[15] T. F. Lunt. Using statistics to track intruders. In *Proceedings of the Joint Statistical Meetings of the American Statistical Association*, August 1990.

[16] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann, and C. Jalali. IDES: A progress report. In *Proceedings of the Sixth Annual Computer Security Applications Conference*, December 1990.

[17] T. F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, H. S. Javitz, A. Valdes, and P. G. Neumann. *A Real-Time Intrusion-Detection Expert System.* Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 1990.

[18] T. F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, H. S. Javitz, A. Valdes, P. G. Neumann, and T. D. Garvey. *A Real-Time Intrusion-Detection Expert System (IDES), Final Technical Report.* Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[19] 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.

[20] J. Picciotto. The design of an effective auditing subsystem. In *Proceedings of the 1987 Symposium on Research in Security and Privacy*, April 1987.

[21] C. Stoll. What do you feed a Trojan horse? In *Proceedings of the 10th National Computer Security Conference,Baltimore, Maryland*, September 1987.

[22] J. van Horne and L. Halme. *Analysis of Computer System Audit Trails — Final Report.* Technical Report TR-85007, Sytek, Mountain View, California, May 1986.

report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

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

[3] P. T. Cummings, D. A. Fullam, M. J. Goldstein, M. J. Gosselin, J. Picciotto, P. L. Woodward, and J. Wynn. Compartmented mode workstation: Results through prototyping. In *Proceedings of the 1987 Symposium on Research in Security and Privacy*, April 1987.

[4] D. E. Denning, P. G. Neumann, and Donn B. Parker. Social aspects of computer security. In *Proceedings of the 10th National Computer Security Conference*, September 1987.

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

[6] R. H. Irving, C. A. Higgins, and F. R. Safayeni. Computerized performance monitoring systems: Use and abuse. *Communications of the ACM*, 29(8), 1986.

[7] R. Jagannathan, T. F. Lunt, F. Gilham, A. Tamaru, C. Jalali, P. Neumann, D. Anderson, T. D. Garvey, and J. Lowrance. *Requirements Specification: Next Generation Intrusion-Detection Expert System (NIDES).* Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, September 1992.

[8] R. Jagannathan, A. Tamaru, F. Gilham, D. Anderson, C. Jalali, C. Dodd, H. S. Javitz, A. Valdes, T. F. Lunt, and P. G. Neumann. *Next Generation Intrusion-Detection Expert System (NIDES) Software Design Specifications.* Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, March 1993.

[9] H. S. Javitz and A. Valdes. The SRI statistical anomaly detector. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, May 1991.

[10] H. S. Javitz, A. Valdes, T. F. Lunt, A. Tamaru, M. Tyson, and J. Lowrance. *Next Generation Intrusion-Detection Expert System (NIDES): Statistical Algorithms Rationale and Rationale for Proposed Resolver.* Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, March 1993.

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

[12] J. D. Kuhn. Research toward intrusion detection through the automated abstraction of audit data. In *Proceedings of the 9th National Computer Security Conference*, September 1986.

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

[14] T. F. Lunt. Automated audit trail analysis and intrusion detection: A survey. In *Proceedings of the 11th National Computer Security Conference*, October 1988.

violations, whether they are initiated by outsiders who attempt to break into a system or by insiders who attempt to misuse their privileges. NIDES is designed to be independent of any particular target system, application environment, level of audit data (e.g., user level or network level), system vulnerability, or type of intrusion, thereby providing a framework for a general-purpose intrusion-detection system using real-time analysis of audit data.

Each target system must install an agen facility (see section 4.1) to collect audit data and put them into NIDES's generic audit record format. We have developed a flexible audit record format and a protocol for the transmission of audit records from the target system. The NIDES protocol and its audit record format are system-independent; our intent is that NIDES can be used to monitor different systems (even simultaneously) without fundamental alteration.

5 Conclusions

Intrusions can be detected by detecting departures from users' normal behavior patterns. In addition, a rule-based approach in which rules characterizing intrusive behavior are constructed for evaluation against observed user behavior can be used. The strength of the first approach is that intrusive behavior that shows up in unforeseen ways can potentially be detected; the weakness is that certain behaviors generally agreed to be abusive or suspicious are not easily monitored for. The strength of the second approach is the ease of stating exactly that behavior that is considered intrusive or undesirable; conversely, its weakness is that only behavior that has been foreseen to be intrusive will be caught: novel or highly sophisticated attacks may go undetected. In addition, the use of other approaches, such as model-based reasoning and neural networks, appears to be promising.

