A Prototype Real-Time Intrusion-Detection Expert System

Teresa F. Lunt and R. Jagannathan
Computer Science Laboratory
SRI International
Menlo Park, California 94025

Abstract

This paper describes the design and implementation of a prototype intrusion-detection expert system (IDES) developed at SRI International. IDES is based on the concept that an intrusion manifests itself as a departure from expected behavior for a user. The prototype monitors users on a remote system using audit records which characterize their activities. It adaptively learns normal behavior of each user and detects and reports anomalous user behavior in real-time.

1 Introduction

Although a computer system's primary defense is its access controls, it is clear from numerous newspaper accounts of break-ins and computerized thefts that access control mechanisms cannot be relied upon in most cases to safeguard against a penetration or insider attack. Even the most secure systems are vulnerable to abuse by insiders who misuse their privileges, and audit trails may be the only means of detecting authorized but abusive user activity.

The goal of IDES (intrusion-detection expert system) is to provide a system-independent mechanism for real-time detection of security violations, whether they are initiated by outsiders who attempt to break into a system or by insiders who attempt to misuse the privileges of their accounts on the system. The IDES prototype, developed by SRI under contract to the Navy's Space and Naval Warfare Systems Command (SPAWAR), is based on the IDES model developed by Dorothy Denning [1,2]. 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.

Anderson [4] 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 the approach taken so far by IDES, which was initially based on the hypothesis that any exploitation of a computer system's vulnerabilities entails behavior that deviates from previous patterns of use of the system; consequently, intrusions can be detected by observing abnormal patterns of use. 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. Such an approach has been taken by others (e.g., [6]). At a later date we plan to include rules in IDES that would characterize certain types of intrusions. Such an enhancement will potentially enable IDES to detect external and internal penetrators.

Anderson had little to offer towards detecting the legitimate user who abuses his or her privileges. However, *a priori* rules for "socially unacceptable" behavior, analogous to those that characterize intrusions, can also be included in IDES. We plan to include such rules. Comparison with the norm established for the class of user to which the user belongs could also be used to detect abuse of privilege; we are considering this approach for the case in which there is a very large number of users (in

the thousands or tens of thousands) who may operate in distinct roles or job capacities.

Anderson offers little hope for detecting clandestine users. The clandestine user can evade auditing by use 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. 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. At least one study has included this approach [6]. 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. As Linde has suggested [5], auditing and intrusion-detection mechanisms can make it so difficult for a penetrator to avoid detection that other methods, such as bribing a user, will be more attractive.

1.1 The IDES Prototype

The IDES prototype is an independent system that runs on its own hardware and processes audit data characterizing user activity received from a target system over a network [3]. The user activity that we monitor in the prototype consists of the following types of user actions: login, logout, program execution, directory modification, file access, system call, session location change, and network activity.

The IDES prototype determines whether user behavior as reported in the audit data is normal with respect to past or acceptable behavior characterized by specific measures[1]. It maintains user profiles[2] for measures. The measures are applied to individual user sessions (i.e., from user login to user logout). There are two kinds of measures: *discrete* and *continuous*. A *discrete* measure is one whose domain of values is a finite, unordered set (e.g., the set of locations). Such measures are generally constant for a particular user session. Discrete measures that we use in the prototype include location and time of login. A *continuous* measure is one whose value is a number or count that accumulates over a user session (e.g., connect time). Continuous measures that we use in the prototype include connect-time, cpu-time, io-activity, and number of protection-violations.

The profiles are periodically updated based on the behavior of the users of the target system. For each measure, the user's profile for that measure defines the user's normal behavior. IDES is driven by the arrival of audit records, each of which describes behavior relevant to possibly several intrusion-detection measures. Anomalous behavior is user behavior that deviates from the normal behavior for some measure by an amount indicated in the user profile for that measure. Because IDES can be configured to monitor arbitrarily detailed user behavior, it is potentially capable of detecting intrusions (for example, masqueraders) that cannot be detected by the target system's access controls.

The IDES prototype has demonstrated its ability to adaptively learn users' behavior patterns; as users alter their behavior, the thresholds maintained in 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, IDES derives its own rules. Thus, IDES is potentially sensitive to abnormalities that human experts may not have considered.

