MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Reply to: Project MAC
545 Technology Square
Cambridge, Mass. 02139

617 253-6011

To: Bisbey, R.
Burner, W.
Clark, D.
Clingen, C.
Daley, R.
Feiertag, R.
Gintell, J.
Greenberg, B.
Hunt, D.
Redell, D.
Roach, R.
Saltzer, J.
Schell, R.
Schroeder, M.
Spitzer, R.
Van Vleck, T.
Vinograd, D.
Webber, S.
Whitmore, J.

From: H. J. Goldberg

Date: July 23, 1975

Subject: Revision 7 of the list of Multics security holes.

Enclosed is the seventh revision of the list of known Multics security holes. This list contains all security problems which have been reported and verified as of July 12, 1975, and have not yet been fixed by installation in the M.I.T. standard system (system 25.10a).

As with previous editions of the list, no security holes yet encountered represent fundamental design errors in Multics--they are all relatively simple implementation goofs or localized design slip-ups which are correctable without revision of any basic assumptions.

As Multics becomes a production system at several different sites, it is more necessary to exercise some care in the distribution of this document. If the information contained here is controlled, it will remain possible to fix the holes described on a scheduled basis, rather than as a crash project. For this reason, your copy has been personally labeled, and I request that you please take some care to protect it and that you not reproduce it. On the other hand, it is important that legitimate users of this information not be denied access. If there is a need for additional copies, please contact me so that I may arrange the preparation of properly labeled copies for the additional recipients. Thank you for your cooperation.

Copy 1 -- J. H. Saltzer

Multics Security Holes List, Revision 7
July 1975
Compiled by H. J. Goldberg

This list of Multics security holes is revision 7, replacing revision 6 of February 22, 1974. This list is constructed from some of the entries in the previous list as well as from some more recently reported security holes. The entries in the first part of this list correspond with those in the previous list, except in those instances where a hole has been properly plugged. Those holes which have been fixed will be described separately, in upcoming RFC's.

The description of each security hole is divided into four parts. First, the problem is categorized according to the consequences of its occurrence: it may result in unauthorized disclosure or modification of information, or denial of service (with the extreme case being a system crash). The second and third parts are are a description of the problem, followed by suggested fixes. The fourth part is the current status of the problem. In the case of security holes for which fixes have been proposed, references are made to any design or installation documents (RFC's, MTB's, or MCR's) which are applicable.

ITT overflow

Categories: This is a denial-of-service problem.

Description: Repeated calls to hcs_$wakeup can cause messages to pile up in the ITT, eventually filling it and causing system messages to be lost, and a system crash, denying service to legitimate users.

Remedy: A quick fix would be to place a limit on the number of ITT entries which may be queued for a single process. A longer-range fix would be to transmit user-originated messages directly from one user to another, eliminating the need for the ITT.

Status: There are two ring 0 procedures that check the number of messages in the ITT in response to each call to hcs_$wakeup. First, hardcore IPC ensures that the number of messages stored in the ITT by user-ring wakeups does not exceed a threshhold value. (The threshhold is the ITT size minus the number of APT entries, allowing one device wakeup entry per process.) If the ITT is filled to the threshhold, then a call to hcs_$wakeup will generate an "ITT overflow" message, together with the process group id, on the operator's console. A second check is made in the traffic controller -- if the ITT ever overflows, the traffic controller will loop, causing a system crash. To prevent the ITT

from becoming clogged with unremovable entries, the traffic controller rejects wakeups sent to the idle process. System crashes due to repeated hcs_$wakeup calls are prevented by the threshhold check. In summary, several small fixes to cover the most frequently occurring accidents have been made, but it is still possible for a determined attacker to exploit systematically the basic vulnerability.

Reused address or device mixup

Categories: An occurrence can result in the unauthorized disclosure of information.

Description: Whenever a device address is accidentally reused or modified, as may happen when the system crashes in the midst of updating the value of a device address in a page map, the system may end up with a segment containing a page belonging to another segment. If there are two or more segments claiming the same page, the salvager can detect the inconsistency and destroy the page, since it does not know which segment ought to contain the page. In all other cases the problem is undetected.

Remedy: The storage representation of a page should be augmented to contain not just the page contents, but also the unique identifier of the segment to which the page is assigned. Then, following a system crash, the salvager can cross check between device addresses as recorded in the file maps and segment identifiers as recorded with the page contents.

Status: The subset of the "reused address" problems which can be detected by the salvager is now treated according to the suggested fix: pages with reused addresses are zeroed. There are classes of reused address problems which the salvager cannot detect. If the rightful owner of a page deletes the segment containing it, a duplicate address of that page can no longer be detected by the salvager. In addition, swapped or permuted page addresses are undetected by the salvager. Future system releases will contain additional consistency and redundancy checks to minimize the possibility of improperly assigned secondary storage addresses. In particular the New Storage System for Multics will update the file map only once, at segment de-activation time, thus narrowing the window in which a problem could arise.

Accounting system hierarchy scan

Categories: This may result in unauthorized disclosure of information.

Description: The accounting system, in order to charge for secondary storage use, must read every system directory, since page-second storage counts are stored in the directories. Thus

all system directories and files must be made accessible to the operator of the accounting system.

Remedy: Revise directories to contain account numbers, and store usage information in separate accounting files, which can be accessible to the operator of the accounting system.

Status: A modification to the accounting system, similar to the suggested fix, is being designed. It should be noted that although all segments are currently accessible to the account manager's process, the process makes use of a special process overseer which restricts the activity of the manager to appropriate accounting functions.