In order to effectively address the various intrusion threats, a system should combine several intrusion-detection approaches. We should begin to see intrusion-detection systems that can intelligently make use of audit data gathered at several different levels from the monitored system (e.g., system call level, command line level, and application level). Profiling files and programs will give us another dimension along which to characterize expected behavior on a system. And there still remains a significant amount of research to be done in determining exactly which aspects of behavior are most indicative of intrusions. To obtain meaningful indicators of intrusive behavior, such research needs to have available many examples of actual intrusions. A library of such examples does not currently exist and is needed. As intrusion-detection systems such as SRI's NIDES become available, we can expect to learn much more as experience is gained through their use.

As the computing workplace changes, we should also begin to see intrusion-detection systems that can monitor networks of user workstations and integrate user behavior observed concurrently on several different machines. An hand-in-hand with improved intrusion-detection capabilities, we can expect to see vastly improved auditing facilities optimized for security analysis.

References

[1] J. P. Anderson. *Computer Security Threat Monitoring and Surveillance.* Technical

may know of others), the combination of the statistical and rulebased approaches is intended to provide comprehensive coverage, providing the ability to detect specific actions that are known to be suspicious (via the rulebased component), masqueraders, and unanticipated or unknown intrusion methods (via the statistical component).

4.2.3 Resolver Component

The resolver screens the alarms generated by the statistical and rulebased components before reporting them to the NIDES security officer user interface. The statistical and rulebased components generate alarms independently of each other and, to a large extent, independently of earlier alarms already reported. Because these alarms do not always pertain to independent events, and because they can come very quickly one after another, the resolver performs filtering using a set of simple heuristics. Once the statistical threshold is passed and the first alarm is generated, the resolver suppresses subsequent alarms until the statistical score drops below the threshold, or until the score passes the next-higher threshold, or until there is a change in the most anomalous measure, or until the rulebased component also reports an alarm.

4.3 NIDES Security Officer User Interface

NIDES includes a comprehensive user interface written to operate under the X-Window system. The user interface is based on the MOTIF toolkit. Access to the various NIDES functions is provided via pulldown menus, point-and-click selections, and occasional text entry. Included in the current user interface is a comprehensive multitiered context-sensitive help system. NIDES also includes a comprehensive user's manual and tutorial.

4.4 NIDES Infrastructural Components

To support data and process management, NIDES includes the following set of infrastructural components:

- **Data storage and retrieval.** The NIDES persistent storage component provides functions for the storage and retrieval of statistical profiles and configuration information. The beta release of NIDES will also provide functions for the storage and retrieval of audit data and analysis results.

- **Interprocess communication.** NIDES components communicate via remote procedure calls (RPCs). Communication between the various NIDES processes and management of those processes is facilitated by a group of agent processes that pass information among the NIDES component processes.

4.5 NIDES Architecture

NIDES runs independently on its own hardware, the current version on a Sun workstation, and processes audit data received from a target system via a network (see Figure 2). NIDES is intended to provide a system-independent mechanism for real-time detection of security

normally monitors a group of targethost systems. For each targethost that provides audit data to NIDES, an agen process converts audit data in the targethost's native audit record format to a generic audit data format used by NIDES. The agen process then transmits the NIDES-formatted audit data to the NIDES arpool process.

- **Arpool**, which stands for *audit record pool*, is a process that resides on NIDES. Using an RPC transfer mechanism, arpool receives data from multiple targethosts via their respective agen processes. Arpool then coalesces the data into a single audit record stream and provides those data to the NIDES analysis components.

4.2 NIDES Analysis Components

NIDES has three analysis components. These are described below.

4.2.1 Statistical Analysis Component

The statistical analysis component flags as suspicious an entity's currently observed behavior if it deviates sufficiently from that entity's historically observed behavior. Each entity's historical statistical profile of behavior is aged so that the oldest behavior contributes less to the decision making. NIDES updates the profiles daily (or according to an update schedule defined by the security officer) using the most recently observed short-term behavior, so that the profiles adapt to changes in user behavior for the most accurate statistical signature possible. A profiled entity can be a user, a workstation, a network of workstations, a remote host, a group of users, and so on. The basis for this type of analysis is that an intruder masquerading as a legitimate user can be detected because of his or her different patterns of behavior; in many cases, such intruders may exploit previously unknown vulnerabilities and could not be detected by any other means. Likewise, statistical anomaly detection may be the only system-independent means of detecting malicious insiders, on the hypothesis that such users' short-term behavior, while they are engaged in their malicious acts, will deviate sufficiently from their long-term normal behavior or the behavior of a group within which they are being monitored. In addition, statistical anomaly detection can turn up interesting and unusual events that could lead to security-relevant discoveries upon investigation by a security officer.

