[12] 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. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 1987.

[13] P. A. Karger. Limiting the damage potential of discretionary Trojan horses. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, April 1987.

[14] D. E. Denning and P. G. Neumann. Requirements and model for IDES—a real-time intrusion detection system. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 1985.

[15] D. E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2), February 1987.

[16] *Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD.* Department of Defense, December 1985.

[17] TRW Defense Systems Group. Intrusion-detection expert system feasiblity study. Final Report 46761, TRW, 1986.

[18] P. G. Neumann. Audit trail analysis and usage data collection and processing, part one. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, January 1985.

[19] P. G. Neumann and F. Ostapik. Audit trail analysis and usage data collection and processing, part two. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, May 1987.

[20] W. T. Tener. Discovery: An expert system in the commercial data security environment. In *Proceedings of the IFIP Security Conference*, Monte Carlo, 1986.

[21] D. W. Haskin. Keeping watch on a VAX. *Digital Review*, December 16 1987.

[22] A. R. Clyde. Insider threat identification systems. In *Proceedings of the 10th National Computer Security Conference*, September 1987.

[23] R. R. Linde. Operating system penetration. In *Proceedings of the National Computer Conference*, 1975.

Acknowledgments

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.

References

[1] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

[2] R. Alan Whitehurst. Expert systems in intrusion-detection: A case study. Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, November 1987.

[3] 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. Technical report, SRI International, Menlo Park, California, July 1986. not available for distribution.

[4] J. van Horne and L. Halme. Analysis of computer system audit trails — final report. Technical Report TR-85007, Sytek, Mountain View, California, May 1986.

[5] T. F. Lunt, J. van Horne, and L. Halme. Automated analysis of computer system audit trails. In *Proceedings of the Ninth DOE Computer Security Group Conference*, May 1986.

[6] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — initial data analysis. Technical Report TR-85009, Sytek, Mountain View, California, September 1985.

[7] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — intrusion characterization. Technical Report TR-85012, Sytek, Mountain View, California, October 1985.

[8] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — feature identification and selection. Technical Report TR-85018, Sytek, Mountain View, California, December 1985.

[9] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — design and program classifier. Technical Report TR-86005, Sytek, Mountain View, California, March 1986.

[10] J. van Horne and L. Halme. Analysis of computer system audit trails — training and experimentation with classifier. Technical Report TR-85006, Sytek, Mountain View, California, March 1986.

[11] 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.

Auditing can be controlled by VMS-format keyboard commands or from programs.

Audit allows analysis of the audit data by random sampling or through selective analysis based on the system manager's knowledge of external events. Audit's analysis produces three reports: a security summary report, which summarizes the activity of high-risk users (as defined by a predetermined set of 14 risk factor tests and a programmable set of weighting parameters); a security event report, which summarizes the events that caused those users to be considered high-risk; and a supporting data report, which includes data from the audit log to support the conclusions of the first two reports. The risk factors for which Audit tests include sessions outside business hours or on weekends or holidays (the definition of normal business hours and holidays can be selected by the system manager); sessions indicating use of the AUTHORIZE or SYSGEN utilities; sessions indicating browsing; file access alarms; other alarms (alarms can be established for certain activities); repeated unsuccessful login attempts; sessions with dial-up or remote terminals; simultaneous logins for the same user; and attempts to turn off auditing. Some of Audit's 14 tests use data contained in the audit logs, and some use information from the VMS operator log file; no test uses data from both. The operator log file is used to test for file access alarms, other alarms, login failures, and attempts to turn off auditing. The other tests use the audit logs.

Each of the 14 tests has an associated weight and three factors. One factor is for after-hour use; one factor is for activity from a dial-up terminal; and one factor is for activity from a DECnet remote terminal. Whenever an event satisfies one of the tests, its weight is multiplied by its relevant factors and the result is added to the score for that user. Users with sufficiently high scores are considered to be high-risk. The weights and factors can be selected by the system manager. The system manager can also add additional tests.

There has been at least one published report of bypasses of certain Audit tests [21]. Allen Clyde reports that Audit has detected "numerous acts of misconduct, including criminal conduct ... on sensitive computer systems" [22].

4 Conclusions

None of the intrusion-detection approaches described is sufficient alone—each addresses a different threat. A successful intrusion-detection system should incorporate several different approaches. In particular, a statistical user profile approach augmented with a rule-base that characterizes intrusions promises to be an effective combination. Because they use this combination of approaches, two prototype systems—IDES and MIDAS—have the potential to become strong intrusion-detection systems. Of these two, IDES is particularly strong in its statistical approach, whereas MIDAS focuses primarily on enumerating a comprehensive (although site-specific) set of expert system rules.

