To: Distribution

From: Andrew M. Kobziar

Date: May 2, 1974

Subject: Additional Storage System Security Controls

Introduction

This MTB covers the changes that Honeywell has agreed to make to the Multics storage system. It is part of a series of MTB's on several different functional components planned for this project and introduced in MTB-047. Although the final goal of this endeavor is to produce a multi-level security system, the contracted implementation will only cover part of the work required. As will be seen later, security "holes" will remain. Proposals for the elimination of these holes are given in a companion MTB entitled "Solutions For Security Holes in Additional Storage Controls."

A brief review of the access control requirements in a military security system is in order. Every object (segment, terminal, message, etc.) is given a level number Lo (0<=Lo<=7) and a category set Co (any of 16 bits). The pair [Lo,Co] will be referred to as a "classification." Every process has a level number Lp (0<=Lp<=7) and a category set Cp (any of 16 bits) which together form a pair [Lp,Cp] termed "clearance." (The nomenclature used for level/category values is installation definable. The military's security system terminology will be used in examples in this MTB.) The security rule is given as follows:

```
                                /* if process's level is too low
                                   or the object's category set is
                                   not a subset of the process's
                                   category set */
   if (Lp < Lo) | ~((Cp & Co) = Co) then mode = null;

                                /* if level and category match
                                   then allow full mode */
   else if (Lp = Lo) & (Cp = Co) then mode = ACL mode;
```

```
                        /* otherwise allow reading */
    else mode = <read, execute> & ACL mode;
                          /* for segments */
              = <status> & ACL mode;
                          /* for directories */
```

This allows a process to "read down" but only write when classification = clearance. Although in the paper world "write up" often occurs when information is given to superiors, on Multics "write up" would permit the sabotage of segments by overwriting information with garbage. Append mode in a directory does represent a safe but useless "write up" capability (since a process would not have access to the created segment) and will not be implemented. In addition to the above rules on classification and clearance, access to documents in the paper system also requires a need-to-know. Here the standard Multics access control list (ACL) duplicates the paper world in placing responsibility for determining need-to-know on the holder/owner of the document.

It is theoretically possible to have the clearance of a process increase automatically as the process references segments of higher classification. (Without a heuristic that automatically classifies a segment from its contents, it is necessary to classify everything that a process writes at least at the level of the highest segment read.) This would allow a user to answer her/his mail and messages and then go on to classified work. To effect a change in clearance all segment descriptor words must be faulted to require new access computations and all temporary segments (such as the stack) would have to be upgraded. It was judged that the usefulness of this capability did not warrant the cost and difficulty of implementation. Thus the clearance of a process will be constant for the life of the process. The classification of a segment is also constant for the life of its unique id.

Background

A principal guideline for changes to the storage system states that the additional security controls must be an integral part of Multics. From our viewpoint, it should also be transparent and inexpensive for installations not using the security features, as well as simple and potentially certifiable.

The paper world provides physical security for the storage of classified material via safes. During the time that a person is working with a classified document (at a desk), the responsibility for protecting that document is placed on the person. Before divulging classified information to another,

clearance must be checked and reasonable need-to-know established. If a person not meeting these requirements walks into the room, the holder must cover or remove from sight all classified material. In the computer world, not only must "physical" security be provided for stored files, but also the security to "cover" all information in current use. There is an added danger in a computer system in that users cannot see every action of the programs that they are executing. Thus in the computer environment the responsibility for eliminating all information paths from a higher cleared user to a lower cleared user (i.e. "write down") is placed on the system.

Design

As in the introduction, the format of this paper will be to present the design along with the various alternatives that were rejected. The first topic is the relationship between the classification of a directory and the classification of its contained branches. Next, the changes necessary to implement a multi-level security system are presented. These include the introduction of "security out-of-service" for reloading, retrieving, and salvaging. Finally, the problems of starting security in an existing installation and the implications of security to users are given.

