LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

1

LA-UR -87-4103

CONF-880456--1

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA--UR--87-4103

DE88 004310

TITLE MONITORING THE USAGE OF A COMPUTER SYSTEM

AUTHOR(S) D. J. Bailey, QS-4

MASTER

SUBMITTED TO IEEE Symp., Security and Privacy
April 1988
Oakland, CA

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO. 836 R4
ST. NO. 2629 5/81

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MONITORING THE USAGE OF A COMPUTER SYSTEM

by

David J. Bailey

Los Alamos National Laboratory
OS-4, Mail Stop G727
P. O. Box 1663
Los Alamos, NM 87545

Monitoring the Usage of a Computer System

David Bailey
Los Alamos National Laboratory

November 1987

Abstract

Controlling the usage of computer systems, particularly those operated for the federal government, is an important topic today. Audit requirements have grown to the point where they can be a significant burden to the proprietors of the system. The paper briefly mentions several proposals for responding to increased audit requirements and for monitoring a system to detect unauthorized activity. A technique is proposed for situations where the proscribed or the intended activity can be characterized in terms of program or system performance parameters. The design of a usage monitoring system is outlined. The design is based on enhancing the audit data provided by the monitored system, capturing the audit data in a separate system to protect it from user access, and implementing one of the audit trail analysis systems currently under development.

If I kiss you, that is a psychological interaction.
On the other hand, if I hit you over the head with a brick, that is also a psychological interaction.
The difference is that one is friendly and the other is not so friendly.
The crucial point is if you can tell which is which.

FORTUNE

Introduction

Controlling the usage of computer systems, particularly those operated for the federal government, is an important topic today. Agencies of the government [DOE 87,DOD 87] are pushing their own computing organizations and those of their contractors to implement audit trailing mechanisms that provide more and more detail. Security officers are being directed to process the resulting larger volumes of audit data within already constrained budgets. They are also being directed to implement additional, more stringent controls on system usage. These legitimate control requirements imposed by the system's sponsors force the security officer to devote a larger fraction of his or her resources to the audit function. This leaves fewer resources for designing and implementing more effective security controls, and makes the activities of the security officer increasingly reactive.

Audit requirements have now grown to the point where they can be a significant burden to the proprietors of the system. For example, a prototype implementation project at the Mitre Corporation [PICC 87], measured normal audit trail volume from a secure workstation at approximately 7 Mbytes per user per day. In the worst case (a heavily used workstation), this rose to almost 136 Mbytes per user per day. There is no reason to suspect that these figures are unusually high. Making effective use of this volume of audit information would be difficult if only one workstation were involved. At a site with hundreds or thousands of workstations, effective use of this volume of data is infeasible without automated tools and substantial computing power.

Auditing has also become a subject of interest in the research community. Anderson [ANDE 80] has characterized to problem of monitoring and surveillance of computer systems, and a number of tools have been proposed to assist the security officer in dealing with the increased volume of audit trail data. These have included expert systems for reading and digesting audit trails [BROW 84,TRW 85], and more elaborate systems [DENN 86,LUNT 85] intended to detect intrusions by outsiders or deviations from established patterns by insiders. The idea underlying these designs is that a user's deviation from established patterns may foreshadow the beginning of unauthorized activities that should be investigated, watched, or stopped. A deviation from established patterns might also signify the presence of a masquerader, a person who has gained access using someone else's identification and authentication data.

One interesting response [CLYD 87] to the need for better control of

system usage is the proposal to collect, analyze, and preserve everything all input keystrokes, all system calls, and all bytes transmitted on I/O channels. Mr. Clyde reasons that if all bytes transmitted to or from the user are collected, there will be a record of any infraction that a user might commit, or at least of the actions the user took to initiate the infraction. The crucial point, of course, is whether you can recognize the unfriendly bytes when you see them. Clyde's intention would be to provide expert tools to read this mass of material to detect deviations from the user's normal patterns.

