To: Distribution

From: Andrew M. Kobziar

Date: May 30, 1974

Subject: Solutions For Security Holes in Additional Storage Controls

Introduction

MTB-081 introduced storage system changes which, together with the changes summarized in MTB-047, implement a multi-level security system on Multics. The effect of security is that users and their files are compartmentalized into disjoint groups. Security requires that no uncontrolled information paths exist between compartments, and that access mode rules for levels are observed within a compartment. The implementation described in MTB-081 leaves open several such paths. Although these paths are security holes in a formal sense, the information bandwidths are on the order of a bit per second and are of little or no value for direct penetration.

The purpose of this MTB is three-fold:

1) To list the security holes that will remain after the MTB-081 implementation.

2) To show solutions for these holes which can be implemented in the present storage system.

3) To establish constraints that must be considered when the storage system is upgraded or redesigned.

Two types of security threats exist which could utilize these information paths. The first is collusion between users. Unfortunately, all side effects of Multics use, such as implicit propagations up the hierarchy or the appearance of a user's name when "who" is typed, are candidates for collusion. Only the elimination of all security holes can remove Multics as a potential vehicle for collusion. The second threat is that of an unknown "extra" (a trojan horse) in an executable procedure. In both cases the closure of all holes limits the compromise to violations of need-to-know in that compartment's user group.

These holes must be removed before attempting security certification. Since certification is some time off and the removal is non-trivial, the implementation described in this MTB may be postponed until the next redesign of the storage system. First, the automatic updating of date-time-used and date-time-modified is discussed. The variable length of file maps is dealt with next. Then a proposal is made for containing the quota changes caused by upgraded directories. Following this is a commentary on the special handling of access by ring 0. Finally, requirements and suggestions are listed for the removable hierarchy study.

DTU and DTM

Date-time-used is updated whenever a segment or directory is deactivated (or status requested on an active segment or directory.) If a top secret process executes an unclassified library routine, the changes in date-time-used can be detected by an unclassified process. A simple change in active segment table management not to update the date-time-used unless at least one process of equivalent clearance references that segment or directory will solve this problem. The global transparent usage switch, which is currently used by backup, will not be turned off unless a process of equivalent clearance takes a segment fault on the segment or directory. Notice that when Multics is operated with everything at one classification, the resetting of date-time-used retains its present meaning. Since finding a segment involves the searching of superior directories, the date-time-used value for directories is the time the directory was last active. In conjunction with the above change the meaning of directory date-time-used could be changed to exclude resetting whenever deactivation is performed. The date would then reflect actual use, such as listing or status.

The modification of a segment or directory causes the updating of the date-time-modified field. In order to aid backup's search for recent changes, this date is propagated through all superior directories. Since the security system has non-decreasing classification with increasing depth in the hierarchy, modifications performed by high level processes could be observed by unclassified processes. The best solution here is to define a new branch item, date-time-subtree-modified for the exclusive use of the dumping process. This would not be available to users, and would allow date-time-modified to hold the local meaning only.

File Map

A more intrinsic problem exists with the branch of an upgraded directory. Since it is impossible to protect a branch (because changes in its size can be observed in the parent directory), items that reflect information about the (possibly changing) physical characteristics of an upgraded directory must be protected. First, current length and bit count must be protected. Second, the mechanism for obtaining larger file maps as the directory grows must be disabled for upgraded directories. The current mechanism obtains new file maps at the 4th and 16th page boundary. A possible solution is to force the directory to acquire and retain the largest file map at the time it is upgraded.

To complete this securing task, the upgraded branch must be split into two classifications. Variable length lists associated with the branch, such as ACLs and names, must remain at the level of the parent, but fixed length items such as current length and bit-count that are usage related have to be at the higher level. The bit-count author has to be further restricted to be a name already in the directory, as the setting of a novel name could increase the size of the parent directory. This will protect the values in the branch which change automatically with the use of the upgraded directory, since the use is at the higher classification.

Quota

In order to prevent the size of the upgraded directory from being reflected in the parent's quota, quota must be self-containing for a directory. Two non-trivial changes are required. Since quota must be enforced by page control, it is stored in the active segment table entry for the directory. But because of a cart and horse situation this value is not available at the time the directory is activated. The value of quota is stored in the directory header (first page) and is not accessible to page control until processing of the segment fault (which includes activation) finishes. The current mechanism waits until the first activation of an inferior branch to fill in the parent's quota since a page fault has to be taken to find the inferior branch anyway. Quota was probably defined to refer to the inferior subtree because it was not available at the time of activation nor is it defined at the time of creation. To be self containing, the value must be stored in the branch rather than in the header. If this scheme is adopted, some special casing of the root, which must be a terminal account (i.e. has a quota > 0) and has no branch, must be worked out for page control as well as for the movequota primitive. The other change required is that switching between terminal and non-terminal status (via movequota) must take into account the directory's own length.