Although, as Linde notes [23], the more skilled penetrator can disable the auditing 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.

Discovery is an expert system that searches for frequently occurring patterns in subscriber inquiries to the data and compares these patterns to daily subscriber inquiry activity to detect variances in normal subscriber behavior. It develops a user profile for each customer by type of service and access method, and updates a user's profile daily if there has been activity for the user access code that day.

Discovery is system-specific in that the intrusion-detection rules are particular to the specific application-dependent data fields, or variables, being monitored. However, Discovery also monitors some variables that are generic to most computer systems, such as date and time of access, type of access, user location, user identifier, password, and port identifier. Discovery allows the security officer to choose the variables to be monitored and the threshold parameters, so that the system can be fine-tuned and the impact of adding new services can be determined. The security officer can also modify, delete, and add variables to be monitored as service offerings change. The thresholds can be set individually for each variable being monitored for each user access code.

Discovery analyzes the daily inquiry activity for each user access code for comparison with the established profile for the customer and also for comparison with a model of illegitimate access. Discovery only analyzes *correct* inquiries submitted by customers; thus Discovery cannot use error patterns as indicators of intrusions. Discovery records all inquiries that fall outside of acceptable thresholds, and provides an explanation for why the inquiry is unacceptable (these are not used in updating the customer's profile). Discovery is not a real-time system, but alerts the security officer to unusual activity at the end of the workday.

While Discovery was under development, a prototype was used to parallel the work of security investigators, in order to ensure that Discovery would make the same decisions as the investigators. TRW found that the use of Discovery resulted in investigative leads being developed more quickly, and the analysis of Discovery's exception data provided more concise leads than did the investigators' conventional methods. Other, unexpected, benefits included the ability to perform marketing analysis on detailed, up-to-the-minute data using Discovery's customer usage patterns. Trends can also be observed by comparing current customer usage data with previous usage data.

3.8 Clyde Digital Systems' Audit

Clyde Digital Systems' Audit is a product that audits users of VAX/VMS machines. Audit can create a complete log of all terminal input and output and provides procedures to help analyze the data collected.

Audit can record every byte that passes between a user's terminal and the system, including control and escape sequences, and stores this data in a file, although certain qualifiers can be specified so that particular special characters can later be discarded from the audit log file (for ease of display and formatting, for example). Audit also provides the option to monitor only terminal input (from the user).

Audit also provides a flexible capability for selective auditing. For example, auditing can be activated selectively for terminal sessions satisfying certain criteria, such as for specific users, or specific times of day (producing an audit trail for the user's terminal session), and the use of specific programs can also be selectively audited (producing an audit trail for the specified program).

is developed by having the user type his or her id and password repeatedly (about 12 times). Once a user's electronic signature is installed, BioPassword operates transparently to the user.

BioPassword is automatically activated when the workstation is turned on or reset or when a user logs off, and can be invoked by software for reverification at any time. BioPassword prompts the user for an id and password and verifies both the contents and keystroke dynamics of each against the stored electronic signature, letting the user proceed with normal work only if the verification is positive. Once access is granted, if the workstation is idle for a prespecified period of time, BioPassword times out and requires reverification of the user before continuation of the idled job. All access attempts and logins are audited.

BioContinuous incorporates all of the features of BioPassword and adds continuous real-time verification of users. BioContinuous is a single-board component for the IBM PC. With its own high-speed processor, the BioContinuous board continuously verifies a user's identity in parallel with the user's work on the PC's processor, using over 110 typing characteristics, including intervals, rhythm, an analog of pressure, and error characteristics. After developing an electronic signature for a user id and password, BioContinuous develops an extended electronic signature for each user. This extended electronic signature contains additional biometric signature data that match a user's keystroke characteristics used in normal work. The learning process takes place over a few days, and, once the learning process is completed, the user's keystroke dynamics are automatically and continuously verified against his or her extended electronic signature.

BioContinuous includes a *programmable security matrix* containing information that indicates what actions are to be taken when a possible intruder is detected. The action can depend on risk factors, such as which data are being accessed or which function is being performed. Thresholds and alarms can be preselected.

International Bioaccess Systems Corporation is developing a product called BioNet that will add flexibility to PCs connected to a local area network by providing a central storage of electronic signatures. This will allow authorized users to use any workstation on the network without having to store their electronic signatures on each workstation. BioNet also will provide for integrated auditing of an individual user's activity across all the workstations on the local network.

