PROJECT MAC September 5, 1973

Computer Systems Research Division Request for Comments No.34

SOME SYSTEM CERTIFICATION TASKS

by J. H. Saltzer

One of the goals of this summer's work has been to lay out in more detail some specific tasks which will allow us to get started on the overall project of developing a certifiably ("auditably") correct protected core for Multics. This note describes several specific tasks which so far have been identified as plausible candidates; the purpose of this note is to promote discussion -- better defining these tasks, and to encourage suggestions for tasks not defined here.

Several of the tasks suggested here involve modifications to the current Multics system. For each of these, two observations are in order: 1) a method of measurement of progress is needed, to establish "how much" each modification carries us toward the goal of an auditable central core; and 2) discussions and negotiations with Honeywell are required to establish whether or not each suggested modification should be targeted toward installation in some current or future standard version of Multics. It seems inevitable that at least some of the changes which will be needed to achieve an auditable system will violate either compatability or performance constraints of the standard system, and thereby force a development of a parallel version, which we might call Multics/A (for Multics/Auditable).

On the whole, it will be seen that most of the initial tasks are directed toward identifying more exactly which functions of the operating system must be privileged, and which, by careful design, can be left to the user (in Multics, on a per ring basis.) This work may be described as better defining where the security perimeter of the system should be located. It is expected that there will be many more such tasks in this class. I estimate that this general class of activity, when completed, will represent perhaps one-half to two-thirds of the intellectual and programming effort of the overall project. Two remaining major areas of work, both more suitably tackled later, are the rewriting of otherwise untouched protected programs in a standard auditable style, and installation of at least one internal firewall or protection ring within the protected supervisor to separate those procedures which actually implement the protection mechanism itself -- a so-called "protection kernel".

The tasks so far identified are the following:

1. Removal of the dynamic linker and library search modules from ring 0. This modification would remove two large and hard-to-audit modules from the protected area. The dynamic linker is especially hard to audit because its correct operation depends on its interpreting a highly structured but unprotected data base (an object segment linkage and definition area) without accidentally getting mixed up. Neither of the modules has need for supervisor privileges or protection from the invoking user; both are currently in ring 0 because of their intimate interface with the storage system. The task includes better definition of the interface to the storage system, and taking advantage of the lower cost of changing protection rings with the 6180 hardware.
2. Removal of the "reference name" concept from ring 0. The notion of a remembered reference name is currently maintained on a per-ring basis, in the per-process known segment table in ring 0. There is no apparent reason why reference names cannot be remembered in the ring of interest; such an arrangement will also permit a subsystem writer to disable reference names if he desires. This change would simplify both the implementation and the description of several supervisor interfaces.
3. Removal of the "working directory" concept from ring 0. The comments regarding reference names apply to the working directory also.
4. Develop a uniform storage system status-returning entry. This minor cleanup would replace about half-dozen distinct supervisor interfaces with a single, more easily audited interface for returning to the user any status information about his segments. (This task is actually the iceberg tip of a larger task to develop a simple, consistent set of supervisor entries.)
5. Modify the traffic controller to provide cheap, rapidly scheduled, wired-down processes which can operate using any descriptor segment which happens to be available in primary memory. This change would allow the present interrupt handlers for the printer, teletype interface, network interface, and tape handlers to be replaced with scheduled processes.

The actual interrupts would do nothing but notify the appropriate process. The virtue of this strategy is that scheduled processes can coordinate their activities with standard coordination primitives (block, wakeup, wait and notify); the present interrupt handlers cannot, for example, wait on an interlock, and are therefore filled with tricky code which uses read-alter-rewrite instructions to avoid encountering interlock situations.

6. Modify the traffic controller (and other per-process data base managers) to permit multiple processes per address space. This modification is the key to untangling several very complex paths through the present supervisor. Typewriter management, network interface management, dialup handling, and quit handling can all be done as simple coordination of parallel processes rather than with the present ad hoc multiplexing of a single process among many conceptually parallel activities. The propagation of this change through the network control is planned as part of the task, to test it's effectiveness.