4.2.2 Rulebased Analysis Component

The rulebased analysis component uses an expert system to detect attempts to exploit known security vulnerabilities of the monitored systems. The rulebased component encodes known intrusion scenarios and target system vulnerabilities that can be exploited by an intruder, or specific patterns of behavior that are known to be suspicious. Observed activity that matches any of these predefined behaviors is flagged. Because the rulebase can quickly become outdated, and is largely target-system dependent, it can be modified and augmented to customize it for a given environment. Even from one release to another of the same operating system, many new vulnerabilities are introduced, and rules relevant to these new vulnerabilities can be added to the rulebase. Because the intrusive behaviors detected by an expert system are limited to those that the knowledge source knows about (seasoned intruders

an IBM mainframe, and trusted Xenix PCs. An alpha release of NIDES was delivered to various NIDES users in the first quarter of 1993. A beta release, which includes many new features, is scheduled for delivery sometime during the first quarter of 1994.

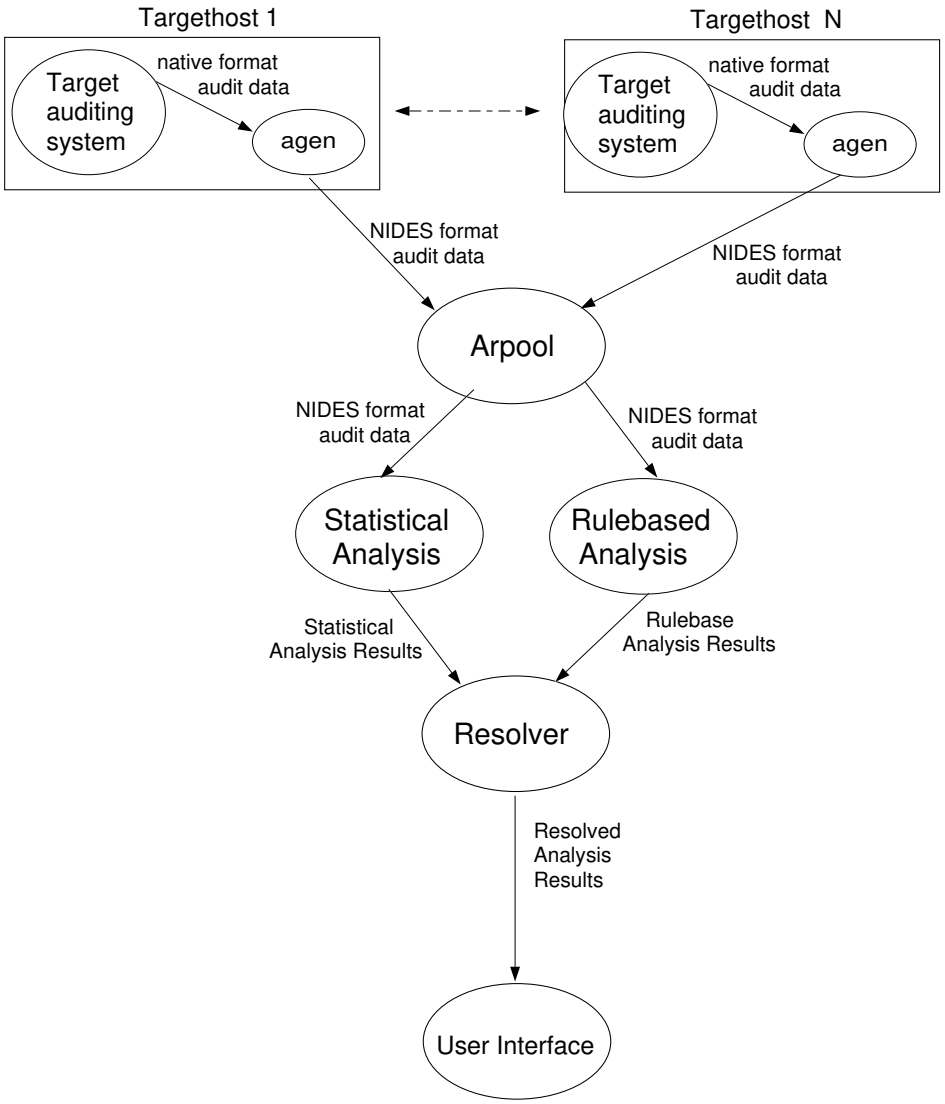

Figure 2: NIDES Component Data Flow

NIDES comprises audit data generation and collection components, an analysis component that includes statistical and rulebased analysis methods, a user interface component, and various infrastructural components to support data and process management. Figure 2 shows the flow of data between the various NIDES components.

4.1 NIDES Audit Data Collection Components

The two key audit data collection components within NIDES are agen and arpool.

- **Agen**, which stands for *audit data generation*, typically resides on what we call a targethost. The targethost system provides audit data to NIDES for analysis. NIDES

which may be relevant to an intrusion-detection analysis, it makes sense not to flood the intrusion-detection system with all of the audit data for it to sift through, but to preprocess the audit data on the monitored system before transmitting it to the intrusion-detection system. This drastically reduces both the storage and performance requirements of the intrusion-detection system.

Preprocessing the audit data on the monitored system also means that a generic audit record format could be established for the intrusion-detection system. The monitored system would format the selected data into the generic format. This opens the possibility that the intrusion-detection system could monitor more than one system or even one type of system — any system that can audit the desired data and put it into the desired format could be monitored.

3.3 Privacy Issues

Some people have voiced privacy concerns about the use of monitoring for security purposes. Some have even suggested that security measures such as intrusion-detection mechanisms may actually *increase* the threat of computer abuse by engendering employee dissatisfaction; whence the emergence of the disgruntled employee and the so-called *insider threat* [4]. There are privacy issues in even such apparently benign security mechanisms as file backups, archives, and keeping an audit trail of user activity (as is required by the DoD Trusted Computer System Evaluation Criteria [5] for systems rated C2 and above), in that there