The IDES prototype has been implemented on two Sun 3 workstations[3]. and uses a relational database system called Oracle[4]. Oracle provides SQL, a *de facto* standard relational database query language, and a preprocessor, Oracle Pro*C, to translate SQL queries embedded in application programs written in the C programming language. The programming language we used is C.

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 its interface specifications.

The IDES prototype currently monitors a DEC-2065 machine at SRI running a locally customized version of the TOPS-20 operating system[5]. The operating system is configured to support as many as 128 simultaneous user sessions. We have modified the TOPS-20 operating system to collect audit data, transform the data into the IDES format, encrypt the formatted data, and trans-

[1] A *measure* is an aspect of user behavior on the target system.

[2] A *profile* is a description of the normal behavior of a user with respect to a particular measure.

[3] Sun workstation, SunWindows, and the combination of Sun with a numeric suffix are trademarks of Sun Microsystems, Inc.

[4] Oracle and Oracle Pro*C are trademarks of the Oracle Corporation.

[5] DEC-2065 and TOPS-20 are trademarks of Digital Equipment Corporation.

mit the records to IDES according to the IDES protocol. Although IDES couid preprocess the target system audit data into the IDES audit record format, we believe 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. In the case of the DEC-2065 machine, there has been no noticeable degradation in performance from the collection and preprocessing of the audit data. The audit data collected on the DEC-2065 machine are immediately queued for transmission to IDES and sent as soon as possible, reducing the chances of their being tampered with on the target system.

We developed a flexible format for the audit records that IDES expects to receive from the target system, and a 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). Currently IDES is receiving only a fraction of the data that could be collected and transmitted by the target system. The IDES protocol allows for the target system to send a great deal of additional information that are not currently being sent. In our ongoing work for SPAWAR, we are collecting additional data from the target system in order to implement a greater variety of intrusion-detection measures, thereby improving the intrusion-detection capability of IDES.

The IDES prototype monitors several event types (namely, login, logout, program execution, directory modification, file access, system call, session location change, and network activity) on the target system. It is able to monitor all users of the target system for these event types in real time.

The IDES prototype has achieved the following results in the context of the event types monitored and the measures that are profiled.

- Demonstration of proof-of-concept.
- Ability to monitor all users on a DEC-2065 system in real time.
- Ability to detect some simulated intrusions on the target system.
- Ability to adaptively learn users' behavior patterns.

2 The Intrusion-Detection Measures

In the IDES prototype, we have implemented several discrete and continuous intrusion-detection measures.

- **Discrete Measures:** A discrete measure is a function of a user session whose range of values is a finite, unordered set that is used to characterize some aspect of user behavior. Each user profile for a discrete measure consists of a list of values and the associated discrete probability density function based on previous behavior for that user for that measure. User behavior that does not match one of the profile values for that user is considered anomalous. A decay factor is applied to the profile data to give them a half-life of 50 days. This way the profile reflects a moving time window of behavior for each user. Any profile value whose probability of occurring is less than one percent is dropped from the profile. Thus a user behavior is considered anomalous if it has less than one percent likelihood of occurring for that user with respect to a discrete measure. The discrete measures that we have implemented so far are:
  - **Time of login** — *Time of login* is the number of sessions for each user for a time period of the day. Each 24-hour period is divided into three categories: day period, night period, and graveyard period. Each user login is classified into one of these three categories.
  - **Location of login** — *Location of login* is the number of sessions for each user initiated from a particular physically identifiable location. The location could be a terminal directly connected to the system or connected through a multiplexor, or could be a remote host over one of several networks (such as Arpanet or Decnet.) Each unique location from which a user logs in defines a separate category. Each user login is from one location.
- **Continuous Measures:** A continuous measure is a function of some aspect of a user session such that the function value changes (accumulates) during the course of the session. Each user profile for the continuous measures contains the first and second moments of the joint probability distribution of the values of the measures over prior sessions for that user. Assuming normal distributions, user behavior that has less than a one percent likelihood of occurring based on the joint distribution is considered anomalous. As for discrete measures, the distribution is decayed by a disccunt factor to give the profile data a half-life of 50 days. The continuous measures that we have implemented so far are as follows.