3.7 Discovery

Discovery is an intrusion-detection expert system developed by TRW to address the intrusion threat in an environment in which computer services are sold to outside customers [20]. In such an environment, the customers may not be as concerned with safeguarding the security and integrity of the service provider's system and information assets as would, say, those users (employees of the service company) who create and maintain that information asset. Thus, the customer cannot be relied upon in any program of security instituted at the service company. The threat is that subscribing customers may give away their passwords, or lend their terminal devices, to would-be intruders. Thus, Discovery attempts to detect imposters who have obtained legitimate user ids, access codes, and inquiry formats. The perceived need was for an intrusion-detection mechanism that operates transparently to the subscribers of the service. The goal is that Discovery become a preventive, as well as a detective, control.

NAURS monitors network activity originating from the TACs and network access controllers (NACs). It collects data about TAC/NAC logins, TAC/NAC login failures, logouts, open and close connections, and TACs coming online, and maintains the data in a database. NAURS provides both background analysis on past activity and real-time analysis of current use. It provides periodic audit trail reports and real-time reports on unusual events, triggered by the events themselves. Interactive query from local terminals is also supported. Incident reports, generated every day from the previous day's audit data, include incidents that satisfy one of three rules about the number of simultaneous logins and duration of sessions.

The threats addressed are twofold: break-in by an outsider (who has discovered a valid TAC id and password) and misuse by a legitimate user (trying to break into various network hosts). (Authorized TAC users are not generally registered users of every host on the network.)

A prototype NAURS exists on a separate machine from the SRI-NIC host. NAURS is not accessible for remote login or file transfer by network users. A planned production-quality NAURS will feature redundancy of equipment, distribution of functionality (five dedicated workstations have been proposed), ability to perform real-time detection of incidents, and redundancy of the audit database. Plans include reports on trends, such as 6-month differential-use trends of port usage (number of logins, length of sessions). Profiles will be maintained for users and devices. Some of these profiles will be established when an individual becomes a registered user, and others will reflect observed user behavior patterns. Proposed additional incident rules are unexpected host connection for a particular user, long idle periods, excessive connect time, simultaneous TAC logins with the same user id but not necessarily from the same TAC, excessive number of simultaneous logins from the same TAC, unusual time of day for a particular user, excessive number of unsuccessful logins from the same TAC and same user id, excessive number of unsuccessful host logins at different hosts, and excessive number of successful host logins at different hosts during the same TAC session.

3.6 Keystroke Dynamics

International Bioaccess Systems Corporation[6] offers a suite of products, collectively called Bioaccess System 2000, that perform intrusion-detection using keystroke dynamics. Among these are two products BioPassword and BioContinuous for biometric access protection for IBM personal computers (PCs).

Keystroke dynamics technology is based on the "fist of the sender" concept from the days of the telegraph when Morse Code operators could identify a sender by listening to the incoming signals. BioPassword produces an electronic signature based on the unique typing characteristics of each authorized user for keystroke-dynamics verification of user id and password. BioPassword is implemented on a board installed in the CPU socket of the mother board of IBM personal computers; no special keyboard is needed.

A user's electronic signature is stored in nonvolatile memory on the BioPassword board. The first time a person logs on the computer following installation of the board, the signature

[6] Bioaccess, BioPassword, BioContinuous, and BioNet are trademarks of International Bioaccess Systems Corporation.

access to sensitive objects, and attempted access to other people's objects. To detect misuse, MIDAS uses rules involving resource overuse, inactive session, and command out of scope for project. To detect Trojan horses and viruses, MIDAS uses rules involving attempted modification to system files and programs and unusual execution of predictable commands (e.g., `who` taking abnormally long).

A preprocessor on the Multics system formats the audit data for MIDAS. The preprocessor collects audit data from the usual Multics auditing program[5] and from additional audit collection software that was written specifically for MIDAS.

In its current implementation, audit data are accumulated in a Multics file and dumped to tape, and then the tape is fed into MIDAS. A real-time capability is planned for a later implementation phase.

3.4 Ask the Experts

TRW is developing an intrusion-detection system for the U.S. Government using traditional expert system technology [17]. The expert system rules attempt to characterize intrusions, either in general (what TRW calls generic common-sense rules), for the particular organization, or for the particular type of system and installation. The rules are obtained using standard knowledge-engineering techniques such as interviewing and working with system security officers. Known cases of intrusions drive the selection of the rules. The system security officers will be able to add new rules and modify old rules in the rule base.

