PROJECT MAC
Computer Systems Research Division

October 25, 1973.
Request for Comments No. 41

DESIGN OF A DYNAMIC LINKER RUNNING OUTSIDE THE SECURITY KERNEL OF A COMPUTING UTILITY.

by P. A. Janson

Massachusetts Institute of Technology.
Project MAC.
Computer System Research.
Cambridge - Massachusetts.
Proposal for Thesis Research in
Partial Fulfillment of the Requirements for the
Degree of Master of Science.

Title: Design of a Dynamic Linker Running outside the Security Kernel of a Computing Utility.

Submitted by: Philippe A. Janson
26 Clinton Street
Cambridge - Mass. 02139.

Signature of the Author:

Date of Submission: October 15, 1973.

Expected Date of Completion: January 1974.

Brief Statement of the Problem: In order to enforce the security of the information stored in a computing utility, it is necessary to certify that the protection mechanism is correctly implemented so that there exists no uncontrolled access to the stored information. Certification requires that the security kernel be much simpler and smaller than the supervisors of present general purpose operating systems. This thesis will attempt to prove the feasibility of reducing the security kernel complexity by removing the linking mechanism for program modules from the supervisor. The dynamic linker of Multics will be used as a test case. A complete design will be proposed for a linker running in the user's protection environment. Key components will be implemented to prove the correctness of the design.

Supervisor Agreement: The program outlined in this proposal is adequate for a Master of Science. The supplies and facilities required are available, and I am willing to supervise the research and evaluate the thesis report.

Michael D. Schroeder, Asst. Prof. of E.E.

I. Security, Protection and Certification of a Computing Utility:

The concept of a computing utility immediately suggests the idea of a community of people sharing the resources of a computer and in particular the information stored in the computer. It is obvious that different members of such a community have different goals and intentions which may be in conflict. The fact that a user stores information in a computer system should not make it possible for any other user to freely destroy, modify, use or steal that information. Information belonging to some person should be just as secure in the computer system as it would in the person's locked drawers. In order to enforce security, the stored information should be protected against unauthorized access. The protection mechanism is the set of components of the computer system which allows a user to impose access restrictions on his information (programs and data) which prevent other user's computations (processes) from accidentally or willfully accessing the protected information unless they are authorized to do so. A protection mechanism is useful only if it is trusted by the users. And the users will trust it only if it effectively protects their files. It is the responsability of the designers of the system to certify that the system is correct and that there exists no way to circumvent the protection mechanism.

II. Security Kernel.

The security kernel of a computing utility is the set of all programs and data bases anyone of which if tampered with, could cause unauthorized release, modification or use of information. The security of the whole system depends upon its security kernel being certified to correctly operate and being protected from subversion by executing programs.

Successful certification requires that the security kernel be small, simple and methodically designed. Size, complexity and obscure design make the certification much harder, if not impossible.

Protection from other executing programs must be sufficient to prevent such programs from breaking, circumventing or modifying the mechanisms implemented by the security kernel. In other words the security kernel must be isolated from all programs and data bases not concerned with protection mechanisms.

The security kernel of present commercial systems is isolated from the rest of the system by various techniques: master mode vs. user mode is the most frequent technique (IBM, Burroughs); execution of certain priviliged functions on separate physical processors prevents the users from subverting the protected functions (CDC 6000); the notion of ring, very close to the domain concept (1), is a powerful and most interesting protection means (Honeywell Multics).

Unfortunately, for most commercial computing utilities, the security kernel does not meet the protection and certification objectives outlined above. For various reasons it contains programs and data which implement functions irrelevant to the control of access to information. Yet, they run in the same protection environment and have the same capabilities as any module actually dealing with protection of information. Therefore, they could potentially subvert the protection mechanism and cause the unauthorized release, modification or use of information. Moreover such modules increase the complexity of the security kernel. As a result, security kernels of present computer systems may not be well protected, are hardly auditable and never certified.

III. The Linker.

The proposed thesis will show that the security kernel of a commercial systme can be reduced in complexity and that its protection can be improved by removing from it a set of programs referred to as the linker (or linkage editor).

A linker is a program or set of programs which is used to link separately produced procedure or data modules together. The action of linking modules together is the replacement of symbolic names in external references and calls by machine language entities (pointers or addresses) to be used at run time to correctly interpret these external references.