Karger [KARG 87] presents an interesting and somewhat different approach. In a paper directed toward countering a specific threat (discretionary trojan horses), but adaptable to the more general situation of monitoring and controlling the partially authorized insider, he suggests advance analysis of the operating requirements of specific programs. The shell or command language interpreter used to execute the program then limits the program to specific actions, files, or directories, as deemed appropriate based on the prior analysis. If the program attempts to operate outside of the established limits, the shell interrupts execution and asks the user what to do. An analog of this process could be used to control the actions of individual users. A usage profile would be created when each user account is established. Attempts to operate outside this profile would occasion additional action by the shell. The additional action might be increased auditing, an alarm to the security officer, or prevention of the action.

Intrusion Detection

An *intrusion* in a system is an attempt by any person to perform actions for which the person is not authorized. This intentionally broad definition includes attempts to access the system by "outsiders" who can be thought of as users who are not authorized to perform *any* actions. It includes activities by "insiders" who abuse their ability to access the system to perform actions for which they are not authorized. It also includes activities by insiders that appear the same (in the audit record) as authorized activities. Using the word processor to write personal letters might fall into this category. The term *Unauthorized Activity* refers to any action by a user that is outside the scope of activities sanctioned by the system proprietors. It might range from attempts to penetrate the system to simply misusing programs that have been provided with the system. Unauthorized activity does not necessarily imply bypassing or spoofing the system's security controls.

As the examples in the previous paragraph suggest, detecting unauthorized activity may be very difficult. Outsiders who succeed in getting on the system use accounts (at least the first time that they succeed) that already exist. That means they look like insiders, and they must do something very unusual to be spotted as intruders. Intruders who intend system penetration must necessarily take actions that are very unusual. If they are skilled their actions will not result in any unusual audit trails. For example, they may be skilled enough to "turn off" auditing, to cover their tracks in the audit trail, or to operate at a level far enough below the user interface of the operating system where there is likely to be little or no auditing. Finally, intruders who are overstepping or misusing their authority might not cause anything strange to appear in their audit trail. For example, if a person is assigned work that involves use of a spread-sheet calculator, and subsequently uses the same program to manage their own personal investments, there may be nothing unusual in the audit trail.

The proposals mentioned above, with the exception of the Karger and Clyde proposals, all follow basically the same procedure. The system audit trails are collected and analyzed to determine whether the activity shown is "normal." The proposals differ in how they determine what constitutes normal activity. The expert system proposals generally find an expert to describe what abnormal activity might look like and then alarm any time a pattern of activity fits the description. The difficulty with this approach is that patterns not thought of by the expert will not be detected.

The SRI Intrusion Detection Expert System attempts to be more uniform in its treatment by adopting a heuristic approach. The program becomes its own expert by constructing an activity profile of each system user. It then examines subsequent audit trails to find deviations from the previously constructed user profiles. This will successfully detect a large class of intrusions, particularly those involving abuses where the only indicator may be a change in work habits. For example, a person who starts managing investments or running a personal business may do it at night to avoid encounters with the boss. However, if the intrusive activity is already occurring when the auditing system is turned on, or if the deviations from normal activity are small, the system will not find it.

The Clyde proposal differs from the others only in that it does not rely on the normal system audit trails. Instead, the Clyde package collects its own (much larger) set of data and performs the search for unusual activity on that data. The collected data consists of all traffic over I/O channels plus system calls made by user processes. This author's experience with

system penetrations would suggest that the activities necessary to penetrate a system are somewhat more likely to appear in this data collection. Skilled penetrators, in particular, might differ somewhat from normal system users, in some systems, in the system calls that their programs use. However, in general, there is no reason to suppose that, even if all the system calls are examined, they would be recognized as unusual.

The common factor in all of the approaches described is that they depend on being able to tell the difference between normal and abnormal activity. Almost all of them try to do this by looking at the standard system audit trail. The audit trail produced by most operating systems is a record of events designed to facilitate customer billing and performance analysis. Many items are also included to allow users to diagnose the reasons for failure of specific jobs. In most systems, no information is specifically included to facilitate detection of intrusions.