Information Paths

Rationale for the relationship between classifications of a parent directory and its inferior segments and directories was given in MTB-047. Inferior directories must have a classification equal to or higher than their parent. Otherwise, a higher cleared process would be able to pass one bit of information to a lower cleared process by changing the name of one of the middle components in the pathname of a segment which the lower cleared process was continually checking. Similar reasoning was used to arrive at the restriction that the classification of a segment must not be less than the classification of its parent directory.

What about a segment at a higher classification than its parent? Information about a segment's attributes (ACL, name, max length) and about its contents (bit count, current length, and date-time values) are stored in the segment's branch, physically located in the parent directory. Changes to the branch data pertaining to the segment's contents could be observed by a lower cleared process unless the branch was classified at the higher level of the segment. Unfortunately this solution is insufficient as three paths to "write down" information would still exist.

1) Changes to the segment's length may be observed in the value of the quota stored in the (lower classified) parent directory

2) The size of the parent directory may be increased by adding to the ACL or name list of the higher classified segment. Also if the directory's size is near a page boundary, growing the segment could require a new file map which would cause the parent directory to grow another page in size. Even setting the bit count, which implicitly sets the bit count author, could cause the parent to grow.

3) The lower cleared process may find the name (and therefore existence) of a higher classified branch by guessing names when creating a segment in the directory and receiving a name duplication error.

The third error could be eliminated via an edict stating that names may not contain classified information. Because no solution short of restructuring directories is known for 1 or 2, the restriction on segments states that they must be of the same classification as their parent. This forces all segment upgrade and downgrade operations to be actual moves to different directories. Downgrading (i.e. declassifing) will be further restricted to be only available to the system security officer.

Upgraded Directories

A mechanism for obtaining a directory at a higher classification than its parent is still necessary: otherwise the entire hierarchy would be at only one classification (as is the MIT system.) A new primitive, "upgrade_directory", will raise the classification of an empty directory with a terminal quota. (The quota must be > 0 to prevent the recording of pages used in a lower classified parent.) This operation will be performed by a process at the clearance of the parent directory as it requires modifying the inferior directory's branch. The use of the lower clearance for this function also prevents a write down to a process watching for the completion of the upgrade. Once upgraded, the directory is not accessible to the upgrading process. A new process at the higher clearance is required in order to use the directory.

Classification of the Branch

Three options for the classification of an upgraded directory's branch were discussed. Classifying the branch at the higher level was rejected since the same write down paths exist as did

for the higher classified segment branch discussed earlier. An ad hoc solution to classify the name at the parent's level and the rest of the branch at the higher level did solve the name duplication problem. Together with the auditing of any attribute changes (such as ACL additions) it might have been effectively secure in a working context, but this was only a postponement for a concrete design. The extra costs of such treatment as well as non-homogeneity are reason enough for its rejection. The last option, and the one to be implemented, considers the branch at the level of the parent directory. Changes to values stored in the branch which reflect the state of the contents of the upgraded directory, as well as the size of this directory reflected in the parent's quota, remain as security holes.

When viewing the Multics hierarchy as an inverted tree, classification increases in the downward direction. All explicit values and (implicit) side effects that propagate up the hierarchy are security holes and render the system unsecure. With the installation of the restructured storage system in system 18-0 an architectual dependance on upward propagation was introduced. This is caused by variable size file maps which can cause every superior directory to grow a page when a segment is grown. Other upward propagations, such as date-time-used, are required by the present implementation but are not essential to the design. Suggestions for eliminating their effects will be given in the future MTB on the changes necessary for a file system design that is able to pass certification.

Part of the storage system modification task will be the inclusion of calls to record audit information. Auditing will also be used to record certain security events such as downgrading. The information stored will be selective by process identification as well as by event or operation in order to minimize cost and output. Another MTB on the subject of auditing is included in this series of MTBs.

Implementation

A. Access Control

The classification of each segment or directory will be stored in its branch. Two words currently labeled as "pad" will be used as follows:

```
dcl 1 security aligned,
    2 categories bit(36),
    2 level fixed bin(17) unal,
    2 pad bit(18) unal;
```

This structure exceeds current security requirements for two reasons:

1) The maximum number of levels and categories permitted by this structure is thought to be sufficient to support all forseeable applications.
2) A tradeoff between storage and code efficiency has been made in favor of code efficiency.