- **Connect Time** — *Connect time* is the length of a user session on the target system. Each session is referred to as a *job* and has a unique job number. Several sessions for the same user can be active at the same time. Each session is delimited by the association and disassociation of a user with a job number.
- **CPU Time** — *CPU time* is the amount of processing time consumed in a user session on the target system.
- **IO Activity** — *IO activity* is the amount of input and output activity in a given user session on the target system.
- **Protection Violations** — *Protection violations* is the number of directory and file access protection violations in a user session on the target system.

The continuous measures are reported with each audit record and are accumulated for each user session. A session is reported as anomalous at the point at which the accumulated value is less than one percent likely based on recent behavior.

It is relatively straightforward to add additional intrusion-detection measures to IDES now that the framework has been established. Different intrusion-detection measures may be appropriate to different classes of user. 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 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. Other pertinent measures are number of directory modifications, number of password errors per day, and user never seen before.

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 in several different locations and even 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 this 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 and who slowly varies 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 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 such "second order" measures to detect such trends in user behavior.

In earlier research at SRI by Javitz et. al., an extensive statistical analysis on audit data from IBM systems running MVS and VM was performed [7]. In that work, a high-speed algorithm was developed that could accurately discriminate between users based on their behavior profiles. These statistical procedures are potentially capable of flagging no more than one percent of user behavior as abnormal while detecting approximately 95 percent of all intrusions.

The purpose of the study was to develop analytical statistical techniques for screening computer system accounting data to detect behavior that may be indicative of intrusions. The techniques detect deviations from statistical norms developed for each user. They used a number of job variables, such as CPU time, terminal used, and the number of files accessed, to construct a profile of jobs for a user and then to determine whether an individual job was normal, using discriminant analysis.

In a discriminant analysis, the multivariate probability distribution (with respect to parameters such as CPU time, time of day, etc.) of normal jobs is estimated for a user. Then every point in the multivariate space is assigned a value equal to the inverse of the "height" of the user's job probability distribution. The points with the largest inverse values form a *critical region* in which the probability of normal jobs belonging to that region is less than a few percent. Once a user's critical region has been determined, a new job for that user can be considered abnormal if it falls into the critical region, and normal otherwise.

To be useful, the statistical algorithms must:

- Maximize *true positive rate* (percentage of intrusive use correctly identified as abnormal).
- Minimize the *false positives rate* (percentage of normal use 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 Javitz study found that discriminant analysis was potentially capable of obtaining a true positive rate of 95 percent while maintaining a false positive rate of only a few percent. In our continuing work, we are extending the statistical techniques developed by Javitz et. al.

More comprehensive attempts to characterize intrusions are being made by several study teams (e.g., MIDAS [6]). These systems encode information about known system vulnerabilities and reported attack scenarios, as well as intuition about suspicious behavior, in rule-based systems. 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. Audit data from the monitored system is matched against these rules to determine whether the behavior is suspicious. A limitation of this approach 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. However, we believe that such an approach, if used in conjunction with our current approach of detecting departures from established user norms, addresses the vulnerabilities and has the strengths of both approaches. Thus, we plan to include such capabilities in IDES at a later date.

3 Preliminary Results

We conducted preliminary experiments to validate the IDES prototype. We used an account on the target system that already had a certain profile established as normal with respect to three intrusion-detection measures: connect time, shift of login, and location of login. We generated a connect time anomaly by logging into the system with a session whose length was outside the normal range for that user, and then logging out. IDES detected the anomaly when it processed the logout record. We generated a location anomaly by logging the user in from a location that did not match the user's profile with respect to that location. IDES detected the anomaly when it processed the login record. We generated a shift anomaly by logging in the user enough times in a shift to cause an anomaly. The prototype detected the anomaly when it processed the offending login record.

The prototype is able to report an anomaly usually within a minute from when the user caused the anomaly at the target system. These statistics were obtained by analyzing several thousand audit records. The transmission of the audit record from the target system to IDES is over two or more Ethernets and across at least one gateway. The processing time includes time taken to determine whether the record signifies an anomaly (if so, generating an anomaly record) and to archive the record.

The prototype is implemented on a hardware base consisting of a Sun-3/260 and a Sun-3/60. The IDES processes that communicate with the target systems; that receive, decrypt, and validate audit records; that match audit records against profiles and compare against thresholds to detect anomalies; and that update the profiles to adaptively learn user behavior are implemented on a Sun 3/260 with a 560 MByte 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.

