J. S.

M.I.T. Laboratory for Computer Science 27 May 1976
Computer Systems Research Division Request for Comments No. 115

Validating the protection mechanism of a computer system

by Philippe Janson

The attached paper is a summary of the presentation I will give next month at the IRIA Workshop on Protection and Security in Data Networks. Even though a copy of this paper has already been sent for the proceedings of the workshop, comments are still welcome as they may help me for the oral presentation.

Validating the protection mechanism of a computer system

by

Philippe Janson
Computer Systems Research Division
Laboratory for Computer Science
Massachusetts Institute of Technology

1. Introduction.

The first objective of this presentation is to emphasize the importance of validating the hardware and the software of a computer system (network or individual system) to guarantee that the information stored in that system is adequately protected against unauthorized release or modification. The second goal of the presentation is to report the developments of a research project that is being carried out by the Computer Systems Research Division of the Laboratory for Computer Science of the Massachusetts Institute of Technology and is aimed at designing computer systems that are amenable to validation.

Throughout the presentation, we will assume the familiarity of the reader with general concepts about the architecture of computers and computer networks as well as with basic concepts related to information protection and security. Background knowledge on this latter topic can be found in a tutorial paper by Saltzer and Schroeder [Saltzer75].

We propose to treat our topic in four steps. First, we will relate the problem of validating the protection mechanisms in a computer network to that of validating the protection mechanisms in an individual computer. We will show that the two problems are analogous and justify our interest in the latter to study the former. Second, we will explain what is meant by validating a system. The concepts of correctness, security and certification will be reviewed. Third, we will describe the research project undertaken at M.I.T. to produce a system that is organized in such a way that in could be accredited secure after adequate validation. In particular, we will justify the approach to system validation that was taken at M.I.T. Finally, we will briefly report on some of the most recent progress that has been made within

Research reported here was performed by the Computer Systems Research Division of the M.I.T. Laboratory for Computer Science. It was sponsored in part by Honeywell Information Systems, Inc. and in part by the Air Force Information Systems Technology Applications Office (ISTAO) and the Advanced Research Projects Agency of the Department of Defense (ARPA) under ARPA order No. 2641, which was monitored by ISTAO under contract No. F19623-74-C-0193.

the framework of the research project towards designing a system that could be validated.

We will conclude the paper by enumerating the main problems that remain to be solved in the area of system validation.

2. Network validation.

The history of the development of protection mechanisms for computer systems comprises two stages. In a first stage, people have been concerned with inventing mechanisms for protecting the information stored in their computers. As more numerous and more sophisticated mechanisms were proposed, people soon realized that building protection mechanisms into their systems was not sufficient to guarantee the security of the stored information. Indeed, the protection mechanism of a system is subject to so complex interactions with so many other mechanisms in the system that it is extremely hard to implement it correctly, i.e. so that it effectively protects the stored information. Consequently, in a second stage, people have been concerned with building not just any protection mechanism but correct protection mechanisms, and with organizing their systems so as to keep a tight control on the interactions between the protection mechanism and other mechanisms.

Today, the development of protection mechanisms for individual systems is well into the second stage of its history. At the same time, the development of protection mechanisms for computer networks has just entered the first stage of its history and is barely starting to acquire momentum. Yet, we are convinced that it will go through the same stages as the development of protection mechanisms for individual systems. Therefore, it is reasonable to consider the validation of networks as an issue that will have to be faced in the near future.

In view of the upcoming problem of validating networks, it is justified to devote our attention here to the validation of individual systems because protecting and validating networks will undoubtedly include and depend on protecting and validating individual systems. If we cannot protect the individual hosts and the packet or message switching nodes of a network, we will not be able to protect the network. On the other hand, experience with the validation of individual hosts and nodes will carry us a long way towards validating networks composed of such systems. Thus, validating single systems is a prerequisite to validating networks. Therefore, it is recommended to study the former problem first and it is relevant to discuss it here before attacking the latter problem.

3. System validation.

In order to guarantee the security of the information stored in a system, it is not sufficient to provide the system with some protection mechanism. It is necessary to provide it with a protection mechanism that is validated, i.e. accredited to "adequately" protect the information in the system.

Verifying that a protection mechanism is "adequate" consists of showing that it contains no flaws. This may be very hard because it is essentially a negative kind of property: a single flaw is capable of defeating the whole protection mechanism.