is potential for abuse of such data. Real-time intrusion-detection systems make possible an even greater degree of invasion of privacy and other potential objectionable activity, such as employee performance monitoring. One study indicated that the use of computerized employee performance monitoring systems can lead to increased stress, lower levels of satisfaction, and a decrease in the quality of relationships with peers and management among the monitored workers [6]. However, this same study found that workers were not opposed to computerized performance monitoring in principle, but that it is how it is used by management that determines the effects. These findings underline the need for concern for appropriate use of intrusion-detection technology. Proposed legislation in the United States would require employee notification of all activity monitored, all information collected, and the use to which the data obtained from monitoring would be put. However, there are many, particularly in the intelligence community, who consider that even knowledge of the existence of an intrusion-detection system might be an aid to a would-be penetrator.

4 SRI's NIDES Intrusion-Detection System

SRI's Next Generation Intrusion-Detection Expert System (NIDES) is a generic intrusion-detection system that analyzes data collected from monitored systems to detect a variety of suspicious user behavior. NIDES uses innovative statistical algorithms for anomaly detection as well as a rulebased component that encodes known intrusion scenarios [8, 9, 7, 10]. NIDES is designed to operate in real time to detect intrusions as they occur. NIDES has been under development at SRI for a number of years, beginning with early feasibility experiments and culminating in the current prototype that can monitor a set of networked Unix workstations,

Several projects have chosen to audit at the command line. This level of auditing 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. And with this level of auditing, a security officer can scan a user's audit records to get a "feel" for what has happened.

The audit subsystem designed for the compartmented mode workstation [3] is implemented at three different levels: the Unix operating system kernel calls; at the interface between the user and the operating system; and within applications programs [20]. The applications-level auditing is performed by certain "trusted" applications. The intention is that the audit trail will thus be easier to comprehend, and the volume of audit data will be reduced. Examples of such applications in the compartmented mode workstation are the window manager and a "trusted" database management system. These applications perform their own auditing, and are permitted to suspend the lower-level auditing of their activity. At the level of the user/operating system interface, certain system calls are audited. In addition, certain kernel calls and their subroutines perform their own auditing, and any kernel routines that require a privilege to execute also perform internal auditing. The compartmented mode workstation also has a program, called *Redux*, that selectively retrieves audit data based on user id, object(s) accessed, the classification of the object(s) accessed, and the event (the particular command, system call, or kernel routine). No analysis or intrusion detection is performed.

