ENHANCING COMPUTER SYSTEM SECURITY

Dennis Hollingworth

August 1973

P-5064

The Rand Paper Series

Papers are issued by The Rand Corporation as a service to its professional staff. Their purpose is to facilitate the exchange of ideas among those who share the author's research interests; Papers are not reports prepared in fulfillment of Rand's contracts or grants. Views expressed in a Paper are the author's own, and are not necessarily shared by Rand or its research sponsors.

The Rand Corporation
Santa Monica, California 90406

ENHANCING COMPUTER SYSTEM SECURITY

Dennis Hollingworth

The Rand Corporation
Santa Monica, California

ABSTRACT

One of the more significant problems in the information processing industry today is that of protecting sensitive computerized information from unauthorized access. As yet, no technique suggested or implemented appears to offer a complete solution to the problem. An attractive and cost-effective partial solution may reside in a different approach to the problem — implementation of an entrapment strategy — where the nature of the system/penetrator interaction is altered via introduction of counter-penetration elements into the system hardware and software.

THE SECURITY PROBLEM

A significant problem in the information processing industry today is that of protecting sensitive computerized information from unauthorized access. The problem has developed as a natural consequence of the rapid increase in complexity and sophistication of computer operating system software, coupled with the desire to take maximum advantage of the capabilities of the electronic computer in the processing of larger and larger volumes of information.

With the single-user, second-generation systems of the late 50's and early 60's, sensitive computerized information could be effectively and economically protected by what essentially amounted to administrative and procedural controls. Controls such as insuring the integrity of the physical site, erasing residual information left in memory after a job had completed processing, and maintaining information in physically separated files provided protection commensurate with existing operating system technology.

With the development of larger, faster computer hardware and supporting time-sharing and multi-programming software systems, there was a concomitant development of new security requirements. The economics of multi-user software systems dictates that the central processing facility be available to a number of users at the same time — that a number of user programs and data be simultaneously co-resident in the system. Thus, each user's programs and data are directly exposed to access by other users, subject to the protection provided by the system's security

protection mechanisms. [1][2]

Linking computers into computer networks causes further complication: the potential user population is greatly expanded and potentially more hostile with the inclusion of users from other installations. In the non-network environment some degree of administrative control may be exercised over the user population with respect to who has access to the computer, when he has access, what level, etc. In the network environment, the effectiveness of such controls is reduced through the dependence of the host installation on the administrative and operational controls of the other network nodes. Administrative controls at the host installation become only as effective as those of the other network nodes which have access to the host computer. Weaknesses in the security controls of the host installation are magnified. Sensitive information which may have been implicitly protected by the threat of punitive action no longer receives such protection in the network environment and must be protected by other means. [3]

As a result of these and other developments, a significant requirement exists for effective hardware and software security protection mechanisms to insure the integrity of user programs and data in the multi-user software environment — a requirement not satisfied by existing operating system technology. This raises two separate but related issues: what to do with the many ill-structured systems which presently exist in which there is already a significant investment in both hardware and software, and how to design adequate data security into future systems such that data security is no longer a dominating concern.

EXISTING SYSTEMS

In general, the security protection provided by vendor-supplied hardware and software has not been adequate. While most vendor-supplied operating systems appear to provide reasonable program and file protection from inadvertent user errors and the penetration efforts of less knowledgeable system users, they provide little or no protection from the machinations of sophisticated users to circumvent the system's security protection mechanisms. Vendors seem to have been motivated almost entirely by considerations of system reliability with little attention given to the problem of deliberate subversion of the system's security protection mechanisms. As the requirement for information security has become more pressing and vendor protection mechanisms have been demonstrated to be inadequate, two strategies for providing the necessary level of security have been employed, neither of which has proven satisfactory.