The following method has been proposed to help validate a system. From the security standards that are perceived as desirable and informally defined by the community of users served by the system, one must derive a formal security model. This security model [see for instance Bell73] must be embodied in the formal specifications of the system. And those formal specifications must be enforced by the implementation of the system. The protection mechanism may then be declared "adequate" if it passes the following three tests:

-- verification of correctness, i.e. a verification that the actual implementation matches the formal specifications;

-- verification of security, i.e. a verification that the formal specifications do indeed embody the desired security model;

-- certification, i.e. an evaluation of the adequacy of the formal security model in representing the informal security standards set by the community of users.

The technically hardest part of the validation process is the verification of correctnes. Such a verification cannot be carried out on existing systems because they are too large and too complex. To simplify the task of validating a system, it is necessary to isolate the subset of procedures and data bases that are critical to security. This subset is called the security kernel. If the security kernel is properly isolated from the rest of the system code, it is sufficient to verify the correctness of only that security kernel to validate the whole system since any program outside the security kernel is, by definition, not critical to security.

4. A security kernel project.

The advent of general purpose, time-shared computer systems and distributed computer networks has fostered the need for operating systems that can be validated. In response to this need, the Electronic Systems Division of the United States Air Force has launched a security program aimed at producing an operating system that can be certified secure according to the U.S. military security standards.

The ultimate objective of the USAF security program is to produce a certified secure version of the Multics system. The Multics system was chosen partly because it already has substantial protection mechanisms built into its hardware. Therefore, Honeywell Information Systems, Inc. (HISI), which markets Multics, has been charged with the task of upgrading the Multics security kernel.

While HISI will provide the actual manpower to produce the new Multics system, it has charged our division at M.I.T., which initially developed Multics, with the task of exploring pragmatically and informally the content and the organization of a prototype security kernel for Multics. Our goal is not to produce the ultimate certifiably secure system. This will be HISI's task. Instead, we are studying what would constitute a suitable security kernel for Multics and how it should be structured to make it easier to understand and to audit informally [Schroeder75]. This pragmatic attitude is based on the axiom that anything we can do to make the Multics security kernel easier for us to understand and audit informally will make it easier for HISI to specify and to verify formally.

Our approach consists of evolving the existing Multics system rather than designing a new system from scratch. There are three reasons for this choice. First, to test a single concept, it is easier and faster to evolve an existing system, by modifying a piece at a time and trying the new piece right away, than it is to wait for the complete design of a new system. Second, by evolving a real system, with all the hardware, performance and functionality constraints, we avoid the pitfall of producing an entirely new but potentially unrealistic, awkward to use and oversimplified toy system. Third, our objective is to engineer a well-organized security kernel for a real system rather than to design a system that has a well-organized security kernel, i.e. we are interested not only in the conceptual aspects of designing well-organized systems in general but also in the engineering aspects of organizing well a given real system.

The task of evolving the existing system is simplified by the fact that Multics was designed to be evolved in the first place and is easy to modify.

In evolving the Multics system, our effort has been directed first at identifying the components of the security kernel, and then at reorganizing these components to make the security kernel easier to understand and to audit. Research towards the organization of the security kernel has been carried out at two levels. At the higher level, two doctoral theses are under way to explore the impact of various techniques for organizing a security kernel. At the lower level (master's theses), various case studies and experiments are being performed to test particular organizational techniques on specific areas of the system.

5. Achievements.

a. Organizational techniques.

The purpose of any organizational technique is to divide a system into a structured set of programs modules so that the modules are small enough to be understood and verified individually and the structure of the system makes it possible to infer a systematic (e.g., bottom-up) verification of the system from the verification of its individual modules. A first doctoral thesis [Feiertag76] has explored a structuring relation called the dependency relation. A module A depends on a module B if it assumes anything about the operation of B, i.e. if verifying the correctness of A requires verifying the correctness of B or in other words, if some (unexpected) action of B can cause A not to operate according to its formal specifications. Using the dependency relation to organize a security kernel in general leads to a partial ordering of the set of modules. The thesis demonstrates the advantages of a partial ordering. It also explains that most dependencies in the security kernel are explicit and easy to identify but some dependencies due to passing arguments between modules that operate in different execution environments are implicit and may be hard to identify.