When an intruder is suspected, several researchers have emphasized the importance of being able to play back a terminal session from the audit data exactly as it originally occurred [2], being able to obtain a listing of all characters input and output on the affected communications line [21], or being able to view a single user's audit records for a session contiguously [1] for more detailed investigation when an anomaly is flagged. Some claim that effective investigation to confirm suspicion or establish innocence depends on evidence gathered at the keystroke and system response level [2]. However, the ability to play back a user session or gain an understanding of what has happened by viewing a user's audit records for a session depends on auditing at the command line or application level.

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 session.

3.2 Analysis on a Separate Machine

Implementing the audit trail analysis and intrusion detection mechanisms on a machine separate from the system being monitored has both performance and security advantages. The performance advantage is that the intrusion-detection analysis would not degrade the response time of the monitored system or otherwise affect its behavior. And a standalone system could be made more tamper-resistant from would-be intruders whose activity is being monitored, so that any security flaws that exist in the monitored system should not allow a penetration of the separate intrusion-detection system.

Because most computer systems collect vast amounts of audit data, only a fraction of

What is evident from the various intrusion-detection studies is that specialized audit trails are needed for security purposes, to which only those data relevant to intrusion detection are reported. Moreover, in addition to the raw audit data itself, the following additional data is necessary or helpful in distinguishing suspicious activity from normal activity:

- External facts, including facts about changes in user status, new users, and terminated users, users on vacations, changed job assignments, user locations, etc.
- Supporting facts about files, directories, devices, and authorizations.
- Profiles of expected or socially acceptable behavior.

Correlation of audit data with other available data may help in detecting intrusion attempts. Anderson suggests, for example, that data from electronic access systems that record the time and point of entry and exit of individuals to a building could be used to detect someone trying to log in from a hard-wired local terminal who is not physically in the building. Other 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 in judging whether observed behavior is suspicious.

Independent audit trails for the operating system, database management system, and applications make it more difficult for intruders to evade auditing.

Good intrusion-detection systems will not come into widespread use until good security auditing mechanisms have been developed and are in use that make the relevant data available for analysis. Not only does the relevant data need to be captured, but the audit mechanisms should be made tamper-resistant and non-bypassable insofar as possible.

3.1.1 Appropriate Level of Auditing

There is considerable difference of opinion as to what is the appropriate level of auditing. Auditing could be performed for low-level system calls, or for the command names and arguments typed at a terminal by a user. Or all keystrokes and system responses could be audited. Different studies and projects have used each of these levels of auditing, and some have used more than one. Each level has its strengths and weaknesses 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.

As Jim Anderson points out, users, particularly those with direct programming access, may operate at a level of control that bypasses the auditing and access controls [1]. In order to detect intruders operating at such a low level, auditing should be performed at the lowest level possible.

Jeffrey Kuhn recommends gathering data at the lowest possible levels, because then it is harder to circumvent auditing. He recommends monitoring system service calls, rather than application-level monitoring or command-line monitoring [12]. He argues that 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. And he argues that since users can write programs to access files directly without leaving any trace in the application audit logs, auditing at the application level will not detect all user activity.

An example of such a rule might be that more than three consecutive unsuccessful login attempts for the same userid within five minutes is a penetration attempt. Audit data from the monitored system is matched against these rules to determine whether the behavior is suspicious.

The rule-based approach also has limitations. An obvious limitation is that we are 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; we are in a position of playing "catch-up" with the intruders. Writing such a rule-based system is a knowledge-engineering problem, and the resulting "expert system" will be no better than the knowledge and the reasoning principles it incorporates. An intrusion scenario that does not trigger a rule will not be detected by the rule-based approach. Besides this, maintaining a complex rule-based system can be as difficult as maintaining any other piece of software of comparable magnitude, especially if the system depends heavily on procedural extensions such as rule ranking and deleting facts.

2.6 Other Approaches

Yet another approach was suggested that would define acceptable, as opposed to suspicious, behavior [11]. Another proposal is to introduce trap doors for intruders, namely, "bogus" user accounts with "magic" passwords, that sound an alarm whenever someone attempts to use them [13]. This technique can be extended to include "tripwire" files, phony passwords as bait on electronic bulletin boards, and other similar decoys.

None of the intrusion-detection approaches described is sufficient alone — each addresses a different threat. A successful intrusion-detection system should incorporate several of them.