This expert system is intended to do the work of the system security officer whose job now is to flip through huge printouts of audit trails looking for problem areas. The benefit from the system is expected to be twofold: first, it will be able to analyze data that are too voluminous for the security officer to thoroughly analyze and to spot long-term trends; and second, it will provide a degree of proficiency that would otherwise be scarce, because experienced security officers are rare.

This system uses the audit trail already produced by the monitored system. Once suspicious activity has been identified, the system is intended to be used to build a case against the intruder. It does not operate in real time, but after the fact (like the security officers it mimics).

In one test, a feasibility system using 50 rules exhibited a false positive rate of about 12 percent, but detected the one intrusion planted in the 500 test cases. In addition, it detected at least one problem that had been thus far undetected. A prototype system is under development.

3.5 NAURS

The Network Auditing Usage Reporting System (NAURS) is used in conjunction with the Terminal Access Controller (TAC) Access Control System for the MILNET and the ARPANET [18, 19].

[5] Because Multics is a B2 system, its auditing facilities satisfy the auditing requirements for B2 trusted computing systems as enumerated in the *Department of Defense Trusted Computer System Evaluation Criteria* [16].

encodes *a priori* rules that define an intrusion. This approach is used in the Multics Intrusion Detection and Alerting System (MIDAS), being developed by the National Computer Security Center to monitor a U.S. government Multics system [2].

MIDAS is implemented on a stand-alone Symbolics LISP machine. It uses a home-grown expert system shell, capable of 150 inferences per second, with a forward-chaining inference engine and an explanation facility. Its rules are elaborated in LISP, and statistical user profiles are maintained in LISP structures. The rules are compiled for fast performance. At the time of writing, MIDAS includes about 40 rules.

MIDAS is based on Denning's intrusion-detection model [15]. MIDAS monitors at the user command line level and logs all commands used. MIDAS uses four types of heuristic rules:

- *Immediate*—These are hard-and-fast rules that make no use of information of past or expected user behavior. They are intended to detect those events that, considered in isolation from any other information, are suspicious.

- *Anomaly*—These rules use statistical user profiles to detect when a user's behavior departs from a pattern established by observing past behavior. User profiles are updated at the completion of a user session. The profiles contain a list of the user's usual commands, the usual access times and location for the user, and the expected typing rate for the user. MIDAS also profiles the observed behavior of remote systems.

- *System-wide state*—MIDAS also can maintain a system-wide profile to characterize what is normal for the system globally. For example, an unusually high number of invocations of the `copy` command might indicate suspicious activity.

- *Sensitive path*—A command sequence can be characterized as abnormal if its probability of occurring is sufficiently low. This type of heuristic rule can also be used to determine whether a user's command sequence is similar to those characterizing a known or postulated attack. Attack scenarios are obtained through interviews with system security officers, hackers, and experts in penetration testing. Use of the sensitive-path heuristic rule could enable MIDAS to detect an attack in progress *before* the damage occurs. The sensitive-path type of heuristic rule is not currently implemented on MIDAS.

MIDAS combines different intrusion indicators to decide whether an intrusion is occurring. A login time unusual for a given user, for example, is not alone sufficient to raise an alarm; but if combined with other anomalous data, however, MIDAS might decide an intrusion was in progress.

MIDAS's rules attempt to detect attempted break-ins, masqueraders, penetrators, Trojan horses, viruses, and misuse. To detect attempted break-ins, MIDAS uses rules involving password failure on a system account, login failure with an unknown user name, login attempt from outside the continental United States, and login attempt to a locked account. To detect masqueraders, MIDAS uses rules involving unusual login (e.g., time, location, terminal type), unusual commands or command patterns, invalid commands, and user logged in simultaneously from different locations. To detect penetrators, MIDAS uses rules involving attempted use of sensitive commands, attempted use of unauthorized commands, attempted

continuous measure is one whose value is a number or count that accumulates over a user session (e.g., connect time, CPU time, and I/O activity).

To determine whether user behavior as reported in the audit data is normal with respect to past or acceptable behavior, IDES includes user behavior profiles for the measures. (A *profile* is a description of the expected behavior of a user with respect to a particular measure.) The profiles are periodically updated based on observed user behavior, and the profile data are aged using a decay factor that gives the data a half-life of 50 days. Thus, the profile reflects a moving time window of behavior for each user. *Anomalous* behavior is user behavior that deviates from the expected 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, by 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 change. This capability makes IDES a flexible system: 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 currently monitors a DEC-2065 machine at SRI running a locally customized version of the TOPS-20 operating system[3]. SRI 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 according to the IDES protocol.