Audit trails used for security have two very different purposes with differing information requirements. The first purpose is to detect intrusions either after the fact or while they are under way. Today the detection us usually performed by a security officer reading the raw audit trail or a reduced version of it the next day. Success requires the ability to make the distinction between legitimate and illegitimate events. Neither the system audit trail nor the data collected under the Clyde proposal ordinarily contains enough information to make this distinction in an automated way. The second purpose of security audit trails is to facilitate a damage assessment after an intrusion has been discovered. The normal audit trail provided by most operating systems is far more useful for this purpose than it is for detecting intrusions. Even if a system provides sufficient audit data to detect intrusions, it will still be necessary to collect the standard audit data to support damage assessment.

A Specific Problem

Any of the systems described above could be made more effective by enhancing or replacing the system audit trail with more useful data. A specific problem will be described below that may be solvable by carefully choosing the data to be audited. However, even situations that do not satisfy the constraints of this special case can be improved. Careful examination of each specific operating system will result in a small set of system parameters that give a much better indication of possible intrusion than the

performance, billing, and damage assessment data normally collected.

For example, many early penetration scenarios for Digital Equipment VMS [1] systems depended on using various techniques to acquire privileges that were not granted when the user signed on. Most of these intrusions would be detected (not necessarily in real time) by auditing changes to a process's effective and permanent privilege masks. Unfortunately, these changes do not produce audit records. For an example in a completely different system, consider CDC's NOS [2] operating system. In NOS, many parameters in a job's control point area, such as the reference address and field length, should never change during any of the job's execution intervals. If the operating system were modified to create an audit record any time these parameters changed there would be a greater chance of detecting intrusions in real time. Similarly, system specific analysis of each operating system will yield addresses, events, types of access, or process or system flags that should be audited. Some items will enhance the ability of any audit analysis mechanism to detect unusual behavior, and some may even serve as real-time indicators that a penetration is in progress.

This paper was originally motivated by trying to solve a specific problem that arises in export control. The problem is to allow installation of a computer system at a specific site with the condition that it not be used to solve problems in a given problem class. For the purposes of the problem under discussion, it can be assumed that the site is not interested in running programs within the proscribed class, but some of the users might be interested. Thus, we can assume that the site management will not try to subvert the protection system. This means that the physical security provided by the site can be used to protect the monitoring system from physical attack and that additional physical security is not needed. If, however, the site management is not trusted, additional problems of physical security, integrity of the audit mechanism, and communications are introduced, which are outside of the scope of this discussion.

With minor variations, this problem description also fits many commercial and government situations. Any computer system dedicated to one or a small number of applications could be monitored to detect whether those

[1] VMS is a Trademark of Digital Equipment Corporation. Reference to VMS here is not intended as a disparagement of the system in any way. VMS includes many security features and has been rated C2 by the National Computer Security Center [NCSC 86].

[2] NOS is a trademark of Control Data Corporation. NOS, like VMS, includes many security features and has also received a C2 rating from the National Computer Security Center [NCSC 86].

applications or something else is running.

With the definition of "intrusion" given in the previous section, the problem is to detect an intrusion by a system insider: a user who appears to be legitimate but who has gone beyond the authorization granted to him or her for use of the system. How to respond to a detected intrusion is not a part of the problem considered here. The response policy must be devised by the proprietors of the specific system and may vary from time to time. Possible responses might include manual intervention by the security officer, terminating the offending process immediately, or turning on additional auditing capabilities for intelligence collection.

The assumption that the site management can be trusted is valid in many cases in export control. In particular, it is valid where a system is to be delivered in a friendly country and export restrictions can be written that allow for periodic inspections. This assumption allows for several simplifications in implementation. It is possible to choose mechanisms that make use of physical security provided by the site and to ignore the fact that the mechanism could be defeated given physical access to it. Thus, we assume that physical access to the mechanism by the intruder is not a problem and we need only design the mechanism to prevent access using the monitored system. For the same reason, the capability of the system to respond can be limited to local reporting of a suspected intrusion event, the mechanism need not be able to implement the entire response policy. Local notification is sufficient if the security staff is able to complete the response.