A skilled penetrator will be able to disable the audit mechanisms in order to work undetected. However, auditing and intrusion-detection mechanisms are still of value in detecting the less skilled penetrator, because they increase the difficulty of penetration. In addition, intrusion-detection systems have great utility in a risk reduction program.

3 System Issues

3.1 The Audit Data

Although existing audit trails (i.e., those not designed specifically for security purposes) can be of some use in intrusion-detection, 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 [19]. The particular data that should be audited may depend on the application. For example, users may have identifiable patterns of access to data or of invocation of functions within an application. Because audit data is so voluminous and little is known about how to analyze it for intrusions in a reasonable amount of time, Anderson has suggested that auditing by random sampling might be a reasonable approach (like the random auditing of taxpayers by the Internal Revenue Service) [1].

- *High cost of algorithm development.* The development time for devising new statistical algorithms and building new software is significant. It is costly to reconstruct the statistical algorithms and to rebuild the software implementing them. We may remove assumptions that are invalid for the audit data we are using, but we may find that we have to modify the algorithms yet again when we apply them to a new user community with different behavior characteristics. Neural network simulators are easier to modify for new user communities.

- *Difficulty in scaling.* New problems are anticipated in applying statistical approaches such as that used in SRI's NIDES prototype to very large communities, for example, thousands of users. Methods are needed for assigning individuals to groups on the basis of similarity of behavior, so that group profiles, instead of a profile for each user, may be maintained. Although users can be grouped manually according to job title, shift, responsibilities, and so forth, this may be inadequate. A neural network could be used to classify users according to their actual observed behavior, thus making group monitoring more effective.

Although the use of neural networks seems to be promising for intrusion detection, initial experimental results indicate that a neural network can simply replace statistical analysis for intrusion detection, because statistical analysis can provide information about which measures contributed to an event being considered anomalous. Finding ways to get explanatory information out of neural networks is currently a research issue in the artificial intelligence community.

2.5 The Use of Expert Systems

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 "experts" in intrusion detection, namely 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 obtained only limited expertise because of the large amount of audit data produced and the tedium and length of time required to perform their checks. Thus, while automating these rules provides the useful function of freeing up the security officer to perform further analysis than they would otherwise have been capable of, such rules cannot be expected to be comprehensive. This approach would be more aptly called a security officer's assistant.

More comprehensive attempts to characterize intrusions are being made by several projects, including NIDES. 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. Thus, the use of a rule-based approach can fill some of the gaps in the statistical approach. With the rule-based approach, an event can trigger a rule apart from any consideration of whether the event is normal for the user. Thus, intrusion scenarios that may not be anomalous for the user (because the intruder has "trained" the system to see the behavior as normal) can be detected by appropriate rules.

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 user behavior changes over time, may be useful in detecting such attacks.

Anderson suggested also profiling the normal or expected behavior of *programs*. Such profiles could maintain statistics on, for example, what files are accessed, cpu time, elapsed time, and number of input and output characters, that are normally associated with the use of that program. Anderson also suggested profiling *files* and other objects [1].

Another real-time approach was taken by a group at SRI who measured certain characteristics, such as typing speed, of a user's keyboard activity — this approach has been called *keystroke dynamics*. Keystroke dynamics has been found to be a powerful means of continuously verifying the identity of the user doing the typing.

2.4 Neural Networks

Matching a user's observed behavior to a model of the user's past behavior is difficult, since user behavior can be very complex. False alarms can result from invalid assumptions about the distribution of the audit data made by the statistical algorithms. Missed detections can result from the inability to discriminate intrusive behavior from normal behavior on a purely statistical basis. The research group at SRI has experimented with the use of neural networks for intrusion detection to address the following problems.

- *The need for accurate statistical distributions.* Statistical methods sometimes depend on some assumptions about the underlying distributions of user behavior, such as Gaussian distribution of deviations from a norm. These assumptions may not be valid and can lead to a high false-alarm rate. Neural networks do not require such assumptions; a neural-network approach will have the effect of relaxing these assumptions on the data distribution.

- *Difficulty in evaluating detection measures.* In SRI's NIDES system, the set of intrusion-detection measures was selected on the basis of our research group's intuition and experience. It is difficult to know, for any postulated set of intrusion-detection measures, how effective these measures are for characterizing user behavior, both for users in general, and for any particular user. A measure may seem to be ineffective when considered for all users, but may be useful for some particular user. A neural network can serve as a tool to help us evaluate the effectiveness of various sets of measures.

