Computers & Security, 12 (1993) 405-418

A survey of intrusion detection techniques

Teresa F. Lunt

Director, Secure Systems Research, Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA

Today's computer systems are vulnerable both to abuse by insiders and to penetration by outsiders, as evidenced by the growing number of incidents reported in the press. To close all security loopholes from today's systems is infeasible, and no combination of technologies can prevent legitimate users from abusing their authority in a system; thus auditing is viewed as the last line of defense.

Over the past several years, the computer security community has been developing automated tools to analyze computer system audit data for suspicious user behavior. This paper describes the use of such tools for detecting computer system intrusion and describes further technologies that may be of use for intrusion detection in the future.

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 used 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.

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 which program and when and 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 so as to assist in the detection of suspicious events.

0167-4048/93/$6.00

2. 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. Alternatively, 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 [1].

3. Detecting different types of intrusion

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 [2]. Anderson categorized the threats that could be addressed by audit trail analysis as:

- external penetrators (who are not authorized the use of the computer);
- internal penetrators (who are authorized use of the computer but are not authorized for the data, program, or resource accessed), including
  - masqueraders (who operate under another user's id and password),
  - clandestine users (who evade auditing and access controls);
- misfeasors (authorized users of the computer *and* resources accessed who misuse their privileges).

Anderson suggested that 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 such 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.

4. 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 [3]. These projects provided the first experimental evidence that users could be distinguished from one another on the basis of 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.

On the basis of 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 occurred. This system, known as IDES (Intrusion-Detection Expert System), takes the approach that intrusions, whether successful or attempted, can be detected by flagging departures from historically established norms of behavior for individual users [4–12]. A survey of other intrusion-detection projects and prototypes can be found in ref. 13.

SRI's IDES 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 IDES 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 on the basis of observed user behavior. Thus, IDES 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 IDES is potentially sensitive to abnormalities that human experts may not have considered.

Figure 1 shows the statistical measures currently implemented by IDES. 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 in which the categories span all the available command names for that system.

The prototype IDES has shown itself capable of detecting abnormal behavior in real time, as demonstrated by preliminary experiments. In ongoing work, SRI is providing additional functionality, including the ability to tune profile

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

Fig. 1. User measures.

parameters and to activate and deactivate measures on a user-by-user basis.

For systems like IDES, 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 9 pm but rarely after 9 or between 5 and 7); 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). 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 a previously established behavior pattern and who slowly varies this behavior over time, until a new behavior pattern has been established within which an attack can safely be mounted. 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 [2].

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.

5. 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. To address these concerns, the research group at SRI is experimenting with the use of neural networks for intrusion detection. Neural networks are being investigated 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 a 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 IDES 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.

• *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 IDES 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, according to our initial experiments, we do not feel that a neural network can simply replace IDES's statistical component. The most important reason for this is that IDES's statistical component provides 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.

6. 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 in the form of 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 the security officer to perform further analysis than he 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 IDES. 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.

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.

7. Model-based reasoning

Intruders often use specific, known procedures to breach a system's security. Examples include programmed password attacks, access to privileged files, or exploitation of known system vulnerabilities. Intruders might be characterized as "joy riders" with no malicious intent, as thieves aiming to appropriate resources of the computer system or those controlled by the system, or as terrorists aiming to destroy or incapacitate the system. With a model-based reasoning approach, one can develop specific models of proscribed activities. These models would imply certain activities with certain observables which could then be monitored. This would allow the intrusion-detection system to actively search for intruders by looking for activities which would be consistent with a hypothesized intrusion scenario. A determination of the likelihood of a hypothesized intrusion would be made on the basis of the combination of evidence for and against it. The intrusion scenarios are expected to vary for different types of intruder and for different systems. SRI is currently engaged in a research study aimed at evaluating the use of model-based reasoning for intrusion detection [14].

With the top-down model-based reasoning approach, the models of intrusion can be used to decide what specific data should be examined next. These models allow the system to predict the action an intruder would take who is following a particular scenario. This in turn allows the system to determine specifically which audit data to be concerned with. If the relevant data does not occur in the audit trail, then the scenario under consideration is probably not occurring. If the system does detect what it was looking for, then it predicts the next step and will then examine only data specifically relevant to confirming the hypothesis of the posited intrusion, and so on until a conclusion is reached. Thus, a model-based system reacts to the situation, using only that data most appropriate to the given situation and context.

Scenarios are specified in terms of the sequences of user behavior that constitute the scenario. For example, one scenario could represent a programmed password attack. This scenario would contain the steps needed to carry out the attack, expressed in terms of the specific user behavior involved (and not in terms of the audit data). The system would reason about hypothesized intrusions by gathering evidence from audit trails and statistical profiles in order to determine the likelihood that specific hypothesized intrusion scenarios are being enacted. For example, the system may hypothesize that user A is carrying out a programmed password attack, because user A was observed to have scanned the directory in which the password file resides. The system will seek additional evidence to confirm or refute this hypothesis.