Usage Monitoring

Our approach to solving the problem posed above is to monitor the activity on the machine by observing critical system parameters. If a pattern of activity, detected as described below, appears that is characteristic of the proscribed class of problems, an alarm is raised for response by the security officer. As an alternative to declaring that a system must not process problems in a particular class, it would be possible to declare that all problems processed must fall within a specified class. In the first case, an alarm would be raised if processing was ever detected within the specified class. In the second, the alarm would occur if processing ever left the specified class. In both cases, the mechanism is basically the same.

The usage monitoring approach is based on the idea that system parameters such as volume of input, volume of output, paging rate, working set

size, system calls used, mean time between system calls, and average program counter value can be used to distinguish between different programs. While this assertion has not been experimentally verified, and that should be carefully done, the phenomenon is well known to people who remember computers with lights on the console. Everyone has heard stories of operators who were so familiar with their particular computer system that they could tell what job was running and what stage it was in by the way the lights flashed and by the sounds the machine made. While modern computers no longer have flashing lights, it is asserted that particular programs can still be detected by their differences in behavior as they run.

The idea can be carried one step further to assert that programs with similar functions will have operating characteristics that are similar. This is reasonable since programs that have reached the robustness and stability required for production use tend to be continuous systems: small modifications to the program tend to produce small modifications in behavior. Thus, two programs that solve hydrodynamics problems will have similar operating characteristics and would both be identifiable as a "hydro code" and not mistaken for the accounting program. Each such program has a "kernel" that does the essential computation and a housekeeping portion that sets up and controls the problem. Specifically, the assertion is that the kernel will be distinguishable from the housekeeping code by its behavior, and that kernels for different types of programs can be distinguished from each other.

We propose construction of a Machine Usage Monitoring System (MUMS) in three separate steps. First, an audit capture and analysis system is constructed and installed. Most of the techniques mentioned in the Introduction could be employed. Some security requirements for the problem of interest will be discussed in the next section.

Second, experiments need to be run on the monitored system or a similar operating system where problems in the proscribed (or demanded) class are being run. The experimental operating system has to be instrumented to report internal system parameters of potential interest. After a number of trials, a statistical analysis is performed to identify the system parameters best able to characterize the problem type to be identified.

Third, once the most useful system parameters have been identified, it is necessary to modify the host operating system so that these parameters are reported to the audit capture mechanism. In practice, the system would be modified to report many more parameters than those initially selected. There are several reasons for this. The selected set might not be exactly the best choice for the operating system being audited, and enough flexibility

should be retained to change the parameters used if that proves advisable. Second, it will almost certainly be desirable to add additional program profiles using different parameters at some later date. And finally, collection of more data than are actually used will make it more difficult for an intruder to defeat the system by controlling the performance parameters of the intrusive job.

As previously indicated, each operating system will have system or job data that should remain unchanged for definable periods of time. Changes to some of these critical data items might be direct indicators of an intrusion in progress. The operating system should be modified to audit these critical parameters as well.

Finally, the auditing granularity need not be the entire machine. It may not be acceptable merely to detect that the machine is doing something outside the accepted list. It may often be necessary to discriminate acceptable from unacceptable program behavior on a per user or per port basis. Karger's idea involved making the user shell a participant in responses to unusual activity. One of the simplest ways to do this is to modify the shell to report "interesting" information to the audit mechanism. The shell can then become a participant by reporting information about its own identity and activities to the audit mechanism. The shell would also have to be modified so that it could respond to a few simple requests made over an interprocess communication link from the audit mechanism to the shell.

Audit Capture

The basic security requirement for the audit capture mechanism is that is must collect all relevant audit messages issued by the monitored system and not allow direct access to the records to any process operating on the audited machine. The capture of audit records can be accomplished in several ways. The simplest and most common way is for a system process that creates audit messages to write them directly to a file or a predefined section to disk kept on the system. Records created in this way may be subject to modification by a clever intruder, and it may be possible for the intruder to corrupt or disable the mechanism that creates the records.