IDES's flexible system-independent audit record format and protocol for the transmission of audit records make it adaptable to different host systems without fundamental alteration (although the particular measures and parameters chosen will depend on the system and users being monitored). SRI's plans are to adapt IDES to monitor a network of Sun workstations and to monitor a large IBM mainframe system running MVS.

Now that the framework has been established, adding additional intrusion-detection measures to IDES is straightforward. In ongoing work, SRI is implementing a greater variety of intrusion-detection measures, including some "second order" measures to detect behavior trends, thereby improving the intrusion-detection capability of IDES. In addition, an expert system and rule-base that encodes information about hypothesized intrusion scenarios and suspicious behavior is being added to IDES.

The IDES intrusion-detection processes are implemented on a Sun 3/260, and the IDES security administrator interface is implemented on a Sun 3/60[4]. The security administrator interface 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.

3.3 MIDAS

SRI's IDES prototype detects intrusions by flagging user behavior that deviates from that user's past behavior. Another approach is to develop an intrusion-detection system that

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

[4]Sun 3/260 and Sun 3/60 are registered trademarks of Sun Microsystems, Inc.

3.1 A Priori Rules for Normal Program Behavior

Paul Karger has suggested what he calls a *knowledge-based name checker* to compare the names and types of objects requested (for reading, writing, creation, or destruction) by a program with the names and types of objects expected for the program [13]. He posits as an example a FORTRAN compiler containing a Trojan horse that surreptitiously modifies a user's LOGIN.CMD file while compiling the user's program. The name checker expects the FORTRAN compiler to require read access to a file with a user-supplied name and a suffix of .FOR and to create new files with the same name but suffixes of .OBJ and .LIS. If the compiler attempts to create or to write to a file named LOGIN.CMD, the name checker would recognize that such a file is unexpected for the FORTRAN compiler. Other rules, for a Unix system for example, could check whether a user program asks for `set-uid` privileges.

Although Karger envisions the name checker being used for access control decisions, it could also be used as a rule-based form of real-time intrusion-detection. He suggests obtaining the rules for the behavior expected of commands from information already known to the computer system; for example, in VAX/VMS from the command definition tables. For user programs and batch jobs, the user would encode the rules in what Karger calls a *special directory tree*, which would enumerate the objects on which the program is expected to operate.

3.2 IDES

SRI International is developing a prototype intrusion-detection system called IDES (Intrusion-Detection Expert System) [11, 12]. The goal of IDES is to provide a system-independent mechanism for real-time detection of all types 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. The IDES approach is 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. The IDES prototype is based on the IDES model developed by Dorothy Denning [14, 15]. 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.

The IDES prototype is an independent system that runs on its own hardware (a Sun Workstation[2]) and processes audit data received in real time from a target system [11, 12]. The user activity monitored by the IDES prototype includes login, logout, program execution, directory modification, file access, system call, session location change, and network activity.

IDES is driven by the arrival of audit records, each of which describes behavior relevant to possibly several intrusion-detection measures. (A *measure* is an aspect of user behavior.) 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, for example location and time of login. A

[2]Sun Workstation is a trademark of Sun Microsystems, Inc.

business hours, an appropriate measure might simply track whether activity is during normal hours or off hours. Other users might frequently login in the evenings as well, yet still have a distinctive pattern of use (e.g., logging in between 7 and 9 p.m. 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.

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). 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 user's range of normal behavior. 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 user behavior changes over time, may be useful in detecting such attacks.

Because the task of discriminating between normal and intrusive behavior is so difficult, another study has taken the straightforward approach of automating the security officer's job. Such an approach lends itself to traditional *expert system* technology, in which the special knowledge of the intrusion-detection experts (the system security officers) is codified as rules used to analyze the audit data for suspicious activity. The obvious drawback to this approach is that the security officers, in practice, have only limited expertise. Thus, while automating these rules frees the security officer to perform further analysis, such rules cannot be expected to be comprehensive. This approach would be more aptly called a security officer's assistant.

Several study teams are attempting to comprehensively characterize intrusions (e.g., MIDAS [2]). 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 three consecutive unsuccessful login attempts for the same user id within 5 minutes is a penetration attempt.) Audit data from the monitored system are matched against these rules to determine whether the behavior is suspicious. A limitation of this approach is that it looks for known vulnerabilities and attacks, and the greatest threat may be unknown vulnerabilities and the attacks that have not yet been tried; one is in a position of playing catch-up with the intruders.