Separate Processing Environments

Faced with the problem of protecting sensitive computerized information from unauthorized user access, some installations have reverted to operating in a more restrictive processing environment, electing to separate sensitive and non-sensitive processing. This has been accomplished by either dedicating special hours of the day to the processing of sensitive programs and data, or procuring separate systems for processing each type

of information. Such approaches have obvious and significant disadvantages.

With separate hours reserved for the processing of sensitive information, the user trying to process such information necessarily suffers the penalty of restricted access to the computer. This processing burden sometimes results in sensitive material being processed with an inappropriate degree of security. Worse, it may encourage users to employ inefficient manual techniques or pre-computer data processing methods to get their work done. Furthermore, changing between processing modes results in inefficiencies of operation and subsequent loss of computational resources.

Use of separate computers is equally burdensome. Separate configurations for each mode of processing requires a costly duplication of hardware, software, and human resources. Two systems have to be maintained; software has to be developed and/or purchased for both; peripherals must be duplicated for each system; duplicate files and libraries must reside in both.

Neither of these approaches is particularly cost effective; both inevitably result in undue difficulty in the processing of sensitive information.

Security Retrofit Efforts

The unpalatability of separate environments for sensitive and non-sensitive processing has motivated efforts to retrofit security into existing operating systems. In some cases, these

have involved major efforts to repair and enhance the security protection features of the vendor-supplied software to produce a system which is impervious to system penetration. In others, the magnitude of the required correction effort has prompted the installation to restrict such activities to the securing of a significant subsystem of the operating system while leaving the host system intact.

Such efforts have typically proven to be inadequate protection against serious penetration activity. [4] The hardened systems have been exposed to formal penetration attempts by special teams — so called "tiger teams" — to determine the added security afforded by the system hardening process. The results have been generally unsatisfactory: the penetration teams have almost invariably found ways to circumvent the hardened system's security protection mechanisms. During one recent exercise, a Rand penetration team was able to:

1. List the contents of a security protected pseudo-classified file at a Rand remote job entry station.

2. Circumvent the standard file access protection mechanisms and process security protected user files directly.

3. List the security protected PASSWORD file at the remote terminal.

4. Insert a permanent software plant in the operating system to facilitate future circumvention of security protection mechanisms.

5. Accomplish the above undetected by the target installation.

The principal result of such penetration experiments has been an increased awareness of the profusion of design and implementation errors in vendor-supplied software, and a general disenchantment with the prospect of successfully retrofitting data security into existing computer systems. The problem has proven to entail more than just diagnosing and correcting the flaws in an existing operating system; it also entails ascertaining the security afforded by the final product. The inability to assess the security of information in the retrofitted system has resulted in a negative attitude towards the securing of present systems and a reliance upon the development of future, demonstrably secure systems for solution of the problem.

FUTURE SYSTEMS

A position presently being articulated is that current and future efforts in the design and development of new hardware and software will yield systems which will be demonstrably secure through application of software validation and program proving technologies. There is some promise, for example, of the

possibility of isolating all elements of the access control mechanism and those portions of the operating system which must run privileged in a well-defined software entity, a security kernel, small enough to be proven secure through application of program proving techniques. Whether or not this can actually be accomplished is uncertain; but until there is some such advance in the design and implementation of operating systems, new systems will be released which may evoke greater confidence in the security protection afforded by the hardware and software, but which are still ambiguous as to the existence of penetrable system flaws. Furthermore, even if systems should be produced in which the system design can be proven secure, problems still exist with regard to the failure of implementation strategies or administrative controls governing alterations to the original product which might invalidate the proof process. Such failures might be all the more damaging as a result of a misplaced confidence in the security of the system resulting from its design having been proven correct. Thus, even in the case of new systems, supportive techniques would seem to be called for.

AN ENTRAPMENT STRATEGY

The principal requirement for a secure operating system is not that the system be free of penetrable system flaws, but that such flaws are never utilized to effect system penetration, i.e., a system is developed with defenses which effectively mask any

residual system flaws such that penetration , while possible, is never attempted through an exploitable system error [5]. A tactic one might employ to establish such a condition is to introduce diversionary mechanisms in the system which induce the system penetrator to attack the strongest points of the system instead of the weakest while, at the same time, announcing his presence to system's personnel.

