M.I.T. Laboratory for Computer Science September 30, 1976

Computer Systems Research Division Request for Comments No. 125

TECHNICAL POSSIBILITIES AND PROBLEMS IN PROTECTING DATA IN COMPUTER SYSTEMS

by J. H. Saltzer

Attached is a copy of an overview/tutorial paper that I prepared for the recent Conference on Data Privacy and Data Security, sponsored by the German and Austrian Computer Societies, and held at the Johannes Kepler University of Linz in Donau, Austria on September 21-23, 1976. If you are interested (and read German) I have a copy of the complete conference proceedings.

This paper is citable in its original form, as follows:

Saltzer, J.H., "Technical Possibilities and Problems in Protecting Data in Computer Systems," in R. Dierstein, H. Fiedler, and A. Schulz, Datenschutz und Datensicherung, J. P. Bachem Verlag, Cologne, Germany, September, 1976, pp. 27-36.

TECHNICAL POSSIBILITIES AND PROBLEMS IN PROTECTING DATA IN COMPUTER SYSTEMS

Jerome H. Saltzer
Professor of Computer Science and Engineering
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts, U.S.A.

1. Summary

This paper briefly reviews the current state of technical knowledge about protecting computer-scored data from deliberate, unauthorized use and modification. The paper suggests that informal semantics have been devised to cop'- with many practical infor-ma Lion protection problems, but that assuring self-consistency and correct implementation of those semantics is still an unsolved research problem. Thus the present state of technical knowledge of computer organization cannot guarantee that use of computer-based information protection is risk-free. This paper provides a quick overview of a large variety of issues. The reader interested in a deeper discussion is referred to an earlier tutorial on this same subject [I],

2. Introduction

Protecting computer-stored data from deliberate, unauthorized use has over the last few years become a very popular topic for study, because of the widening use of computers as long-term data storage devices and at the same

time the widespread (if belated) realization of the dangerous potential for trouble if stored information is not properly protected [2]. A variety of incidents ranging from stealing of goods by unauthorized manipulation of a warehouse control program to penetration by attack teams of the most secure systems available provide convincing evidence that the potential is real, not imagined [3].

A large variety of techniques have been proposed for safeguarding information stored in computer systems. This paper concentrates on one sub-area: logical safeguards provided by the architecture (hardware and software) of the computer system itself.

Logically, one would hope that safeguards provided by the computer architecture could be perfect, in the same sense that addition of two numbers by a correctly-designed adder can be perfect. However, things are not yet as simple as that. Probably the main reason that protection is not as simple as addition, is that protection policies do not (except in some special cases) rest on a precise, mathematically manipulable basis such as that of addition. Most descriptions of protection policies are informal and imprecise, and therefore an architecture built to support those policies cannot be easily tested for compliance.

A second reason for difficulty in achieving perfection in safeguards implemented in computer architecture is that the architectural mechanisms that implement the protection policies tend to be logically much more complex than the mechanism of an adder, and are therefore more subject to design or implementation errors that are hard to discover. It is feasible to test an 8-bit adder design by exhaustive trial of all possible inputs, and perhaps a 32-bit adder consisting of four 8-bit adders could be shown to be correct by exhaustive demonstration of all possible carry situations, together with the previously confirmed assumption that the underlying 8-bit adders have been correctly designed. The same kind of pattern of establishing correctness of

a protection system design might also be feasible, but is still the subject of several research projects. Therefore, present protection systems should be presumed to contain undetected design and implementation errors.

3. Available Informal Semantics for Protecting Information

A wide variety of techniques has been used in computer systems to protect information, even without a formal, precise definition of how those mechanisms work or of their overall effect. Most effort at devising such techniques has been directed at shared, interactive systems, since they seem to have the greatest potential for abuse. Dedicated, single-user systems are often protected by mechanisms outside the computer architecture (e.g., placing them behind locked doors) while multi-user batch processing systems are usually of very old designs that have been widely considered "unprotectable" [4].