7. Develop a uniform process coordination/message passing strategy. The current Multics has several different coordination and message passing schemes in it, each with slightly different properties as to the scope of naming and details of interface:

   -- Wait and notify, used for storage system signalling

   -- Block and wakeup, used for I/O coordination

   -- Interprocess communication, used for multiplexing processes among event call channels

   -- Signals, used to generate interrupts in a process

   -- Message segments, used to queue messages in a catalogued place

   -- Mail facility, used for inter-user mail

   -- Lock and Unlock, used for coordinating data base use

   -- The I/O system, used for message passing and queueing

   The task here is to develop one or two moderately flexible process coordination and message passing facilities which can be used to support all of the various users of these facilities. The payoff in simplification of the central supervisor should be quite high.

8. Merge the network interface with the typewriter communications interface. These two interface programs are two of the largest protected subsystems; they largely duplicate each other. The typewriter control system should use the network code conversion strategy which does not require protection; the network interface should use a buffering strategy more similar to the typewriter modules. With moderate effort, the interface between the 6180 and the DataNet 355 communications computer can be made essentially identical to the network host-to-IMP interface, allowing further control program sharing. By taking the best design from each of the two systems, a compact and effective communication interface module should result, with minimum privileged code.

9. System Census. This task consists of conducting a census of the number of programs, number of lines of source code, and number of lines of generated text (machine instructions) in the protected supervisor. This census will be useful for two purposes: identifying subsystems which are unreasonably large or complex for further study, and to keep track of progress in simplifying and reducing the size of the protected supervisor.

10. ALM program catalogue. A list of all protected programs currently written in ALM (the Multics Assembly Language) is being developed, with the goal of identifying all reasons why assembly language has been used. This task includes the development of proposals to eliminate the need for assembly language completely. Such elimination is an important step in simplifying the description of the system and of simplifying the job of an auditor.

11. Development of coding style standards. A standard programming style will need to be developed, one which emphasizes clarity in program structure to an auditor. Undoubtedly, the programming style will borrow much from the emerging area of structured programming. The task includes the experimental rewriting of some parts of the storage/directory system to the new standards to test their viability.

12. Use of unique segment number. The implication, in terms of simplifying system structure, of using unique identifiers for segment numbers will be explored. An immediate implication of such a strategy would be that pointers containing segment numbers could be left in permanently

catalogued, shared storage; many programmed tricks to accomplish the equivalent effect could be eliminated from the system. There are many other implications for system creation, interprocess communication, dynamic linking, and hardware addressing architecture which should be examined; many simplifications seem to follow. An intermediate strategy, of using unique identifiers to replace the absolute addresses in a segment descriptor word, and developing a microprogrammed memory controller architecture which responds to such unique identifiers and contains in a separate box all virtual memory implementation seems worthy of exploration as part of this task.

13. I/O hardware architecture proposal. A key result of the thesis just completed by D. Clark is that, with correct design, essentially no I/O strategy or device management code, except that dealing with multiplexed channels, needs to be protected. Since I/O software is a significant part of the present protected supervisor, a detailed design proposal for a new hardware I/O architecture along the line of Clarks' thesis is in order. Thanks to the modular organization of the 6180, it is relatively easy to envision actually building and trying out this design at some point in the future.

14. Reconfiguration hardware proposal. A fair amount of very intricate machine language code in the protected core of Multics is devoted to the dynamic reconfiguration of processors and memory, a valuable feature. Much of the intricacy can be attributed to performing reconfiguration with hardware not designed for it. A general design developed by R. Schell in his 1971 Ph.D. Thesis should be reviewed and a specific hardware proposal for the 6180 system should be constructed along the lines suggested by Schell. Such a design would probably influence future rather than current versions of the Multics hardware but the result is of interest now to establish how large is the effect in reducing complexity of the protected supervisor. In addition, operation of a secure system probably requires padlocking many of the control panels currently used by the operator to accomplish dynamic reconfiguration.

15. System description improvement. If an auditor is to review a supervisor program for correctness, he must have a complete, concise statement of what the program is intended to do. Today's description consists of English language supervisor interface descriptions, with PL/I calling sequences. There is no simple description of the "state" of the supervisor and the things a user may do to legally alter its state. The first step in this task is simply to collect in one place all the present documentation of the protected supervisor interface, and evaluate it. The next step is to try to develop a more precise state description of the supervisor, and the ways in which a user can change or observe its state. This task seems to include becoming expert in description languages, such as the Vienna Definition Language, so as to develop equivalently powerful methods of describing an operating system.

Some of the fifteen tasks described above are already being worked on; others may not be started for some time, depending on availability of personnel to do them. A detailed schedule of target completion times has not yet been developed.