4 The IDES Design

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. 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 user profile for each applicable intrusion-detection measure. After an audit record has been processed, it is placed into an archive file for eventual backup. The profiles themselves change dynamically on the basis of observed activity; each measure has an associated profile update period indicating the frequency of such changes. When an anomaly is detected, the an anomaly record is generated and recorded in the database. The IDES administrator interface is structured so that the IDES administrator can selectively examine different types of anomalies graphically or by using a query interface.

The IDES prototype consists of processes which interact with each other through the Oracle relational database system, which manages the IDES database. Figure 1 shows the structure of IDES.

4.1 The Audit Data

IDES receives audit records from the target system using audit datagrams, each of which completely describes a single audit trail event (including such information as cause, location, and time). Each datagram consists of a

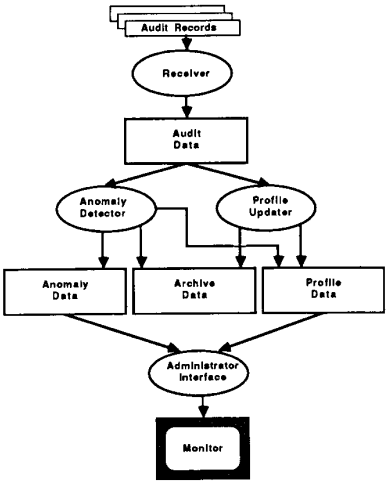

Figure 1: Structure of the IDES Prototype

plain-text header, to identify the datagram to IDES, and an encrypted data message containing the audit data. No information about an audit event is ever communicated in plain-text.

The header contains four decimal fields for use in processing the datagram. The first field is a number, assigned by IDES, that uniquely identifies the sending target system (providing the capability to receive audit data from several target systems simultaneously). The second field identifies the format of the encrypted datagram. The third field identifies the format of the decrypted datagram. The fourth field is the audit record sequence number, which is used to indicate the order of events and to encrypt the datagram.

The datagram event data consist of a variable number of fields which provide information about the event. This information is categorized into three major fields: subject, action, and object. Many fields contain additional subfields that further categorize the information being provided. The *subject* field serves to uniquely identify the user and location of the event. The *action* field identifies what action the subject was attempting. Each datagram may report only a single action. The *object* field describes what the subject was performing the action upon. This field may contain many different items depending upon the action being performed. The *error-code* field is used to indicate that an error occurred when the subject attempted the action on the specified object. The action is assumed to have been performed successfully if this field is omitted. The *resource-info* field is used to report resources used by the subject since the current session began. The currently defined resources are cumulative (since login) connect time, cpu time, io activity, and protection violations. The *time* field reports the date and time at which the event occurred.

4.2 The IDES Database

The IDES database consists of the following data:

- **Audit Data** — The *audit data* consist of a table of valid decrypted and unprocessed audit records received from the target system.
- **Active Data** — The *active data* record the accumulated activity for each user for each intrusion-detection measure (since the last profile update for that measure), so that anomalies can be readily detected. They are accumulated between profile update periods and are used to periodically update the profiles for the various intrusion-detection measures. At the end of each period, when the profile for a measure is updated, the accumulated total for each user in the *active data* is reset.
- **Archive Data** — 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.
- **Profile Data** — The *profile data* consist of several tables, one for each intrusion-detection measure. Each table contains a record for each user, defining normal behavior based on past behavior for that user. Each record also contains a threshold for that user, used to differentiate anomalous behavior with respect to that measure. Each profile table has an associated profile update period, at the end of which the statistics defining normal behavior and the thresholds are recalculated using the most recently observed behavior (from the *active data*.)
- **Schedule Data** — The *schedule data* consist of a single table containing a record for each intrusion-detection measure. Each record specifies the period (in seconds) for updating the associated profile table and the period (in seconds) for resetting the current data in the associated *active data* table. Each record also indicates when the profile was last updated and contains a default threshold value to use initially (i.e., before any activity has occurred for a user with respect to the measure.)
- **Anomaly Data** — The *anomaly data* consist of a set of tables, one table for each type of anomaly, containing anomaly records. An anomaly record is generated when the user's current behavior deviates abnormally from the normal behavior as specified in the user's record for a particular profile.