A second method is to write the audit records to an autonomous system via a network connection of some sort. For example, the audit machine might be connected to a terminal or network port on the monitored machine. If the audit capture accepts only text or predefined messages from the monitored

system, it should be possible to protect the audit data from destruction or modification by users operating on the monitored system. This method is still susceptible to being turned off by a clever intruder. In the author's opinion, this is the minimum that must be done to provide a secure audit mechanism.

A third possibility is to create a monitoring system that has direct hardware connections to the monitored system. This would allow the monitoring system to perform the auditing function directly and would not be susceptible to disruption by an intruder. This method, however desirable, requires modification to the monitored system, and may, therefore, be too complicated to implement in most cases.

We envision implementing the MUMS on a separate machine as in the second method above. Disabling the audit mechanism is a potential problem, but the MUMS would be designed to alarm if a continuing flow of audit messages were not received. This doesn't solve the problem but does require the intruder to be more sophisticated than to simply disable the audit mechanism. Because of the assumption that the system proprietors can be trusted, physical security beyond that provided for the monitored system is not needed.

In addition to collecting audit data and analyzing it as it arrives, the MUMS would provide several other facilities. For example, it will be necessary to install, modify, and delete job or system profiles. It will be necessary to be able to specify the response policy. Different detected incidents would engender different responses. The capability to create new program profiles would allow the system to grow with changing requirements in the monitored system.

Conclusion

There is an increased interest today in controlling the usage of computer systems. The interest has been extended from controlling who may use a system to include controlling what they do with the system. The simple response to this interest is to collect more audit data and to direct the security officer to spend more time analyzing it. However, this response is sufficiently burdensome that it may result in reduced security rather than enhanced security because effort must be diverted from other areas.

Three proposals have been discussed that can make auditing a more effective tool in some situations.

- Modify the operating system to audit internal state and performance information selected for its ability to discriminate between authorized and unauthorized activities on the system.
- Select the operating system status data by creating behavioral profiles of authorized activities. Build the audit mechanism so that it is able to profile jobs as they run and compare their profiles with those previously constructed for authorized activities.
- Modify the user shell program or command language interpreter to accept messages from the audit mechanism so that it can become an active part of the incident response mechanism.

Together these proposals could lead to a more automated, more effective audit mechanism without forcing the security officer to allocate all available effort to reading audit trails.

References

[ANDE 80] J. P. Anderson, *Computer Security Threat monitoring and Surveillance*, James P. Anderson Company, April 1980.

[BROW 84] C. L. Browne, *DOITYE: An Interlisp Program for the Analysis of the Network Security Controller Files (U)*, LA-9920-MS, Los Alamos National Laboratory, January 1984, Confidential.

[CLYD 87] A. R. Clyde, *A Surveillance Gate Model for Automated Information Security and Insider Threat Identification on Sensitive Computer Systems*, pre-print, February 1987.

[DENN 86] D. E. Denning, *An Intrusion Detection Model*, Proceedings of the 1986 IEEE Symposium on Security and Privacy, April 1986.

[DOD 87] Department of Defense, *DoD 5200.28*, 1987. *Need more complete reference here.*

[DOE 87] Department of Energy, *Order 5636.2A*, *Classified Computer Security Program*, draft, August 1987.

[KARG 87] P. A. Karger, *Limiting the Damage Potential of Discretionary Trojan Horses*, Proceedings of the 1987 IEEE Symposium on Security and Privacy, April 1987.

[LUNT 85] T. Lunt, ?????, SYTEK, Inc. *Need more complete reference here.*

[NCSC 86] *Evaluated Products List for Trusted Computer Systems*, National Computer Security Center, 31 December 1986.

[PICC 87] J. Picciotto, *The Design of an Effective Auditing Subsystem*, Proceedings of the 1987 IEEE Symposium on Security and Privacy, April 1987.

[TRW 85] TRW proposal and report on intrusion detection. *Need more complete reference here.*