Below is a survey of these various intrusion-detection systems.

CPU time, time of day, etc.) of normal jobs for that user. They assumed a certain multivariate probability distribution for intrusive jobs. They then used classical statistical paradigms to determine a rule for classifying a job as normal or abnormal. With this approach, every point in the multivariate space is assigned a value equal to the ratio of the height of the intrusive job probability distribution to the height of the normal job probability distribution. The points with the largest ratios 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.

Because audit data containing intrusive jobs were not available, two different approaches were taken to determine a hypothetical multivariate probability distribution for intrusive jobs:

- Nonintrusive profile approach: Assume that intrusive jobs have a uniform distribution
- Surrogate intrusive profile approach: Use other users' jobs to develop a probability distribution for intrusive jobs

For MVS jobs, discriminant analysis produced a true positive rate of over 90 percent and a false positive rate of only 6 or 7 percent. For VM sessions, although the single parameter rules had by far the lowest false positive rates (averaging less than 1 percent), the discriminant analysis method had a much higher true positive rate (over 80 percent). The false positive rate was shown to increase with the number of days since the profile was last updated. The SRI group estimated that with daily profile updating the results would be even better. If additional security-relevant audit data were used in the analysis, they estimated that a discriminant analysis would produce a true positive rate of 90 to 98 percent and a false positive rate of 1 to 3 percent. Thus, these statistical procedures are potentially capable of reducing the audit trail by a factor of 100 while detecting approximately 95 percent of all intrusions [3]. The security officer would still have to determine whether the statistical abnormalities represent actual intrusions.

3 The Intrusion-Detection Systems

The early evidence of the Sytek and Javitz studies was the basis for a real-time intrusion-detection system, that is, a system that can continuously monitor user behavior and detect suspicious behavior as it occurs. This system, known as IDES (Intrusion-Detection Expert System), is based on the approach that intrusions, whether successful or attempted, can be detected by flagging departures from historically established norms of behavior for individual users [11, 12].

Another real-time approach, called *keystroke dynamics*, is based on measurements of certain characteristics, such as typing speed, of a user's keyboard activity. Keystroke dynamics has been found to be a powerful means of continuously verifying the identity of the user doing the typing.

For systems like IDES, different intrusion-detection measures may be appropriate to different classes of user. For example, for users whose activity is almost always during normal

run, number of files accessed, number of changes of index levels, and number of VSAM records inserted, retrieved, updated, or deleted.

They created a database containing 4 weeks of audit data for a set of representative users. They selected a job as the unit of analysis for batch users and a session for interactive users (both are referred to as jobs hereinafter) and characterized each job by variables such as CPU time used, terminal used, and the number of files accessed. They then used these variables to construct a profile, representing the statistical norm for the user's jobs, for each user.

The SRI researchers developed statistical techniques to detect jobs that deviated from the profiles developed for each user's jobs. The statistical algorithms aimed to maximize the *true positive rate* (percentage of intrusions identified as abnormal) and minimize the *false positive rate* (percentage of normal use identified as abnormal). A true positive rate of 100 percent means that all intrusions are detected. The *detection ratio* is the true positive rate divided by the false positive rate. The more variable user behavior is, the higher the false positive rate will be. 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.

They compared the results obtained when the following criteria were used to define an abnormal job:

- A job is abnormal if the value of one of its parameters has a probability of occurring of less than 10 percent for that user.

- A user's job is abnormal if a discriminant analysis shows that the probability that the job was submitted by another user is greater than the probability that the job was submitted by the user in question.

They split the 4 weeks of audit data into two sets: the training set (the first 2 weeks of data), used to determine the statistical thresholds for the test, and the verification set (second 2 weeks of data), used to measure the test's performance.

Because all activity represented in the sample database was assumed to be normal, there were no examples of intrusive behavior. Thus, for each measure the researchers calculated a *surrogate* true positive rate (the probability of identifying a user's job as abnormal when measured against another user's job profile) and a *surrogate* false positive rate (the probability of identifying a user's own job as abnormal). These are the true and false positive rates discussed below.

For VM sessions, by far the best indicators were the type of login and the terminal used; both of these had extremely low false positive rates (and low to medium true positive rates, but very high detection ratios). Minidisk id had an extremely high true positive rate, but also a high false positive rate. The computer system used had a fairly high true positive rate and a low false positive rate, with a detection ratio of seven. Most other characteristics had detection ratios of between one and two.

