called Tom) whose behavior is characterized by four intrusion-detection measures--two continuous and two categorical. The example starts by considering how IDES initializes Tom's profile on his first day on the target system. Next we consider how IDES updates Tom's statistical profile on his second day. Finally, we consider how IDES tests for abnormal usage on Tom's third day. (The example should not be construed to suggest that in general IDES will wait two days before commencing statistical testing. The example is divided into three daily parts to simplify the presentation of the statistical concepts only).

Initializing the User Profile-Day One

Tom's behavior is characterized by four intrusion-detection measures. For each session Tom initiates, IDES monitors the natural logarithm of the CPU time expended, the number of lines printed, the names of terminals used, and the names of files accessed. The time segment IDES uses for these four behavior measures is a session.

On his first day, Tom executes four sessions with the following values for the four intrusion-detection measures.

Session	Ln CPU	Print Lines	Terminals	Files Used
1	2.4	45	A, B	F1
2	-1.5	205	C	F2
3	1.8	0	A	F3, F2
4	4.2	123	A, C	F1

At the end of the day IDES processes these data to update Tom's profile. First, IDES converts the active data for the categorical measures, namely, terminals and files used (stored as occurrences per category), to equivalent continuous values. To accomplish this, IDES substitutes metric values for terminals and files used. The metric value that IDES assigns to a categorical measure is the negative natural logarithm of the category's "probability" of occurrence, as recorded in the profile. Because IDES has never seen Tom use these terminals or files before (since Tom does not have a profile yet), it assigns them a small probability of occurrence, 0.001. The negative of the natural logarithm of 0.001 is 6.901. To obtain a single continuous value when multiple categories "occurred" in a session, IDES *sums* the metric values for all the categories that occurred. Thus, because Tom accesses only one file in his first session, IDES uses the single metric value of 6.901 as the continuous value, and because Tom uses two different terminals in his first session, IDES adds their metric values together to obtain a single continuous value. (One might argue that the metric values of the two terminals should be averaged. Adding the metric values together is equivalent to multiplying their probabilities of occurrence. We believe that this is better than averaging probabilities. For example, suppose that terminal H and K were historically used 98 % and 0.4 % of the time respectively, and that in a particular session both terminals were used. Such a session should be identified as anomalous. The average of the metric values of the two terminals corresponds to a relative probability of 0.063, which would not be small enough to raise an anomaly Wag. The sum of their metric values corresponds to a relative probability of 0.0039, which is probably small enough to raise an anomaly flag.)

IDES obtains the following metric values for Tom's four sessions.

Session	Ln CPU	Print Lines	Terminals	Files Used
1	2.4	45	13.802	6.901
2	-1.5	205	6.901	6.901
3	1.8	0	6.901	13.802
4	4.2	123	13.802	6.901

The effective count vector is (4.0,4.0,4.0,4.0) for ln CPU, print lines, terminals, and files used, respectively, because each of these was measured for four sessions. Now IDES develops a vector of mean values and a matrix of cross products of the metrics shown above. At the end of the first day, the vector of mean values is (1.725, 93.25, 10.358, 8.626) for ln CPU, print lines, terminals, and files used, respectively.

The complete cross-product matrix is as follows.

28.7	317.1	93.2	60.0
317.1	59179.0	3733.4	2574.1
93.2	3733.4	476.2	333.4
60.0	2574.1	333.4	333.4

The cross-product matrix is obtained using the formula in Section 5.2. For example, entry (2,1) of the matrix is computed as follows:

$$(2.4 * 45 + (-1.5) * 205 + 1.8 * 0 + 4.2 * 123) = 317.1.$$

where 2.4, -1.5, 1.8, 4.2 are the values of ln cpu and 45, 205, 0, 123 are the values of print lines for Tom's four sessions.

From the above matrix it is easy to calculate the covariance matrix and its inverse. Basically, the (2,1) entry is 317.1/4 - 0 * 0 = 79.28 using the formulas in Section 5.2, where 317.1 is the (2,1) entry of the cross-product matrix, 4 is the number of sessions, and the means for ln cpu and print lines are 0 and 0.

The effective count vector, the mean vector, the covariance matrix, and the inverse covariance matrix form Tom's profile. For categorical measures, namely terminals used and files accessed, tables of metrics need to be computed. Because Tom has no previous history, his new profile for terminals used and files used is based entirely on the data from his first four sessions. IDES builds these profiles by going back to the original active data (rather than the transformed metric values).

Tom's profile for terminals used is as follows.