Since most of the techniques mentioned here have been thoroughly described elsewhere [1], the following brief list confines itself to discussion of some implications of each technique:

- user identification and authentication. Most interactive systems require that each user have a distinct, locally unique name, and that each use of the system be preceded by presenting both that name and some evidence that the presenter is authentic, such as a password, badge, or encryption key. Most of the remaining protection mechanisms of the system are then organized around checks based on the user's name. When family or given names are used, there can be a problem of maintaining uniqueness among user's names. Also, when one user authorizes another to do something (say to share a file) the first user mu&t know precisely the system's name for the second user. These problems are not severe in systems with small numbers of users, and generally, identification and authentication are areas that are fairly well understood.

- authorizing access to data. Two quite different techniques have evolved to allow one user of a system to authorize another to have access to data. In one, each differently protected data object has a list of authorized users (the access control list system). In the second, each user has a list of data objects that that user is authorized to use (the capability system). Generally, the access control list system provides auditability and and makes it easier to understand the implication of making an authorization, while the capability system allows a very rapid implementation of access control checks at program execution time. In both systems, what is controlled is access to the container of the data (that is, the storage area) rather than access to the data itself; neither system limits what a program can do with information it has derived from protected data it was authorized to use. (See the section on formal semantics, below, for one scheme of controlling flow of data.) Another problem, even with access control list systems, is that authorization systems have a large variety of options, features, and possibilities that cover specially needed situations; the novice user is overwhelmed by these features and makes many mistakes in specifying access authorization. There is an unsolved human engineering problem involved in making an easy-to-understand authorization system.
- limited-use systems. A widely-suggested strategy for reducing the risk of unauthorized information release is to restrict a system so that the only programs that can be executed are those provided by the system's proprietors. If no user-written programs are ever allowed to execute, the possibilities for unauthorized access are greatly reduced, and the difficulty of an intruder gaining access is greatly increased. This approach might be effective, for example, in a dedicated data base system used only for interactive queries with a high-level query language. One trouble with this technique is that there are many views about what

constitutes a limitation on use. There is a continuous spectrum between a system programmable in machine language and one that admits only simple typed inquiries and it is not at all understood how safety increases as one moves across the design spectrum,

- protected subsystems. A few research or development laboratories have experimented with hardware architecture that provides collections of programs and data (subsystems) with the property that access to the data is limited to only the programs of the containing subsystem [5,6,7]. Most proposals for protecting data base management systems have presupposed that such architecture is available for encapsulating in a protected subsystem the data base management programs together with the data base [8,9].

4 Policies that have been formally specified

To date, only one protection policy has been formally modeled, tested for consistency, and used as a basis for specifying correct operation of a compuiter-based protection system [10]. That policy is the protection model consisting of a small number of nested sensitivity levels and orthogonal, independent compartments. This protection model corresponds closely to the information classification system used by the U. S. Department of Defense, and also the "company confidential" classification system used by many private corporations. Even this fairly simple policy, under careful analysis, produces formal constraints that are not what one might have immediately expected. In particular, a program that has had access to information of high sensitivity level or from several compartments must be constrained so that it cannot write into any file unless that file is of equally high sensitivity level and labeled with the union of the several compartments. This constraint prevents a program from "declassifying" information it has had access to, on the assumption that a program's judgement on such matters cannot be trusted.

As such, it represents an example of direct control of information flow.

Notice that correct formal modelling of such a policy requires that all outputs of a running program be enumerated. While explicit program outputs (e.g., writing into a file) are relatively easy to identify, many programs have also implicit outputs that are much harder to discover. For example, the length of time the program runs, the rate of paging in a virtual memory, or the order of access to a read-only file may all be observable by other programs and therefore represent (perhaps very low-bandwidth) information flow paths that must be controlled for mathematical completeness. Identifying and cutting off implicit information flow is known as the confinement problem, and there has been some argument about whether or not there can exist useful shared systems that correctly provide confinement [11].

5. Areas with remaining problems

As is apparent from the above discussion, there is both a large collection of technical strategies in use, and several unsolved problems underlying their use. Some other specific areas that represent technical problems are the following:

- further formal specification of protection models. It would be of considerable interest if a complete formal model of access control on a name-by-name basis, with arbitrary patterns of information sharing, could be developed. Ties between these models and external world models, such as systems of privacy law, copyright law, and rules of fair use need to be developed [12].
- data base systems represent at least two problems that are not well handled yet. First, the existing access control mechanisms tend to be too ponderous or expensive to apply to small objects, such as individual field elements in a data base. Second, a lot of information in data bases is contained implicitly in indexes and summaries, and it may be

possible to infer a data value even if not directly accessible. Suggestions have been made that the output of protected data base systems might be deliberately distorted by the data base system, to prevent inference.