The clearance of every process will be stored in two words in its PDS using the same structure declaration. Also stored in the PDS will be special privilege bits and special auditing bits. (The special privilege bits will be used to skip security checks for processes such as the initializer, I/O coordinator, and retriever which must be able to work at many levels.)

Presently the storage system calculates a process's raw access mode to a segment via the procedure "access_mode" and the effective mode (which includes ring brackets) via "fs_get". Incorporating the extra security checks in "access_mode" appears to be the best solution for both security mode calculation and auditing. The returned mode will now have the security rules factored in. Certain procedures will still require a raw mode and this will be available via another entry in "access_mode". For example, both "status_" and "quota" require the raw mode bits. The inclusion of security checking in "access_mode" will add about 20 instructions to every access mode calculation. The classification of all new branches will be set by "append" to the value in the parent directory.

B. Movequota

When computing access to move quota, the problem of obtaining modes is somewhat more complicated due to the possibility of a move between a parent and an upgraded directory. Since the move can be observed at the classification of the parent, the process performing the move must be at that clearance. Thus quota must itself calculate a non-security access mode to the upgraded directory. The move is only allowed if quota is increased in the upgraded directory (since subtracting quota would be a way of finding its value as this would fail when quota = used in the upgraded directory.) Implementing a quota move in this manner only allows a "write up," as it is always done at the lower clearance.

C. New Procedures

New procedures to either reset or return classification and clearance levels will have to be added to ring 0. Resetting the classification will only be available to a reload/retriever process and the system security officer. The procedure resetting classifications will also have an entry, available via hcs_, to upgrade empty, terminal account directories. Return of

classification and clearance will be available via hcs_. Two commands will be modified to make use of the new hcs_ entries. List will have a "-effective_mode" control argument which would return the effective mode (this includes security and ring brackets) instead of the raw mode on everything listed. Status will be changed to print out the classification when a "-mode" control argument is given as well as including this item in a "-all" control argument.

D. Backup

The backup process will operate at the highest clearance and must circumvent security in order to set the date-time-dumped field in all branches. Rather than use a special access bit to allow backup to circumvent security in all directory references, a special gate entry is proposed which would only be available to Backup.SysDaemon to update the date-time-dumped field. If a new gate is not wanted then hphcs_ could be used. To preserve the classification, backup will require modifications to store a new type branch listing which includes the classification. At some time prior to the installation of the changes that back up the dad field, a salvager which zeros this field will be installed. A version of "append" which copies the classification will be installed in conjunction with the backup changes.

E. Reloader

No compromise threat exists during a complete reload since no users are logged in. It is only necessary to insure that the hierarchy is security consistent at the end of the reload (even when parts of the dump tapes were garbaged and not reloaded.) Since several processes can perform a reload simultaneously, only the last one to finish is required to perform a consistency check. This can be implemented by introducing a new branch item, "security_oos" (security out-of-service), which is set in every reloaded directory. After the reload finishes, the consistency check will be accomplished by a procedure which scans the reloaded subtree, insures that all upgraded directories have a terminal quota, and turns off the "security_oos" bit. This procedure will be run by the system security officer.

F. Retriever

The retriever, being on-line, has several security problems to handle.

1) It must be able to create directories at many different classifications.
2) It must insure the terminal quota status for upgraded directories.
3) It must handle the case of missing intermediary directories without knowing the classifications of these directories.

Whenever the retriever restores a directory that is at a higher classification than that of the nearest on-line parent, all the intermediary directories are created by "build_tree" at the level of the on-line parent. These directories are marked "security_oos" (by setting the bit in the branch) to prevent any compromise due to an error in the choice of the classification. If at some later time (on the next tape) the retriever finds that an intermediary directory should really be at a higher classification, it raises the classification. This may involve increasing the classifications of several inferior intermediary directories.