Terminal I.D.	Count	Metric
A	3.00	0.69
B	1.00	1.79
C	2.00	1.10

For example, the metric for terminal A is computed as follows using formula given in Section 5.4.

$$-log_e(3.00/(3.00 + 1.00 + 2.00)) = 0.69.$$

Tom's profile for files used is as follows.

File I.D.	Count	Metric
F1	2.00	0.92
F2	2.00	0.92
F3	1.00	1.61

For example, the metric for file F3 is computed as follows using the formula given in Section 5.4:

$$-log_e(1.00/(2.00 + 2.00\ 1.00)) = 1.61.$$

Updating the User Profile-Day Two

On the second day, Tom completes three more sessions. The relevant data for these sessions are as follows.

Session	Ln CPU	Print Lines	Terminal	Files Used
1	-2.5	123	B	F3
2	2.9	2	D	none
3	0.3	60	A, C	F3, F4

Because there is still very little data for the profiles, testing for intrusions is delayed until day three. At the end of day two, IDES will, however, update the profiles.

For each session in day 2, IDES determines a metric value for the categorical variables. The metric value for terminals used in session 1 is the transformed probability value for terminal B from the historical profile for terminal used after day one (i.e., 1.79). Because terminal D has never been observed, the metric value for terminal used in session two is -ln(0.001) = 6.90. The metric value for terminals used in session three is the sum of the transformed values for terminals A and C, which equals 0.69 + 1.10 = 1.79 where 0.69 is the metric associated terminal A and 1.79 is the metric associated with terminal C for Tom. The metric values for files used can be similarly calculated. Because IDES has never seen a session with "no files" accessed (i.e., "no files" has a relative probability of zero), IDES assigns a metric value of 6.901 to files used in session two. After making these metric assignments, the metric values for the three sessions in day two are as follows.

Session	Ln CPU	Print Lines	Terminal	Files Used
1	-2.5	123	1.79	1.61
2	2.9	2	6.90	6.90
3	0.3	60	1.79	8.51

IDES uses these metric values to update the effective count vector, the vector of mean values, and the cross product matrix. To do so, IDES first applies the daily decay factor. This factor assures that the data in the profile have a certain half-life. (For this example, we use 30 days; in the current implementation of IDES, we use 50 days.) For 30 days, the decay factor is $e^{log_e(0.5)/30} = 0.9772$. Applying this factor to the previous days' effective count vector, IDES ages it to (3.91, 3.91, 3.91, 3.91). The new effective count vector after incorporating the three sessions from the current activity is (6.91, 6.91, 6.91, 6.91). For example, the new effective count for terminal used is $4.0 * 0.9772 + 3.00 = 6.91$ using the formula in Section 5.3.

IDES now updates the vector of mean values. The new mean value for ln CPU is given by the applying the formula given in Section 5.3. Given that the mean for ln CPU is 1.725, the new effective count is 3.91, and the three session values are -2.5, 2.9, and 0.3, and the number of sessions for the day is 3, the new mean for ln CPU is

$$((3.910 * 1.725) + (-2.5) + (2.9) + (0.3))/(3.910 + 3.000) = 1.077.$$

Proceeding in this fashion for the other measures, IDES obtains the updated mean value vector of (1.08, 79.54, 7.37, 7.34).

The cross product matrix can be updated using a similar formula to the one given above. The complete cross product matrix that IDES obtains is as follows.

14.75	-283.7	16.07	18.54
-283.7	18733	341.37	722.43
16.07	341.37	54.02	65.73
18.54	722.43	65.73	122.63

This is obtained using formula given in Section 5.2. For example, the (4,3) entry corresponds to the cross-product sum of terminal and files used which is given by

$$1.79 * 1.61 + 6.901 * 6.901 + 1.79 * 8.51 = 65.73.$$

Finally, IDES updates the profiles for the categorical variables. To do so, IDES ages the counts in the original profiles and then adds in the new counts. For the measure terminals used, this leads to the updated profile.

Terminal I.D.	Count	Metric
A	3.932	0.919
B	1.977	1.607
C	2.954	1.206
D	1.000	2.289

Tom's updated profile for files used is as follows.

File I.D.	Count	Metric
F1	1.954	1.515
F2	1.954	1.515
F3	2.977	1.093
F4	1.000	2.184
none	1.000	2.184

In the above profiles for categorical variables, the relative probability for all terminals and files in the profile exceeds 0.001. If terminal D were not used for many days, eventually the decay factor and the use of other terminals would reduce its relative probability below 0.001. To limit the amount of storage used, IDES adopts the convention that whenever a category of a categorical variable exhibits a relative probability below 0.001, that category is dropped from the historical profile (i.e., its relative probability is set to zero). New categories must be assigned a relative probability of at least 0.002. This convention will assure that new categories that enter the profile are not immediately removed. This convention is necessary, since otherwise whenever the count exceeds 1000, new categories would not be allowed to enter the profile.