One problem with the current form of security protection mechanisms is that they are in some sense passive elements of the system. The penetrator is essentially free to gather whatever information he can about the defenses of the system while running little risk of exposure. He can diagnose and attempt to exploit errors in a mode in which the only consequences of failure are often program termination or notification of a failure condition. Worse, from the standpoint of defending against system penetration, such systems are often designed to supply to the user information describing the failure condition and, in some cases, the proper technique for invoking system services. Such information facilitates the penetration effort. Security in such systems tends to be more of a zero or one phenomenon — that is, it must be assumed that the penetrator will bring all the resources he has available to bear on the task finding a penetrable system flaw should the value of the information warrant the effort. Further, it must be assumed that he can find such a flaw should an exploitable error remain, since there is little to deter his effort.

The objective of an entrapment strategy is to alter this classical mode of penetrator/system interaction through the

introduction of counter-penetration elements into the system hardware and software. The nature of these elements would depend to a large extent on the degree of security desired in a system and the nature of the standard hardware and software. The introduction of such mechanisms alters the zero/one nature of security protection in standard systems, permitting one to define levels of confidence in the security of such systems. These confidence levels would be based upon the projected upper limit to the number of errors remaining in the system and the number of entrapment mechanisms inserted in the system software and hardware. Despite the level of resources brought to bear in the penetration attack, there would now be some active defenses existing in the system which would serve as counter-penetration elements, introducing a measurable probability that the penetrator would be detected in his activities.

Implementation of an entrapment strategy involves a two-phase effort:

- o Preparatory system conditioning,
- o Introduction of entrapment mechanisms

PREPARATORY SYSTEM CONDITIONING

The primary objective of preparatory system conditioning is to produce a secure enough version of the target system such that the penetrator must necessarily carry his search for exploitable

errors to the target system itself , and thereby be exposed to entrapment mechanisms. Thus, the level of effort required depends to a large extent on the characteristics of the system being secured and the degree of security desired. The standard vendor products currently being marketed would likely require significant effort to diagnose and correct the majority of exploitable system flaws before the system were ready for the introduction of entrapment mechanisms. Such an effort might well involve several iterations of system penetration testing/hardening to eliminate the majority of penetrable errors. With systems developed with data security as a fundamental design criterion (e.g. MULTICS [6]) such requirements would be lessened, but not eliminated.

INTRODUCTION OF ENTRAPMENT MECHANISMS

Entrapment mechanisms are special hardware and software extensions to the standard system designed to detect and trap the potential system penetrator during the penetration activity. They consist of:

- Pseudo-flaws
- Entrapment modules

Pseudo-flaws

Pseudo-flaws are elements of the internal system defense camouflaged as external system implementation errors. Depending on their function and implementation, they may require software

and/or firmware modifications to the central system. Their primary function is to divert the system penetrator from searching for the more obscure system errors which may still exist in the system by providing deceptively convenient vehicles for system penetration.

Pseudo-flaws may be introduced at several levels of user/system interaction. For higher level service requests, the user interacts with command processors, interpreters, file editors, utility packages, etc; he requests general system services such as copying a file, scheduling a job, or editing a program or data. Such interaction does not require specific knowledge of the internal characteristics of the system. Thus, pseudo-flaws introduced at this level may be implemented via software modifications to the system with little likelihood of detection by the system penetrator. Consider the following examples:

- A pseudo-flaw inserted in the LOGON processor ostensibly permits unauthorized users to log on to the system undetected. After several instances of entry of an erroneous user-ID or password, the information in error is accepted by the system. The user, however, is tagged by the LOGON processor as a potential system penetrator, an alarm condition is raised, and entrapment modules are invoked to intervene in subsequent user/system interaction.