A second doctoral thesis [Janson76] has explored an organizational technique based on the concept of type extension. This concept has been used thoroughly in the programming language area and in system design areas outside the security kernel, where it depends on the existence of virtual information containers that are always addressable, uniform and "growable". The thesis explores the problems of using type extension in the environment of the security kernel where such information containers cannot be utilized because they do not exist.

Two more techniques are envisioned to organize a system: partitioning and the use of parallel processes. However, they have not yet been explored thoroughly and need further reseach. Partitioning consists of enforcing the modularity of the security kernel by encapsulating each module in a protection domain. It has been proposed to use dedicated parallel processes to handle such asynchronous events as interrupts. Such a design captures t rue parallelism as opposed to the current Multics design where interrupts can be handled by any user process, thereby exhibiting a parallelism that does not necessarily correspond to the intrinsic parallelism of interrupt handling.

b. Kernel identification.

The major result of the kernel identification task has been the removal from the kernel environment of certain functions that are not critical to security. The dynamic linking function [Janson74, Janson75] and the name space management function [Bratt75], which are part of the top level of the file system, have been removed from the security kernel environment. In a first effort, they have been installed in the user environment. A project is currently under way to install them in an environment that does not contain the privileges of the security kernel but is nonetheless protected from the user. The linking and name space functions may operate in the user environment because that data they manage is private to each user. However, they are vulnerable to user mistakes. In a separate environment, their robustness will be enhanced. The use of a protected environment is also being considered for such operating system functions as the command processor.

c. Kernel reorganization.

Four areas of the Multics system have been reorganized: the system initialization mechanism, the process initialization mechanism, the multilevel memory management mechanism and the processor control mechanism.

In the system initialization area [Montgomery76], the bulk of the initialization task, which used to be performed when the system was brought up for service is now performed prior to bringing up the system. Thus, the tape that is used to bring up the system contains an image of the initial state of the system that can be verified correct, instead of containing copies of uninitialized procedures and data bases. A few tasks cannot be performed prior to bringing up the system because they depend on the configuration that needs to be brought up. These tasks are no longer regarded as initialization but are treated as part of dynamic reconfiguration.

In the process initialization area [Luniewski76], much of the code that creates and initializes processes, and authenticates users has been removed from the kernel, as it was deemed not critical to security. The security kernel only supports the creation of a process "frame" and the insertion of an execution site in a protection domain. A user is then given control over a process "frame" in the same way as an execution site enters a domain. "Entering" a process "frame" consists of inserting an execution site at an entry point of an initial procedure, called a gate, which is under the control of the owner of the process "frame". Access to the gate is further controlled by an access control list. This design makes the task of authorizing a user to control a process much simpler. In particular, authentication information, which can be saved by the kernel, allows a user to gain control over as many processes as he wants to create without having to authenticate him for every process he creates.

In the multilevel memory management area [Huber76], it has been recommended to use parallel processes to capture the parallelism of different functions. Each user process is responsible for bringing into core, on demand, the pages of information it references. However, the page removal algorithm is implemented by one dedicated system process for each level of memory. That process is responsible for continually maintaining available space at the level it is managing so that new pages can be moved to that level of memory.

Finally, in the process management area [Reed76], a two-level implementation of the processor multiplexing mechanism has been proposed. A structural problem encountered in many existing systems is a mutual dependency between the processor multiplexing mechanism and the memory multiplexing mechanism. The memory multiplexing mechanism uses the processor multiplexing mechanism to switch processes when a process waits for the completion of an I/O operation. The processor multiplexing mechanism uses the memory multiplexing mechanism to store the state of non-running processes. The two-level implementation of the processor multiplexing mechanism eliminates this dependency loop. The lower level multiplexes physical processors among a fixed number of virtual processors. That number is sufficiently small so that the state of all non-running virtual processors can reside in core. The higher level of the mechanism multiplexes virtual processors among user processes, whose state is not necessarily in core. Thus, the higher level of the processor multiplexing mechanism uses the memory multiplexing mechanism but this latter mechanism uses only the lower level of processor multiplexing to switch virtual processors. This two-level design has the additional advantage of allowing the use of virtual processes to implement kernel processes whose state is permanently in core, and which therefore need not depend on the memory multiplexing mechanism. The question is whether or not the two-level design of processor multiplexing can be implemented effectively.

This question is currently being explored.

d. The storage system.