MEASURE	DESCRIPTION
CPU Usage (ordinal)	CPU usage
I/O Usage (ordinal)	I/O usage
Location of Use (linear categorical)	# of connections from each location
Mailer Usage (linear categorical)	# of times each mailer was used
Editor Usage (linear categorical)	# of times each editor was used
Compiler Usage (linear categorical)	# of times each compiler was used
Shell Usage (linear categorical)	# of times each shell was invoked
Window Command Usage (linear categorical)	# of times each window command was used
Program Usage (linear categorical)	# of times each program was used
System Call Usage (linear categorical)	# of times each system call was used
Directory Usage (linear categorical)	# of times each directory was accessed
Directory Usage (binary categorical)	Whether a directory was accessed
Commands Used (ordinal)	# of different commands invoked
Directories Created (ordinal)	# of directories created
Directories Deleted (ordinal)	# of directories deleted
Directories Read (ordinal)	# of directories read/accessed
Directories Modified (ordinal)	# of directories modified
File Usage (linear categorical)	# of times each file was accessed
File Usage (binary categorical)	Whether a file was accessed
Temp File Usage (ordinal)	# of temporary files accessed
Files Created (ordinal)	# of files created
Files Deleted (ordinal)	# of files deleted
Files Read (ordinal)	# of files read/accessed
Files Modified (ordinal)	# of files modified
User IDs Accessed (linear categorical)	# of times user ID was changed
User IDs Accessed (binary categorical)	Whether another user ID was accessed
System Errors (ordinal)	# of system-related errors
System Errors by Type (linear categorical)	# of times each type of error occurred
Audit Record Activity (linear categorical)	# of audit records for each hour
Hourly Activity (binary categorical)	Whether an audit record was rec'd for each hour
Day of Use (linear categorical)	# of audit records for each day
Day of Use (binary categorical)	Whether audit records were received for each day
Remote Network Activity (ordinal)	Amt. of remote network activity
Network Activity by Type (linear categorical)	Amt. of network activity of each type
Network Activity by Hosts (linear categorical)	Amt. of network activity for each remote host
Local Network Activity (ordinal)	Amt. of network activity within the local system
Local Network Activity by Type (linear categorical)	Amt. of local network activity of each type
Local Network Activity by Hosts (linear categorical)	Amt. of local network activity for each host

Figure 1: User Measures

or attempted, could be detected by flagging departures from historically established norms of behavior for individual users [17, 15, 16, 18]. A survey of other intrusion-detection projects and prototypes can be found in [14].

SRI's NIDES prototype determines whether user behavior as reported in the audit data is normal with respect to past or acceptable behavior. Various intrusion-detection measures are profiled for each user. (A measure is an aspect of user behavior; a profile is a description of the expected behavior for a user with respect to a particular measure.) As NIDES observes the behavior of each monitored user, it keeps statistics for each user for each intrusion-detection measure. These statistics form a user's historical *profile*. The profiles are periodically updated based on observed user behavior. Thus, NIDES adaptively learns the behavior patterns of the users of the monitored system. As users alter their behavior, the thresholds maintained in the profiles will increase or decrease. Thus NIDES is potentially sensitive to abnormalities that human experts may not have considered.

Figure 1 shows the statistical measures currently implemented by NIDES. The measures fall into two main groups: ordinal and categorical. Categorical measures are further subclassified as linear or binary, defined as follows:

- An *ordinal measure* is a count of some numerically quantifiable aspect of observed behavior. For example, the amount of CPU time used and the number of audit records produced are ordinal measures.

- A *categorical measure* is a function of observed behavior over a finite set of categories. Its value is determined by its frequency relative to other categories.

- A *binary categorical measure* does not count the number of times that each category of behavior occurs, only whether the category was invoked (i.e., the category count is either 0 or 1). This type of measure is sensitive in detecting infrequently used categories, such as changing one's password.

- A *linear categorical measure* has a score function that counts the number of times each category of behavior occurs. For example, command usage is a linear categorical measure, where the categories span all the available command names for that system.

Different aspects of user behavior may be useful in discriminating between normal and abnormal computer use for different classes of users. 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 login 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.

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

- 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 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. He suggested that masqueraders can be detected by observing departures from established patterns of use for individual users. All of these approaches have been adopted by subsequent studies.