- Pseudo-flaws are inserted in standard system utility functions such that illicit access to security protected user or system files appears possible. Upon detection of an unauthorized attempt to access a security protected file, the system records the attempt, and then creates linkages to a specially prepared dummy file containing erroneous information. Subsequent file access requests are mapped onto the dummy file, and the user's activities are, thereafter, monitored and evaluated.

Pseudo-flaws might be introduced into the system substructure to thwart the more sophisticated penetration effort. For basic system service requests, the user interacts with the operating system nucleus, requesting services via supervisor call instructions. The more knowledgeable system penetrator operates at this same level of interaction, exploiting system design and implementation flaws to directly circumvent the system's security protection mechanisms; it is at a comparable level that such activity must be combated. Significantly greater opportunities exist in this area to develop innovative entrapment mechanisms for processors with micro-programmed instruction sets than for processors with hard-wired instruction sets, but there are still undetectable techniques which may be exploited for the latter type of machine (such as exploiting the address compare stop switches). Consider the form such pseudo-flaws might take specific to the IBM System/360 or System/370:

- Selected system service routines are modified such that specification of a unique set of calling parameters ostensibly causes the system to return control to the penetrator in a privileged state (i.e. user program operation with the system's memory protection access rights or the processor in supervisor mode). In reality, specification of that particular set of parameters causes the SVC routine to program check. Thus, the Program Check Interruption Handler receives control prior to control returning to the user. The program check is distinguished from standard system or user program interruptions and handled via a special entrapment module.

- The processor hardware/firmware is modified to support special pseudo-flaws which are undetectable by the system penetrator. Selected SVCs (supervisor calls), as a result of hardware or firmware modifications, generate a program check interruption instead of the expected SVC interruption. The service routines which normally receive control are, in this case, dummy routines coded to appear as if they return control to the user in a privileged state. These routines, in fact, never do receive control; control is transferred, instead, to the Program Check Interruption Handler. The program check handler recognizes the penetration attempt, reports it, and passes control to a designated entrapment module.

Entrapment Modules

To augment the use of pseudo-flaws, the installation might introduce entrapment modules in the system software. Entrapment modules are system routines invoked, as a result of an attempted system penetration, to intervene in subsequent penetrator/system interaction. The functions of entrapment modules are to:

1. Preserve the utility of the system defense mechanism which detected the system penetrator by invoking a script to provide positive but inaccurate feedback to the penetrator indicating that his penetration technique was a success.

2. Direct the penetration attack down fruitless avenues of effort.

3. Monitor and record the efforts of the system penetrator to provide further evidence of intent to the system administrators.

At selected points in the entrapment module the monitoring could be discontinued if it were determined that the user had made an inadvertent error. If, on the other hand, the user's actions appeared suspect, his efforts might continue to be monitored, or, if appropriate, aborted. Consider the operation of entrapment service modules with some of the pseudo-flaws described previously:

- In the case of illegal LOGON activity an entrapment service module apparently allows the user to log on the system, but, in reality, invokes terminal echoing functions to record on a separate terminal all subsequent interaction the user has with the entrapment module. The entrapment module simulates execution of the user's commands until he has either satisfactorily established his intent or tried to execute a command the module can not handle. At this point, the user's efforts are aborted, and appropriate action taken.

- Entrapment modules invoked by the Program Check Interruption Handler return control to the penetrator in problem mode with his original storage protection key in effect. The penetrator is able to execute non-privileged instructions and access his own memory space, but unable to execute privileged instructions or access another user's memory space as he anticipated. Any attempt to do so causes another program check interruption, clearly establishes intent, and results in his activities being aborted.

Implications of an Entrapment Strategy

System Completeness:

Entrapment offers an approach to enhancing computer system

security which does not require that all exploitable system flaws have been located and corrected. Rather, it requires that (1) the majority of such flaws be corrected so that the penetrator must necessarily carry his search for an exploitable error to the target machine itself, and (2) the pseudo-flaws introduced into the system have significantly higher visibility than any residual system flaws such that penetration is attempted via pseudo-flaw.