4.3 The IDES Processes

IDES consists of several processes:

- **Receiver** — The *receiver* implements the IDES protocol with the target machine and decrypts, parses, and validates each audit record as it is received. Valid audit records are acknowledged to the target system, and invalid audit records are negatively acknowledged to the target system. Only valid audit records are inserted into the *audit data.*

- **Anomaly Detector** — The *anomaly detector* retrieves audit records from the *audit data.* Each audit record describes a user action; the *anomaly detector* updates each record for this user in the *active data* tables for the relevant intrusion-detection measures. For each relevant measure, the current audit record is compared with the user's profile; if the user behavior is considered anomalous, then a record is generated and inserted into the *anomaly data.* After an audit record is processed, the *anomaly detector* removes it from the *audit data* and inserts it into the *archive data* for eventual archival.

- **Archiver** — The *archiver* periodically backs up processed audit records from the *archive data* to an external file.

- **Profile Updater** — The *profile updater* updates each profile table from the corresponding tables of the *active data* at the end of the profile period for each measure, and then resets the associated *active data* to zero. The *profile updater* also recomputes the probability distributions (after appropriate decaying) for each affected profile record.

- **Active Data Resetter** — The *active data resetter* periodically updates the *active data.* Each table in the *active data* is associated with a particular measure and has a predetermined reset-period. For example, the reset period for shift of use is 24 hours, for location of use is 24 hours, and for connect time is precisely the length of each session. At the conclusion of each reset-period, the activity recorded during that period is accumulated into a total activity field, the number of periods is incremented (these fields are used by the profile updater to compute the new probability distributions), and the current activity field is reset to zero.

4.4 IDES Administrator Interface

One of the strengths of the current IDES prototype is its window-based administrator interface, which is implemented using the SunView window system, and which provides the IDES administrator with a powerful and comprehensive view of the target system being monitored.

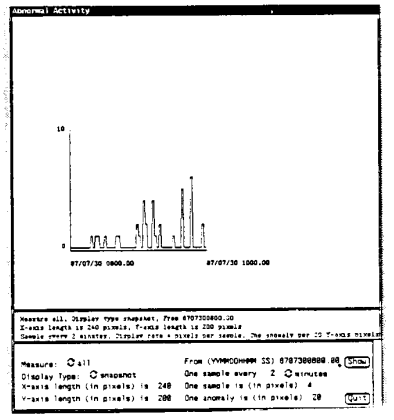

Figure 2: An Anomaly Monitor Window

The IDES 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 administrator can quickly and easily determine which user at what location is generating the anomaly.

The administrator interface consists of the following displays:

- A *status monitor* that displays the current status of IDES, including the rate at which audit records are being received, rejected, and processed.

- An *anomaly monitor* that displays the anomalous behavior detected in the target system for a particular measure or for all measures, in the past or at the present.

- A *query monitor* that allows the IDES administrator to probe the IDES database by using built-in or ad-hoc queries.

The IDES administrator can create several instances of each display, where each instance is presented in its own window. Figure 2 shows an example of a window that the security administrator could display on the Sun workstation screen.

5 Conclusions

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 are currently investigating measures

that would be better discriminators of anomalous activity from normal activity. We are also including in the administrator interface capabilities to fine-tune user profiles, activate and deactivate measures, and report anomalies in different ways.

Acknowledgments

This work was supported by the U.S. Navy, SPAWAR, under contract N66001-84-D-0077, Delivery Order 19. We wish to acknowledge the other members of the project team — Dorothy Denning, Dave Edwards, Hal Javitz, Peter Neumann, and Al Valdes — for their important contributions to the work we have described.

References

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

[2] D. E. Denning. An Intrusion-Detection Model. *IEEE Trans. on Software Eng.* 13-2, p. 222, Feb. 1987.

[3] D. E. Denning, D. E. 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, 1987.

[4] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. James P. Anderson Co., Fort Washington, PA, April 1980.

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

[6] R. A. Whitehurst. Expert Systems in Intrusion-Detection: A Case Study. Computer Science Laboratory, SRI International, Menlo Park, CA, Nov. 1987.

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