Applying a Statistical Test - Day Three

Now suppose that we are satisfied that enough sessions have been gathered to provide a reliable profile for Tom. (In practice, this would actually take many more days if, as in this example, Tom's profile was started from scratch on day one. However, a default profile might be used for Tom as his starting profile, which would allow the statistical testing to commence when Tom logs on for the first time. The default profile would presumably be drawn from users with similar job assignments as Tom).

Beginning on the third day, IDES performs statistical tests of the normality of Tom's sessions. (The tests are described in Section 5.7.) On the

third day, Tom executes one session. The data for this session appears on six audit records. The first audit record reports that Tom logged on using terminal A. At this time no CPU cycles are reported as being expended. The second audit record reports that Tom accessed file F2, and that cumulative CPU expended is 0.001 seconds. The third audit record reports that Tom has accessed file F5 and that cumulative CPU expended is `0.009` seconds. The fourth audit record reports that Tom printed 255 lines and that cumulative CPU expended is 4.3 seconds. The fifth audit record reports that Tom switched to terminal C. The sixth audit record reports that Tom's session completed with a total usage of 6.9 seconds of CPU. Our cumulative record of this session is as follows.

Audit Record	Ln CPU	Print Lines	Terminals	Files Used
1	N/A	0	A	none
2	-6.91	0	A	F2
3	-4.71	0	A	F2, F5
4	1.46	255	A	F2, F5
5	1.46	255	A, C	F2, F5
6	1.93	255	A, C	F2, F5

IDES performs an updated composite statistical test each time it receives additional information concerning a session. Therefore, IDES performs six statistical tests for this one session, one for each audit record received. Before these tests can be conducted, IDES substitutes metric values for the categorical variables of terminals used and files used.

IDES first translates the categorical variables into metric values using the updated distributions at the end of day two. For example, terminal A translates into 0.919 and the combination of terminal A and C translates into 0.919 + 1.204 = 2.123. (These numbers are obtained from the metric profile for terminals from the last profile update.) Because file F5 has never been seen before, it receives a metric value of 6.901. So the combination of file F2 and F5 receives a metric value of 8.416, since the metric for file F2 is 1.515. Next IDES substitutes a mean value for a (categorical or non-categorical) continuous value whenever the continuous value is less than the mean value, *except* when the session is complete. For example, the first audit record reports below-average continuous values of ln CPU, print lines, terminals used and files used, so the mean values are substituted for all of these variables. Mean values are used for low values of measures in partially completed sessions because we do not want IDES to issue anomaly warnings simply because the session has only recently begun. The mean

and covariance matrices are appropriate for completed sessions; in order to use them for partially completed sessions IDES needs to ignore low values for measures. Of course, when the session is complete there is no further justification for substitution of mean values. This substitution rule yields the following results.

Audit Record	Ln CPU	Print Lines	Terminals	Files Used
1	1.08	79.53	7.374	6.377
2	1.08	79.53	7.374	6.377
3	1.08	79.53	7.374	8.416
4	1.46	255	7.374	8.416
5	1.46	255	7.374	8.416
6	1.93	255	2.125	8.416

For audit record 2, the continuous values of ln CPU, print lines, terminals used, and files used are -6.91, 0, 0.919, and 0.66. The corresponding mean vector from the profile is (1.07, 79.53, 7.37, 6.38). The actual values are less than the corresponding means; thus, the means are used instead of the actual continuous values. For audit record 6, the continuous values of ln CPU, print lines, terminals used, and files used are 1.93, 255, 2.125, and 8.416. The corresponding mean vector from the profile is (1.07, 79.53, 7.37, 6.38). Since the actual values exceed the respective means, the actual values are used.

The six statistical tests are conducted with the above data as each record is received. The composite test statistic is compared to a tabled value. If it exceeds that value, it is declared abnormal and individual tests are applied to find the top five individual measures whose z-values exceed 2.0. This information is conveyed to the IDES security administrator. The z-value for a measure is defined to be the metric value for that measure less the mean value, multiplied by the square root of the corresponding diagonal element in the inverse covariance matrix. Note that once a session is abnormal, it is not likely that it will later become normal, since all of the measures are monotonically increasing once the mean value has been exceeded.

At the end of the day, the data from the completed sessions are used to update the profiles.

Bibliography

[1] D. E. Denning, D. L. Edwards, R. Jagannathan, T. F. Lunt, and P. G. Neumann. *A Prototype IDES — a Real-Time Intrusion Detection Expert System*. Computer Science Laboratory, SRI International, Menlo Park, California, 1987.