A quota redesign that is currently being discussed suggests that all directories have quotas, and that pages used be reflected through all superior directories to the root. If an overcharge occured anywhere along the line, then a quota overflow would be signaled. Thus only the root's quota would be "terminal" and the quota on each directory only a check point. Because security requirements forbid upward propagations, this scheme must be altered to retain the terminal nature of quota on all upgraded directories.

The SysDaemon Window

Whenever a segment or directory is created the virgin ACL is initialized with an entry for "rw *.SysDaemon.*". Next, the Initial ACL is added. Finally, an ACL entry for the user is added. The placing of the SysDaemon entry was done in good spirit, saying "trust all our system services." During the storage system dark ages before Initial ACLs, this was the only automatic method to eliminate the worry about Backup and I/O for carefree users. A user could always delete the SysDaemon entry later, leaving only a small window of time for SysDaemon mistrust.

What is significant here is not that "*.SysDaemon.*" is on the ACL to a segment but that it is placed there implicitly by ring 0. How can you prevent an administrator from registering "President.SysDaemon," a process which keeps or truncating your segment "impeachment_data"? (Multi-level security will allow the restructuring of adminstration based on essential access. If the administrator is limited to run unclassified, he does not have to be cleared or trusted, as he will not be able to force access to any classified data even though he has modify permission on user-dir-dir.) If ring 0 gives special permission to the SysDaemon project, a certification of ring 0 must also include all SysDaemon processes. Clearly this is not the direction we wish to take, since certification today is viable only on a small scale.

Today a user can close the window by adding "null *.SysDaemon.*" to the Initial ACL. Since the Initial ACL is added after the SysDaemon entry by "append," this resets the mode to null. But unless ring 0 forces Initial ACLs to start with a "null *.SysDaemon.*" entry, this does not fulfill the certification requirement. An alternative mechanism which retains the carefree characteristic but doesn't require SysDaemon certification is proposed. "Append" is changed to copy the Initial ACL rather than put "sm *.SysDaemon.*" when creating directories. As long as new users have *.SysDaemon in their home directory Initial ACL the desired utility is preserved. This change is significant because now the installation administrator, project leader, and

user are the ones deciding the appropriateness of the SysDaemon entries and not the supervisor. If this change is adopted, users must be made aware of the necessity for checking (and correcting) Initial ACLs whenever they create directories.

Implicit Access From Ring 0

Special access to directories is given to the Initializer process by ring 0. Either modifications to allow system functioning without this special access or the certification of the Initializer process is required. If the latter is chosen, admin mode, which allows execution of Multics commands from the Initializer, must be disabled.

Removable Hierarchy

Although the ability to physically remove parts of the hierarchy (via disk packs, for example) is not included in the present task, the design committee evolved several guidelines which should be of interest to the new storage system project. These recommendations are:

1) Each physical disk pack must be identified as part of the hardware known to the storage system, such that it would be impossible for any process to use such a pack for I/O. Some hardware interlocks such as that provided on prerecorded cassettes is desirable.

2) There must be a unique, machine-readable header on each physical pack. No disk pack may be used for demountable segments until the header has been initialized and the pack's existence recorded in permanent storage. If encryption is used to protect the data on the packs then the location of the header could be a function of a unique id kept in permanent storage and available only to ring 0. This would help prevent the use of the structured nature of the header in an illegal decoding attempt. The header must identify the highest classification storable on this volume. This classification should be used by the system when deciding on which pack to place a segment.

3) No specific relationship between the contents of a disk pack and the logical hierarchial position of the segments is necessary. On the one hand, it appears to be more economical to group on the basis of process group id or by directory subtree. On the other hand, the more secure approach is to have no relationship so that knowledge of an unclassified segment being online cannot be used as

knowledge that inferior, classified segments had been referenced. If we assume a "demount/move_online segment" command, then the choice of which pack is used for the request must be totally managed by the system as well as the exact time of movement to or from a volume. The allocation of drives to packs must also be under complete control of the system. Physical requests to mount a pack will be issued on the basis of a label but will be checked by comparing the permanently stored pack identification with the pack's header. Any discrepencies must be audited for security evaluation.

4) There exist no security reasons to limit removability only to segments, especially if the allocation to a volume is at one classification level. We assume that removed packs will receive as much physical protection as do the permanent ones. If a pack is ever removed from the protected area, then the contents cannot be trusted and it must be completely recreated, unless sufficient encryption was used during generation and the pack contains information at only one classification.

Summarizing, the major weakness these recommendations address is the deductive knowledge (which can be used as a bit of information) gained from the on-line status of a demountable segment. Assigning only one classification and only one project to a pack are methods which localize the problem. A total solution short of allocation of packs/person is not yet known.