When evidence has been discovered for the occurrence of a specific intrusion model, the system would seek additional evidence to confirm or refute the model. The system would then hypothesize the next step in the scenario that is expected to occur. Then this hypothesized behavior would be translated into the specific attributes and values of the audit data that would indicate that behavior. The system then would determine the specific audit data to examine next. The examined audit data would then be used to confirm or refute the hypothesized scenario.

For example, the hypothesized next step might be that user *A* will copy the password file. The system would translate this hypothesized behavior into the specific attributes and values of the audit data that would indicate that behavior. In other words, the system would figure out how the hypothesized behavior would show up in the audit data. For example, the hypothesis that user *A* will copy the password file might be translated into the following things to look for in the audit data: user *A* uses the 'copy' command, user *A* opens the password file, and user *A* writes a new file.

The system would then use this information, that is, the particular items in the audit trail that are indicative of the behavior in question, to develop a plan for the specific audit data to examine next. The system would compare the values in the plan to the actual values of the data observed, in an attempt to confirm or refute the hypothesized scenario.

This process would progress until enough evidence is obtained to put the likelihood for a particular intrusion scenario over some predetermined threshold. At this point, the system would announce that a potential intrusion has been detected.

The mapping of aspects of user behavior to reveal how the behavior will show up in the audit data must exhibit properties that differentiate the particular behavior of concern from everything else that might be occurring. These distinguishing properties must have the following characteristics.

- They must be easily recognized, so that they can be readily detected.

- They must be clearly associated with the behavior in question. These are called *criterial features*, because they always occur in the behavior you are looking for.

- They must not be associated with other "normal" behavior. These are called *distinguishing features*, because they generally do not occur in behavior that is considered normal.

Thus, in addition to the descriptions of how the intrusive behavior will show up in the audit data, there also must be included descriptions of other, or normal, behavior. However, normal behavior may be defined simply as anything other than the particular behavior the system is looking for. In this case, the models of intrusion must be specified so as to include only aspects of behavior not exhibited unless the intrusion scenario is being enacted.

The benefits of using model-based reasoning technology in intrusion detection applications are manyfold, including the following.

- Much more data can be processed, because the technology allows you to selectively narrow the focus of the relevant data. Thus, at any given time, only a small part of the data collected need be examined.

- More intuitive explanations of what is being detected can be generated, because the events flagged can be related to the defined intrusion scenarios.

- The system can predict what the intruder's next action will be, on the basis of the defined intrusion models. Such predictions can be used to verify an intrusion hypothesis, to take preventive action, or to determine which data to look for next.

If the stream of audit data contains a significant number of intrusions in comparison with the total volume of audit data (i.e., if there is a large signal-to-noise ratio), then an approach in which all the incoming data is examined and analyzed can be successful. However, if the number of intrusions is very small in comparison with the total volume of audit data (a small signal-to-noise ratio), then the amount of data to be examined can quickly overwhelm the intrusion-detection system. The system will be drawing very many conclusions, most of which will be dead ends. In this case, a more efficient approach would be to examine only the specific data in the audit data stream that are relevant at the moment. Thus we can, in effect, increase the signal-to-noise ratio in particular areas by looking only in those areas. This top-down approach to data analysis will be more efficient in the intrusion-detection domain, where the signal-to-noise ratio is extremely small. In contrast to this approach, an expert system's rules are always being used and evaluated against all the incoming audit data.

As is the case with expert systems, the approach is limited in that it looks for known intrusion scenarios, whereas the greatest threat may be unknown vulnerabilities and the attacks that have not yet been tried. Thus, a model-base reasoning approach will work best in combination with a statistical anomaly detection approach.

A model-based component in IDES could also make use of the information generated by the statistical component, because the statistical anomalies detected could be used as evidence by the model-based component. Moreover, the model-based component could be used to adaptively add or delete rules in the expert system rule base, as the situation requires.

Although an expert system can also be used to build models of intrusions, the model-based reasoning technology allows these models to be specified much more easily and directly. The technology allows one to specify intrusion scenarios, and then the intrusion-detection system can generate the specific rules needed for identifying supporting evidence for these scenarios from the audit data. The models can more accurately represent the undesirable behavior for which evidence is being looked for in the audit data. This is because the models can be expressed naturally in terms of the sequences of events that define the intrusion scenarios. By contrast, in an expert system, the rules are generally specified in the language of the audit data.

Model-based reasoning supports a sound theory for reasoning under uncertainty. This technology allows uncertainty in the rules—whether the behavior implies something illegitimate—and uncertainty in the significance of the data. Such a capability cannot easily be added to an expert system. And although some rule-based expert systems allow the handling of approximate information, they are based on an *ad hoc* theory, so that it is difficult to know what the results mean.