[2] T. F. Lunt and R. Jagannathan. A Prototype Real-Time Intrusion-Detection System. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy,* April, 1988.

[3] D. E. Denning and P. G. Neumann. *Requirements and Model for IDES — A Real-Time Intrusion Detection System*. Computer Science Laboratory, SRI International, Menlo Park, California, 1985.

[4] D. E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering* 13-2, p. 222, February, 1987.

[5] J. P. Anderson. *Computer Security Threat Monitoring and Surveillance.* James P. Anderson Co., Fort Washington, Pennsylvania, April, 1980.

[6] R. R. Linde. Operating System Penetration. In *Proceedings of the National Computer Conference,* 1975.

[7] H. S. Javitz, A. Valdes, D. E. Denning, and P. G. Neumann. *Analytical Techniques Development for a Statistical Intrusion Detection System (SIDS) Based on Accounting Records.* SRI International, Menlo Park, California, July, 1986.

[8] T. F. Lunt, J. van Horne, and L. Halme. *Analysis of Computer System Audit Trails – Initial Data Analysis.* Sytek Technical Report TR-85009, Mountain View, California, September, 1985.

Computer Science Laboratory
SRI INTERNATIONAL

Technical Reports - 1988

SRI-CSL-88-1, January 1988
Higher Order Functions Considered Unnecessary for Higher Order Programming, by Joseph Goguen.

SRI-CSL-88-2R2, January 1988, Rev. 2, August 1988
What is Unification? A Categorical View of Substitution, Equation and Solution, by Joseph Goguen.

SRI-CSL-88-3, January 1988
Petri Nets Are Monoids, by José Meseguer and Ugo Montanari.

SRI-CSL-88-4R2, August 1988
OBJ as a Theorem Prover with Applications to Hardware Verification, by Joseph Goguen.

SRI-CSL-88-5, May 1988
A Descriptive and Prescriptive Model for Dataflow Semantics, by R. Jagannathan.

SRI-CSL-88-6, June 1988
Access Control Policies: Some Unanswered Questions, by Teresa F. Lunt.

SRI-CSL-88-7R, Sept. 1988
Quality Measures and Assurance for AI Software, by John Rushby.

SRI-CSL-88-8, August 1988
Secure Distributed Data Views. Vol. 1: Security Policy and Policy Interpretation for a Class Al Multilevel Secure Relational Database System, by Teresa F. Lunt, Peter G. Neumann, and Dorothy Denning (Computer Science Laboratory, SRI International) and Roger R. Schell, Mark Heckman and William R. Schockley (Gemini Computers, Inc., Monterey, Calif.).

SRI-CSL-88-9, August 1988
Introducing OBJ3, by Joseph A. Goguen and Timothy Winkler.

SRI-CSL-88-10, Sept. 1988
Cell and Ensemble Architecture for the Rewrite Rule Machine, by Sany Leinwand, Joseph A. Goguen, and Timothy Winkler.

SRI-CSL-88-11, Sept. 1988
Software for the Rewrite Rule Machine, by Joseph A. Goguen and José Meseguer.

Continued on next page.

SRI-CSL-88-12, Oct. 1988
IDES: The Enhanced Prototype, A Real-Time Intrusion-Detection Expert System, by Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Sherry Listgarten, David L. Edwards, Peter G. Neumann, Harold S. Javitz, and Al Valdes.

1987

SRI-CSL-87-1, May 1987
The Rewrite Rule Machine Project, by Joseph Goguen et al.

SRI-CSL-87-2, May 1987
Concurrent Term Rewriting as a Model of Computation, by Joseph Goguen, Claude Kirchner, and José. Meseguer.

SRI-CSL-87-3, May 1987
An Abstract Machine for Fast Parallel Matching of Linear Patterns, by Ugo Montanari and Joseph Goguen.

SRI-CSL-87-4, June 1987
Security and Inference in Multilevel Database and Knowledge-Base Systems, by Matthew Morgenstern.

SRI-CSL-87-5, June 1987
An Intrusion-Detection Model, by Dorothy E. Denning.

SRI-CSL-87-6, June 1987
A Multi-level Relational Data Model, by Dorothy E. Denning and Teresa F. Lunt (SRI International), and R.R. Schell, M. Heckman, and W. Shockley (Gemini Computer, Inc.).

SRI-CSL-87-7, July 1987
Unifying Functional, Object-Oriented and Relational Programming with Logical Semantics, by Joseph Goguen and José Meseguer.

Computer Science Laboratory
Attn: Technical Reports/BN186
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025

Tel. (415) 859-5924, ARPAnet:burgess@csl.sri.com