A new procedure that upgrades subtrees will be used. A subtree upgrade could fail if an inferior segment (which implicitly defines the classification of its parent directory) or an inferior directory had been previously reloaded at a lower classification. To implement "security_oos", a new entry to create a directory with this bit on will be added to "append." A modification to "find_" will prevent any process that doesn't have the special skip security bits on from making that directory (as well as the inferior subtree) known. At the completion of the retrieval requests, a procedure which scans the subtrees retrieved will be invoked. It will set default terminal quotas of 1 on upgraded directories if necessary, and will turn off the out-of-service bits. This is the same procedure as used after a reload. If Multics crashes during a retrieval, the reloaded subtree can only be placed into service by running the consistency procedure. Since only the system security officer is allowed to run this procedure, he is responsible for validating the classification of all intermediary directories.

G. Salvager

The salvager will require some modifications to preserve classifications and to insure that the downward increasing classification rule and the terminal quota for upgraded directories rule are observed. If a security error is found then the salvager will mark the entire subtree "security_oos" and record this on the operators console and/or audit log. This mechanism is preferable to one where a hardware failure might cause the entire hierarchy to be overclassified if the salvager automatically increased the classification to the consistent value. The security officer will be required to check and correct all classification values before placing the branch back into service. Forced resetting of classification, along with the setting of "security_oos," will be provided for the system security officer. But "security_oos" can only be turned off by executing the consistency checking procedure.

Startup of Security

Users of an installation at a single classification will be completely oblivious to the storage system changes described above. Even at installations allowing use of several classifications and starting with no user files, no effects on user utility of the system appear. This is not the case with the Pentagon installation. Here, Multics will have been in operation at an implicit level of top secret for a year. Two problems will occur after the security installation. The first is the necessity to classify all existing work at the correct level/category values. It has been suggested that current users organize their files in a manner which will allow procedures to perform the actual upgrading. It is expected that Honeywell will help in this task. A special session will be used to allow the setting of classifications with access mode security disabled. (This could be done by having a system without the security version of "access_mode" on the system tape.) Retrieval requests spanning the turn-on date require special handling because no classification was recorded. The security officer must verify all requests in order to validate that an old top secret segment (implicit from AF Multics operation today) is not reloaded into a directory that is now unclassified.

At new Multics installations this problem does not occur. Security is simply "turned on" by allowing users to log in at classifications other than zero, and making "upgrade_directory" available to all users.

Utility Implications

The second problem for the Air Force, and for all sites using security, is the more permanent one dealing with system utility. Since users will be able to choose their clearance (up to a limit, of course) at login time, they may not be able to read their mail and messages or successfully execute the start_up exec com. In general, there will be a sudden decrease of user sharability as well as access glitches. For example, any procedure that uses the access mode bits returned by "status_" as if they indicated the effective mode may no longer work correctly. Although not necessary for this project, "status_" and "star_" could be modified to return the effective mode (including rings) rather than the raw mode. This will make use of multiple rings easier as well, but costs in execution time. A multi-level security system essentially compartmentalizes the user community based on category sets and clearance levels.

Summary

Segments and directories will contain two additional branch attributes: level and category. These will be compared to the values stored in the PDS and the raw access mode will be modified to take into account the security relationship.

Only directories can have a classification higher than their parent. Such directories are termed "upgraded directories" and must have terminal quotas. The branch of an upgraded directory is protected at the level of the parent.

A new security out-of-service bit is introduced to be used by the salvager and retriever. The retriever uses this during the processing of requests to prevent incomplete data from presenting compromise windows. The salvager uses the bit to indicate security inconsistencies and these must be corrected on-line by the security officer before placing the directory and subtree back into service.

Although there will be special flags that indicate security checking should be suspended for processes such as the initializer, current Multics access control is unaltered and is never suspended. Users of an installation that has everything classified at one value will see no change in Multics utility or functionality. The only (beneficial) change introduced is that a recreated subtree is out-of-service during a retrieval.