8. The IDES resolver

SRI is designing a *resolver* for use in IDES, which would review the conclusions of the two independent IDES components (the statistical and expert-system components) and provide further analysis as to the severity of the flagged events. Such a system might be able to make more complex deductions, thereby reducing the false-positive rate of anomaly reports and eliminating the possibility of the same suspicious behavior generating multiple alarms, that is, the same anomaly being reported by both the rule-based anomaly detector and the statistical anomaly detector. Other functions the resolver might perform include determining that an anomaly is extremely serious because both the expert system and the statistical component are complaining about it; deciding that unusual behavior is not worth worrying about because it does not have any security ramifications; deciding that a user who is on a trip may legitimately log in from some remote location instead of the normal location, and so on. The resolver will be

an intelligent system that can take information from the other detection components and make decisions based on this information.

The resolver could correlate audit data with other available data. In addition to the raw audit data itself, the following additional data would be helpful in distinguishing suspicious activity from normal activity:

- information about changes in user status, new users, terminated users, users on vacations, changed job assignments, user locations, and so forth;
- information about files, directories, devices, and authorizations.

9. Other approaches

Yet another approach was suggested that would define acceptable, as opposed to suspicious, behavior [15]. 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 [16]. 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.

10. 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 [17]. 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 US Internal Revenue Service) [2].

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.

11. 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. Alternatively, 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 intrusion 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 Anderson points out, users, particularly those with direct programming access, may operate at a level of control that bypasses the auditing and access controls [2]. In order to detect intruders operating at such a low level, auditing should be performed at the lowest level possible.

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 [18]. 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.

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 [19] 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 [21], being able to obtain a listing of all characters input and output on the affected communications line [1], or being able to view a single user's audit records for a session contiguously [2] 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 [21]. 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.

12. 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 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.

13. 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* [22]. 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 [23] 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 potentially 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 [24]. However, this same study found that workers were not opposed to computerized performance monitoring in principle, but that how it is used by management determines its 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.

14. 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. The strength of the second approach is the ease of stating exactly what behavior 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 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. And hand-in-hand with improved intrusion-detection capabilities, we can expect to see vastly improved auditing facilities optimized for security analysis.

References

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

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

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

[4] 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.

[5] T.F. Lunt, R. Jagannathan, R. Lee, S. Listgarten, D.L. Edwards, P.G. Neumann, H.S. Javitz and A. Valdes, *Development and Application of IDES: A Real-Time Intrusion-Detection Expert System*, Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, 1988.

[6] T.F. Lunt, R. Jagannathan, R. Lee, A. Whitehurst and S. Listgarten, Knowledge-based intrusion detection, in *Proceedings of the 1989 AI Systems in Government Conference*, March 1989.

[7] T.F. Lunt, Real-time intrusion detection, in *Proceedings of COMPCON Spring '89*, March 1989.

[8] T.F. Lunt, IDES: an intelligent system for detecting intruders, in *Proceedings of the Symposium: Computer Security, Threat and Countermeasures, Rome, Italy*, November 1990.

[9] T.F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. 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.

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

[11] 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.

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

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

[14] T.D. Garvey and T.F. Lunt, Model-based intrusion detection, in *Proceedings of the 14th National Computer Security Conference*, October 1991.

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

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

[17] 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.

[18] 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.

[19] 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.

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

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

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

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

Teresa F. Lunt is Program Director for Computer Security in SRI's Computer Science Laboratory. She is leading two landmark programs: the SeaView multilevel secure relational database system and the IDES intrusion-detection system. She is also leading new research areas in security for real-time systems, inference control in multilevel database systems, security for knowledge-based systems, and using AI techniques for computer security. She won an SRI Exceptional Achievement Award in 1988 and an SRI Team-Building Award in 1990.

Prior to joining SRI in early 1986, Lunt worked at the MITRE Corporation for four years and later at SYTEK's Data Security Division for two years. She has worked on audit trail analysis, automated security guards, security models, and formal verification of secure systems. She received the AB degree from Princeton University in 1976 and the MA degree in applied mathematics from Indiana University, Bloomington, in 1979.

She has published over 40 conference and journal papers and over 20 technical reports in the area of computer security. She won the Outstanding Paper Award at the 11th National Computer Security Conference in 1988 and the Best Paper Award at the 1987 IEEE Symposium on Security and Privacy. She is editor of the Springer Verlag book *Research Directions in Database Security*. She was founding editor, publisher, and principal contributor to the *Data Security Letter*. She is a member of IEEE, the IEEE Computer Society, the Technical Committee on Security and Privacy of the Computer Society of the IEEE, and ACM. She is program co-chair for the 1990 and 1991 IEEE Symposium on Security and Privacy, Vice Chair for the 1992 Symposium and General Chair for the 1993 Symposium. She has been on the program committee of every major computer security conference for many years, including the IEEE Symposium on Security and Privacy and the National Computer Security Conference. Teresa Lunt founded, and is the principal organizer of two series of technical workshops: the semi-annual SRI intrusion detection workshops, and the annual Rome Laboratory database security workshops. She recently chaired a workshop on object-oriented system security for the European Institute for System Security. She served as the general chair for the 3rd IFIP WG 11.3 Workshop on Database Security. She founded the Bay Area Trusted Systems Symposium, which meets three times yearly at member companies in the San Francisco Bay Area.