Discriminant analysis was superior to measures of job abnormality based on extreme values of job parameters. In the discriminant analysis, the researchers used a user's training set to estimate the multivariate probability distribution (with respect to parameters such as

records within a session that were flagged by the feature. They then computed a certainty for a session as the sum of the certainties for all 12 features.

In tests to analyze for intrusion-detection strength and false-alarm rate, the pattern classifier successfully detected all the simulated intrusions. However, the false-alarm rate was high (between 40 and 70 percent). Much better performance could be expected if a longer training period were used.

Four of the selected features pertained to a user's command usage patterns. These features were very good at detecting the intrusion scenarios but had very high false-alarm rates. Believing that command usage patterns were potentially very useful in discriminating between normal and abnormal behavior, Sytek decided to modify these features to improve their performance. It decided to make these features *fuzzy*; that is, to allow the computed value for a user's session to be a certain distance from the range in the user's profile before the session was flagged as abnormal. The greater the fuzziness, however, the greater the chance of missing intrusions. For each of the four measures, Sytek analyzed the effect of increasing the fuzziness on both the intrusion-detection strength (percentage of scenarios flagged) and on false-alarm rate (percentage of normal sessions flagged). To reach an acceptable false-alarm rate, the intrusion-detection strength was also greatly reduced. Sytek found, however, that for features *proportional real duration of command* (percentage of a session's real elapsed time spent in the command) and *proportional CPU duration of command* (percent of a session's CPU time spent in the command), an acceptable false-alarm rate was achieved with a relatively modest reduction in intrusion-detection strength. Hence, these two features showed considerable promise as discriminators of intrusive behavior.

2.2 The SRI Study

A group at SRI led by Hal Javitz performed an extensive statistical analysis on audit data from IBM systems running MVS and VM. The purpose of the study was to develop analytical statistical techniques for screening computer system accounting data to detect user behavior indicative of intrusions. A high-speed algorithm was developed that could accurately discriminate among users based on their behavior profiles.

Audit data were obtained from normal system accounting records for IBM VM and MVS systems. Because the overwhelming majority of information in the accounting records concerned system usage parameters that either were beyond the control of the user, bore no reasonable relationship to intrusion attempts, or were redundant with information contained on other accounting records, subset of the accounting data was selected that could be used to detect logon activity outside the user's normal range, execution time outside the user's normal range, activity from other than a user's normal set of input or output (I/O) devices, and files for which the user is authorized but which are only infrequently accessed by the user.

From VM accounting records the research group obtained logon time and day of week, logon mode (normal or autolog), terminal used, minidisks accessed, connect time, computer system accessed, CPU time, number of page reads and writes, number of virtual cards read and punched, and number of virtual lines printed. From MVS accounting records they obtained time and day of week, computer system used, job priority, CPU time, number of job steps, I/O units, card reader class and type, number of card images, number of programs

determined from the audit data). The study then tested the features for their ability to discriminate between normal sessions and sessions containing staged intrusions. A session was flagged as intrusive by a feature if the value of the feature calculated for the session was outside the user's range or set of expected values. Features that successfully detected the staged intrusions were combined to create a for each user *user profile*—the collection of the normal ranges for each feature.

Sytek wrote software to collect audit data from a Unix[1] system that were analogous to data available in general-purpose operating systems. An audit record containing the command name, associated files, process statistics, and file statistics was generated whenever a user issued a command.

The Sytek team collected one week of audit data and generated a set of statistics to identify features of the audit trail that were potentially useful in discriminating among users [6]. Each identified feature was trained on the presumedly normal audit data to establish a range or set of expected values exhibited during each user's sessions. The Sytek team then enacted and audited various intrusion scenarios in such a way that the intrusions were embedded into the audited behavior of legitimate users of the system [7]. The intrusion scenarios included break-ins by outsiders, intruders and legitimate users masquerading as other users, and users deliberately subverting the system in various ways. Sytek then tested the selected features to see whether they were useful in detecting the simulated intrusions [8]. Those features that detected one or more of the simulated intrusions were retained for further study.

Sytek then tested each feature still under consideration against an additional week of (presumedly normal) audit data to determine the percentage of normal sessions the feature flagged as abnormal (i.e., the false-alarm rate). Sytek found that the features *password changed*, *user identity queried*, and *access to system dictionary* performed extremely well. It found the most effective file statistics were *device on which the accessed file resides*, *file size*, *oversized file associated with this command*, *group id of the owner of the accessed file*, and *user id of the owner of the accessed file*. The most effective process statistics were *time of use*, *day of use*, *user program CPU time*, and *maximum total memory use*. These 12 features had low (under 15 percent) false-alarm rates and were selected for use in a pattern classifier that analyzed their composite performance [9, 10].