Pseudo-flaw Placement:

Should pseudo-flaws be introduced at locations where there had previously existed significant errors in the standard vendor product, the penetrator would likely find and attempt penetration via one of these pseudo-flaws instead of searching for any obscure errors which might still exist. Thus, through the judicious placement of pseudo-flaws, the penetrator may, unknowingly, be directed toward attacking the strongest points of the system instead of the weakest.

Establishing Intent:

Utilization of entrapment mechanisms enables installation personnel to more clearly establish the intent of the system penetrator without divulging further information about the defenses of the system. A well disguised penetration attempt may be cloaked in a higher level language or self-modifying code, which, upon detection, the penetrator may assert represents an

inadvertent programming error. Failure of the technique gives the penetrator additional information about the nature of the system defenses being employed; the inability of systems personnel to clearly establish intent may permit the penetrator to attempt penetration by some other means. Entrapment mechanisms disguise the method of detection by permitting further penetrator/system interaction beyond the point of detection — without allowing access to protected information. Subsequent monitoring of the penetrator's interaction with the system may 1) yield additional evidence which clearly establishes that the penetrator was attempting security system penetration, and 2) indicate the particular information the penetrator was trying to access.

Confidence Levels:

Assuming a reasonable upper bound can be placed on the number of uncorrected, exploitable errors remaining in the security hardened system, it would seem possible to derive some estimate of the probability of occurrence of a successful system penetration for a given system. The introduction of counter-penetration mechanisms in the system offers some probability that the penetrator would encounter an entrapment mechanism prior to detecting an uncorrected, exploitable system flaw. A rough measure of the probability of detection for a given system might be based upon the number of holes likely to remain and the number of pseudo-flaws introduced in the final product. This could be further refined based upon some determination of the relative

visibility of the pseudo-flaws compared to any holes likely to remain. The result would define the level of confidence in the final product. A given installation might opt for a higher confidence level by increasing the ratio of counter-penetration mechanisms to the projected number of exploitable flaws in the system.

Applicability:

An entrapment strategy is applicable to new systems being developed as well as those in which security is being retrofitted. Unless the new systems are constructed such that they are demonstrably secure through application of program proving techniques to the design of the system, the problem of protecting sensitive computerized information from unauthorized access still exists, although, possibly to a lesser degree. Further, the logistics problem of implementing and maintaining a validated system still remains. Thus, it would seem prudent to incorporate fall-back defenses even in provably secure systems to allow for the possibility of unanticipated circumstances affecting the integrity of the hardware or system software.

Publicity:

It should be emphasized that, until completely secure systems are built, deterrence of potential penetrators can contribute more toward avoiding system compromise than can system hardening

retrofits. The most effective deterrent to system penetration activity could be to publicize the existence of defenses such as pseudo-flaws and entrapment modules, while not disclosing their implementation. General knowledge of their use as part of the system defense might serve to discourage penetration activity, since the effort required for a successful penetration would be appreciably greater, the likelihood of being detected increased, and the disguising of penetration attempts significantly more difficult.

REFERENCES

1. B. Peters
Security Considerations in a Multi-programmed Computer System
AFIPS Conference Proceedings SJCC 1967 vol 30

2. W. Ware
Security and Privacy in Computer Proceedings
AFIPS Conference Proceedings SJCC 1967 vol 30

3. L. Roberts (panelist)
Problems in the Age of the Computer
ASIS Symposium Proceedings Sept 1967

4. D. Branstan
Privacy and Protection in Operating Systems
Computer January 1973

5. R. Turn, R. Fredrickson, D. Hollingworth
Data Security Research at the Rand Corporation
P-4914 October 1972

6. M. Schroeder and J. Saltzer
A Hardware Architecture for Implementing Protection Rings
ACM Third Symposium on Operating System Principles 1971