Anderson offered little hope for detecting clandestine users and the legitimate user who abuses his or her privileges. However, to detect a user abusing his or her privileges, it is possible that *a priori* rules for "socially acceptable" behavior could be established; this approach has been taken by a few studies. It is also possible that comparison with the norm established for the class of user to which the user belongs could detect abuse of privilege; this is one of the approaches being used by our research group at SRI.

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. SRI's intrusion detection prototype includes this approach.

2.3 Detecting Departures from Normal Activity

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. One such project at SRI used existing audit trails and studied possible approaches for building automated tools for their security analysis. Another such project considered building special security audit trails and studied possible approaches for their automated analysis [22]. These projects provided the first experimental evidence that users could be distinguished from one another based on their patterns of usage of the computer system, and that user behavior characteristics could be found that were capable of discriminating between normal user behavior and a variety of simulated intrusions.

Based on this early evidence, work was begun at SRI on a *real-time* intrusion-detection system, that is, a system that would continuously monitor user behavior and be capable of detecting suspicious behavior as it occurs. This system, known as NIDES (Next-Generation Intrusion-Detection Expert System), takes the approach that intrusions, whether successful

Although most computers in sensitive applications collect audit trails, these audit trails were generally established for performance measurement or accounting purposes and offer little help in detecting intrusions. Difficulties include the large quantity of audit information that is too detailed, voluminous, and often meaningless to a human reviewer. Moreover, single items of audit information may not in themselves be indicators of an attempted or successful intrusion. Also, such audit trails may omit information that is relevant to detecting intrusions. Nevertheless, even accounting audit trails provide information, such as who ran what program and when, what files were accessed, and how much memory and disk space was used, which is potentially useful for detecting intrusion attempts. To make audit trails useful for security purposes, automated tools are needed to analyze the audit data to assist in the detection of suspicious events.

The first part of this paper discusses a variety of techniques for automated analysis of computer system audit trails to detect suspicious user activity. The second part of this paper discusses system issues related to building and using intrusion detection systems. The third part of the paper describes an intrusion detection system being built at SRI.

2 Analysis Techniques for Intrusion Detection

2.1 Types of Audit Trail Analysis

The purpose of automated tools for the security analysis of computer system audit trails may be for audit data reduction; that is, to screen the data so as to drastically reduce the amount of audit data that a security officer must manually review. Or, automated tools may attempt to pinpoint actual intrusions or security violations during offline, after-the-fact, analysis. More ambitious tools attempt to detect intrusions and intrusion attempts in real time, as they occur. The following types of audit data analysis are relevant for security purposes:

- In-depth offline (after-the-fact) analysis of audit data.
- Real-time testing of audit data, so that an immediate protective response is possible.
- Subsequent analysis of the audit data for damage assessment.

Here we focus on the first two types of audit trail analysis. Cliff Stoll gives some ideas for using auditing in combination with other methods both for damage assessment and for tracking down and gathering evidence against a discovered intruder [21].

2.2 Detecting Different Types of Intrusions

The last several 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. The earliest work was a study by Jim Anderson [1]. Anderson categorized the threats that could be addressed by audit trail analysis as

- External penetrators (who are not authorized the use of the computer)

1993 Conference on Auditing and Computer Technology

Detecting Intruders in Computer Systems

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

Abstract

Although a computer system's primary defense is its access controls, computer system access controls 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. While many computer systems collect audit data, most do not have any capability for automated analysis of that data. Moreover, many systems collect large volumes of data that are not necessarily security relevant. To address the need for automated security analysis of audit trails, SRI is developing a real-time intrusion-detection expert system (NIDES). NIDES is an independent system that runs on its own workstation and processes audit data characterizing user activity received from a target system. NIDES provides 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 their privileges. NIDES detects masqueraders by keeping statistical profiles of past user behavior and raising an alarm when observed activity departs from established patterns of use for individual users. NIDES also includes expert-system rules that characterize certain types of intrusions. NIDES raises an alarm if observed activity matches any of its encoded intrusion scenarios.

1 Introduction

Timely detection of unauthorized intruders into computers and computer networks is a problem of increasing concern. Although a computer system's primary defense is its access controls, it is plain 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. Most computer systems have security susceptibilities that leave them vulnerable to attack and abuse. Finding and fixing all 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 then to provide evidence to establish the guilt or innocence of suspected individuals. Moreover, audit trails may be the only means of detecting authorized but abusive user activity.