The pattern classifier flagged those sessions that did not fall within the pattern defined by the user profiles. The idea was that the resultant set of flagged sessions should be sufficiently small to enable a security officer to examine the set manually.

The performance of the pattern classifier could be different from that of the individual features taken separately, because (1) if several features individually each flagged a certain session, the composite would flag that session only once, so the composite could flag *fewer* normal sessions and thus have *better* performance than the features taken individually and (2) one feature might not flag the same normal sessions as another feature, so the combination of features could flag *more* normal sessions and thus have *worse* performance than the features taken individually.

Sytek also attempted to compute a *certainty measure* that would indicate the degree of certainty that a flagged session actually represented an intrusion or the degree of suspicion for a user session. They computed a certainty for each feature, namely, the number of audit

[1] Unix is a trademark of AT&T.

Anderson suggested that external penetrators could be detected by auditing failed login attempts and that some would-be internal penetrators could be detected by observing failed access attempts to files, programs, and other resources. He suggested that masqueraders could be detected by observing departures from established patterns of use for individual users. All of these approaches have been adopted in subsequent studies.

Anderson offered no suggestions for detecting legitimate users who abuse their privileges. To detect such abuse however, *a priori* rules for *acceptable* behavior could be established; this approach has been taken in a few studies. Comparison with the norm established for the class of user to which the user belongs also could detect abuse of privilege; this approach is under consideration by the research group at SRI.

The clandestine user can evade auditing by using system privilege or by operating at a level below which auditing occurs. The former might 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 might be addressed by low-level auditing, such as auditing system service or kernel calls. Anderson suggested monitoring certain system-wide parameters, such as CPU, memory, and disk activity, and comparing these with what has been historically established as usual or normal for that facility. At least one subsequent study has included this approach [2].

2 The Experiments

Subsequent to Anderson's study, early work focused on developing procedures and algorithms for automating the offline security analysis of audit trails. The aim of such algorithms and procedures was to provide automated tools to help the security administrator in his or her daily assessment of the previous day's computer system activity [3, 4]. One such project used existing audit trails and studied possible approaches for building automated tools for audit trail security analysis [3]. Another such project considered building special security audit trails and studied possible approaches for their automated analysis [4]. These projects provided the first experimental evidence that users could be distinguished from one another based on their patterns of use of the computer system [3], and that user behavior characteristics could be found that could be used to discriminate between normal user behavior and a variety of simulated intrusions [4].

2.1 The Sytek Work

A tool that ranked user sessions by their suspiciousness would allow the system security officer to analyze audit trail records that are most likely to represent intrusions without having to wade through volumes of records of mostly normal user activity. The Sytek work sought to provide a feasibility demonstration for such a tool [5].

Sytek's work was guided by concepts from pattern recognition theory. User sessions were recognized as normal or intrusive based on patterns formed by the individual records on the audit trail for that session. The Sytek study defined several audit record *features* as functions of the audit record fields. For each user, expected values for the features were determined through a process called *training* (that is, for each feature, the set or range of values was

11th National Computer Security Conference, October, 1988
Outstanding Paper Award

Automated Audit Trail Analysis and Intrusion Detection: A Survey

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

Abstract

Today's computer systems are vulnerable to both abuse by insiders and penetration by outsiders, as evidenced by the growing number of incidents reported in the press. Because closing all security loopholes from today's systems is infeasible, and since no combination of technologies can prevent legitimate users from abusing their authority in a system, auditing is viewed as the last line of defense. What is needed are automated tools to analyze the vast amount of audit data for suspicious user behavior. This paper presents a survey of the automated audit trail analysis techniques and intrusion-detection systems that have emerged in the past several years.

1 Introduction

The last few years have seen a sudden and growing interest in automated security analysis of computer system audit trails and in systems for real-time intrusion detection. There is a growing number of research activities devoted to the subject, and some operational systems and even a few commercial products have appeared.

The earliest work on the subject was a study by Jim Anderson [1], who categorized the threats that could be addressed by audit trail analysis as

- External penetrators (who are not authorized to use the computer)
- Internal penetrators (who are authorized to use the computer but not 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 (who are authorized to use the computer *and* resources accessed but misuse their privileges)