- Verifying compliance of a system to a formal model. Even for a formally specified protection model, there is a problem of establishing that an implemented system is in compliance with the model. Ideally, this verification might proceed in several steps:
  a) creation of a mathematically precise specification of the function of the computer system,
  b) verification that the system specification. does match the formally specified protection model,
  c) verification that the system implementation actually matches the system specification,

  This last step resembles a proof-of-correctness for parts of the computer operating system, a step that is a major effort and has not yet been completed for any significant system. A preliminary requirement for such a step is that the parts of the computer operating system that affect protection be identified, segregated, and organized in a systematic way [13,14].

6. Conclusion

The technical aspects of information protection by computer architecture include both a collection of pragmatic engineering approaches that are in wide use but still have a level of risk associated with them, and a set of formal approaches to information protection that promise to substantially reduce the risks but are at present incomplete and relatively narrowly directed. The former aspects seem to be sufficient to allow protection of mildly sensitive information, for which an intruder would be unwilling to risk detection

of his penetration attempts. On the other hand, information of high economic or strategic value is probably still best protected by physical controls outside the architecture of the computer itself. The formal approaches, required to allow protection of really sensitive information, are promising, and considerable research effort is underway to complete them and expand their breadth. In all cases, it must be remembered that protection provided by computer architecture is only one component of protecting information in computers, and many other aspects, such as physical site security, careful personnel selection, communication protection, and carefully designed operational procedures, must also be involved to achieve the goal.

REFERENCES

[1] SALTZER, J.H.
SCHROEDER, M.D.
The Protection of Information in Computer Systems
Proceedings of the IEEE 63, 9 (September, 1975) pp. 1278-1308.

[2] SALTZER, J.H.
Ongoing research and development on information protection.
ACM Operating Systems Review 8, 3, July, 1974. pp. 8-24.

[3] PARKER, D.
KYCOM, S.
OURA, S.
Computer abuse.
Stanford Research Institute, Project ISU 2501, November, 1973.

[4] ANDERSON, J,
Computer security technology planning study.
Air Force Elec. Syst. Div. Report ESD-TR-73-51, October, 1972.

[5] SCRROEDER, M.D,
Cooperation of mutually suspicious subsystems in a computer utility.
Ph.D. dissertation, M.I.T., Cambridge, Mass., 1972. (Also available as M.I.T. Proj. MAC Tech. Report TR-104.)

[6] COHEN, E.
JEFFERSON. D,
Protection in the HYDRA Operating System.
ACM Operating Systems Review 9, 5, November, 1975, pp. 141-160.

[7] NEEDHAM, R.
Protection systems and protection implementations
AFIPS Conference Proceedings 41, Part I, FJCC, pp. 571-578.

[8] HOFFMAN, L.J
The Formulary Model for Access Control and Privacy in Computer Systems.
Stanford Linear Accelerator Center, Stanford Univ., SLAC Report 117, May, 1970. Ph.D. dissertation.

[9] CONWAY, R.
MAXWELL, W.
MORGAN, H.
On the implementation of security measures in information systems.
Communications ACM 15, April, 1972, pp. 211-220.

[10] BELL, D.E.
LaPADUIA,, L.J
Secure Computer Systems: Mathematical Foundations.
The MITRE Corporation, MTR-2547, November, 1973.

[11] LAMPSON, B.
A note on the confinement problem.
Communications ACM 16, October, 1973, pp. 613-615.

[12] ROTENBERG, L.
Making computers keep secrets.
Ph.D. dissertation, M.I.T., Cambridge, Mass., 1973. (Also available as M.I.T, Proj.-MAC Tech. Report TR-115.)

[13] NEUMANN, P.G.
FABRY, R.S.
LEVITT, K.N.
ROBIINSON, L.
WENSLEY, J.H.

On the design of a provably secure operating system.

Institut de Recherche d'Informatique et d'Automatique (IRIA), International Workshop Protection in Operating Systems Rocquencourt, France: IRIA, August, 1974.

[14] SCHROEDER, 'M.D.

Engineering a Security Kernel for Multics.

ACM Operating Systems Review 9, 5, November, 1975, pp. 25-32.