In many current computer systems, the linker is part of the security kernel. This is usually justified by reasons of feasibility, ease of implementation and economy at run time. The linker/loader of CDC 6000 (2) computers is executed in a peripheral processor. It is provided as part of the system and is invoked by the scheduler when appropriate. The call function of Burroughs 5500 (3) systems allows linking and loading of segments by means of interrupts causing the Master Control Program to run in privileged mode. The Honeywell Multics (4) dynamic linker is also invoked on interrupts and runs in the ring (domain) of the security kernel. In this last case, the design was particularly influenced by the initial high cost of domain crossing at run time.

IV. Motivations.

There is one fundamental reason why a linker should run outside the security kernel, in the user's protection environment. As is clear from the function of a linker, it handles information directly derived from source code provided by the users of the system. There is fair chance that such code contains -purposely or not - inconsistencies capable of causing the linker to malfunction or perform unexpected operations. In order to certify the security kernel we have to prove that malfunction or unexpected behavior cannot happen. It is hard to think of appropriate security checks on user code because potential inconsistencies in object code are numerous, vague and essentially unpredictable. Therefore, the only solution to the problem of certification is to remove the linker from the security kernel so that its eventual malfunction cannot affect the protection mechanism of the whole system.

It is not sufficient to show - as above - that the linker should be removed from the security kernel. We have to show that it can be removed i.e. that it does not need to be part of the security kernel to perform its linking function, and that the security kernel does not need the help of the linker to perform its task. These are the goals of the proposed thesis. The Multics system will be used as a test case.

V. A test case: MULTICS (4).

The Honeywell 6180/ MULTICS system was chosen as a test system for two reasons. Firstly, the system is available, maintained and developped at M.I.T. Secondly, among all commercially available systems, Multics is the only one which had protection of stored information as an initial design goal. Because Multics attempts to implement secure information storage, it will be a good system in which to evaluate the proposed changes.

The equivalent of a domain is a ring in Multics. The only conceptual difference is that domains in general can have totally different capabilities while rings can be viewed as nested domains (concentric circles) for which capabilities decrease as perimeter increases. There are eight rings in the address space of a Multics process. The innermost ring, called ring zero (o) has the most capabilities. Ring o is the domain where the security kernel is isolated as a protected subsystem.

The Multics linker is a dynamic linker. Instead of being linked before they are executed, modules are linked dynamically when an external reference is issued. The dynamic linker is invoked on a "linkage fault" interrupt each time the processor encounters a marked word, such a word indicates that access is needed to an external segment and that the link has never been snapped (set up) since the initialization of the current process. When invoked by a linkage fault, the linker executes in ring o.

The thesis will propose a completely new design of the dynamic linker. Instead of forcing control to transfer to procedures in ring o, a linkage fault will invoke the linker in the user's protection environment. Control will be transferred to the linker in the ring where the linkage fault occured. The link will be snapped in that ring.

It will be shown that the linker does not need to be in ring o to correctly perform its task: it needs no more capabilities than the faulting ring to snap a link.

It will also be shown that the security kernel does not need the linker to perform its task: the concept of dynamic linking is absent in ring o as the security kernel is prelinked when the system is initialized.

In order to demonstrate the correctness and the feasibility of the proposed design, selected key parts of it will be implemented.

References.

(1) M.D. Schroeder - Cooperation of Mutually Suspicious Subsystems in a Computer Utility. TR 104 MIT Project MAC - Cambridge - Mass. - Sept. 1972.

(2) Scope 3.4 Workshop Handbook
CDC 6000/7000 Development Services - 1970

(3) A Narrative Description of the B 5500 Master Control Program
Burroughs Corporation - Detroit - Michigan - Oct. 1969.

(4) Multics Programmer's Manual - Part I
Revision 12 - MIT Project MAC - Cambridge - Mass. - Nov. 1972.

VI. Proposed Work.

The work required for this thesis can be divided into two phases: the design and the implementation of the dynamic linker.

The purpose of the design phase is to conceptually prove that the linker can be removed from the security kernel. This phase is almost completed.

The purpose of the implementation is to demonstrate the feasibility of the design and to take advantage of it in future versions of the system. There seem to be three parts in the implementation phase: the programming of the dynamic linker itself; the modification of data bases and programs of the security kernel to cope with the new dynamic linker; and the programming of new features required in the security kernel to support the initialization of the linker. These three parts seem fairly separate and could be undertaken in parallel with no trouble. Within the scope of the proposed thesis, we will concentrate on the third part of the implementation. It seems to be the most complex part and therefore a potential source of mistakes which the design must avoid. It also is the most interesting and educational part of the implementation in terms of computer system design and development.