Perhaps the most intricate area of the system is the storage system or virtual memory mechanism. By removing the dynamic linker and the name space manager from the kernel and by reorganizing the management of multilevel memory and the multiplexing of processors, the virtual memory mechanism has already been somewhat simplified. Yet, much work remains to be done because the mechanism performs exceedingly complex functions, including naming, protection, addressing, backup and resource allocation and accounting. The interactions of all these functions have not been well understood and the modules that implemented them are too few, too large and not well distinct from one another.

The type extension technique is being used to organize the virtual memory mechanism into a set of small and distinct type manager modules that is structured according to the dependency relation. The technique has proved to be particularily useful in identifying the implicit dependencies between modules, which usually are a source of complexity. It has also proved to be useful in guaranteeing that the system is deadlock-free, a property that usually is hard to verify.

This research project has lead us to conclude that type extension is a viable technique for organizing a real virtual memory mechanism. It has been used successfully to design a well-organized system. And it has proved able to cope with the complex functionality of a real system, which is more than most other organizational techniques have been demonstrated capable of.

6. Conclusion.

At this stage of the project, we have made substantial progress in identifying those functions that must be part of the Multics security kernel and in reorganizing the memory and processor management mechanisms of the security kernel. Work remains to be done in the I/O area, which has remained essentially unexplored until now.

Yet, before a certified secure Multics system can be produced by HISI, substantial progress will have to be made on program verification techniques. As modular and structured as the Multics system may become, it will never be amenable to verification techniques as they stand today. It may be amenable to informal human auditing. However, such auditing is a slow tool. If a system evolves rapidly, this tool may not be sufficiently fast to help certify every system version that is produced.

Leaving the scope of single hosts to envision the validation of computer networks, current technology is even further away from achieving substantial results. While we believe that efforts towards the validation of single hosts like Multics are useful contributions to the study of network validation, we are convinced that much more work will be necessary to validate a network. New problems that simply do not exist in the framework of single systems will have to be faced at the level of computer networks. For instance, the problem of validating distributed storage systems will certainly be raised. One may hope that techniques, such as type extension, developed to organize the storage systems of individual hosts will be applicable to distributed system. However, such hypotheses certainly deserve some thinking and would have to be tested.

Acknowledgements.

Research reported here is the fruit of the effort of a group of Faculty members, graduate students and staff members of the Computer Systems Research Division of the M.I.T. Laboratory for Computer Science. Considering the size of this group of people, we cannot give credit individually to each member for his or her contribution to the project. We can only give credit to the group at large for the whole project. Preparation of this paper has been aided by comments and suggestions of Professors D.D.Redell, J.H.Saltzer and M.D.Schroeder.

References.

[Bell73] D.E.Bell, L.J.LaPadula, "Secure computer systems", ESD-TR-73-278, Mitre Corporation (Nov.73).

[Bratt75] R.G.Bratt, "Minimizing the naming facilities requiring protection in a computing utility", MAC-TR-156, MIT LCS (Sep.75).

[Feiertag76] R.J.Feiertag, "A methodology for designing certifiably secure computer systems", to appear as TR, MIT LCS (1976).

[Huber76] A.R.Huber, "A multi-process design of a paging system", to appear as TR, MIT LCS (1976).

[Janson74] P.A.Janson, "Removing the dynamic linker from the security kernel of a computing utility", MAC-TR-132, MIT LCS (Jun.74).

[Janson75] P.A.Janson, "Dynamic linking and environment initialization in a multi-domain process", Proc. ACM 5 Symp. on Oper.Syst. Princ., ACM Oper. Syst. Review 9 5, pp 43-50 (Nov.75).

[Janson76] P.A.Janson, "Using type extension to organize virtual memory mechanisms", to appear as TR, MIT LCS (1976).

[Luniewski76] A.W.Luniewski, "A certifiable system initialization mechanism", to appear as TR, MIT LCS (1976).

[Montgomery76] W.A.Montgomery, "A secure and flexible model of process initiation for a computing utility", to appear as TR, MIT LCS (1976).

[Reed76] D.P.Reed, "Processor multiplexing in a layered operating system", to appear as TR, MIT LCS (1976).

[Saltzer75] J.H.Saltzer, M.D.Schroeder, "The protection of information in computer systems", Proc. IEEE 63 9, pp 1278-1308 (Sep.75).

[Schroeder75] M.D.Schroeder, "Engineering a security kernel for Multics", Proc. ACM 5 Symp. on Oper. Syst. Princ., ACM Oper. Syst. Review 9 5, pp 25-32 (Nov.75).