Absentee/Daemon overload

Categories: This is a denial-of-service problem.

Description: Any user may submit an arbitrary number of absentee jobs or I/O daemon requests, effectively denying absentee or I/O daemon service to authorized users.

Remedy: Place an administrative limit on the number of outstanding absentee jobs and I/O daemon requests that a single user is allowed.

Status: This problem still exists.

IPC event channel loop bug

Categories: This is a denial-of-service problem.

Description: If an IPC message with an illegal event channel name is sent to another process, the other process will loop on an out of bounds error inside IPC. If sent to initializer, will cripple system.

Remedy: Add bounds check to IPC message handling procedure.

Status: This bug is still there. A user-ring IPC utility procedure, ipcprm $retrieve_channel, determines whether an ECT entry corresponding to a given event channel name exists. It uses part of the event channel name as the offset into the ECT, without checking to see if the offset is too large.

High speed line carrier detect problem

Categories: This may result in unauthorized modification and disclosure, as well as denial of service.

Description: When in output mode, the system cannot detect that the user has hung up his 1200 baud line, since there is no carrier detect feature on the 202C6 dataset. Another user can then dial in and continue to use the line, with access to the previous users' files.

Remedy: Obtain datasets providing a carrier detect feature, and add software to log out user on carrier failure, just as for low speed typewriter lines.

Status: The use of Vadic modems solve this problem, but currently Multics still supports the 202C6 modems which are used by the ARDS terminals.

Linker bug

Categories: This is a denial-of-service problem.

Description: Certain types of incorrect link definitions will cause the linker to go into a loop inside ring zero. This is a specific instance of a more general problem: the linker must interpret a user-constructed data base (a linkage section of an object segment) which may contain errors, malicious traps, or even be dynamically changing. The linker is thus exposed to a "battle of wits" with a systematic attacker; it is difficult to convince oneself that there is no way to exploit this vulnerability.

Remedy: The suggested remedies are (1) fix the linker to accept only valid definitions; (2) add a time out in ring zero to catch all such problems; and (3) move the linker out of ring zero.

Status: All known bugs of the sort described have been fixed. We cannot be sure that additional bugs of this sort will not be discovered, so removing the linker from ring zero, as described in RFC's 23 and 41, MTB 35, and in MAC TR-132 will isolate the effects of such bugs.

Mailbox is open bug

Categories: This may result in unauthorized disclosure or modification of information.

Description: Although we have begun to change over to using message segments for mailboxes, the old style mailboxes are still supported. Those who still maintain the old style mailbox (ring 4 segment) are still vulnerable to unauthorized release or modification of information as described in the previous holes list.

Remedy: Revise the mail command to use message segments only rather than a directly writeable mailbox.

Status: The revised mail command uses message segments if possible. The design is described in MTB 70. Currently the mail command and its associated commands and active functions will attempt to use the new ring 1 message segment mailbox if available. If it does not exist the older ring 4 message segment will be used. This "feature" is supposed to be phased out in the near future.

process id argument validation

Categories: The most likely consequence of exploitation of this security hole is denial of service, but unauthorized disclosure or modification of information may occur.

Description: The low-order bits of a process identifier are actually the offset of that process' entry in the Active Process Table. Entry point hcs_$wakeup does not verify that the process_id given as an argument is a legitimate value for a table offset. It does look for the process id at that location, but this check is not foolproof.

Remedy: The offset value obtained from the process identifier must be checked for legality.

Status: This situation still exists. This bug is similar to the IPC event channel loop bug.

Syserr masking too long

Categories: This may result in denial of service.

Description: When an error inside the system occurs, the syserr routine masks the CPU against interrupts while printing a message on the operator's console. If too many interrupts come in, IOM status queues will overflow, crashing the system. The user can trigger an apparent system error by calling hcs_$wakeup too much; he can then generate enough interrupts to crash the system by sending a stream of characters with incorrect parity.

Remedy: The syserr printing routine should be revised to permit interrupts to be handled normally during message printing.

Status: The syserr routine has been modified to manage its buffers under interrupt control, as documented in MTB 16 and MCR 80. Thus, in most circumstances, syserr will copy its message into a wired buffer and return, with the posting interrupt arriving later. However, if the wired buffer is full, then syserr may wait as before until it can do something with the

message. This is another example of a fix which takes care of the most common accidents but still permits systematic exploitation.

Call limiters on gates

Categories: This may result in unauthorized disclosure or modification of information, as well as denial of service.

Description: Although the 6180 hardware can interpret the call limiter field in a segment descriptor, and the storage system now pports the call limiter field as an attribute of a segment, the call limiters are still set in an ad-hoc manner.

Remedy: To facilitate obtaining the proper call limiter value for a gate segment, language translators and the binder need modification.

Status: MTB 187 proposes changes to object segment format and includes a small discussion on call limiters. Non hardcore gates do not yet have the call limiters set in the branches.

Bulk card input

Categories: This may result in unauthorized disclosure or modification of information, as well as denial of service.

Description: Card decks input to Multics are assumed unauthenticated as to the submitter; yet the submitter is allowed to specify that a link be created in the file system hierarchy. The contents of a card deck are brought into the Multics hierarchy, at the submitter's request, by a SysDaemon process. The Daemon process stores the information from the card deck in a uniquely named segment in >daemon_dir_dir>cards. It then creates a link in a directory, through which the contents of the deck can be referenced. Both the name of the link and the name of the directory are specified by the submitter of the deck. A penetrator can place the link in any directory accessible to the SysDaemon process, making subversion of other processes possible. One possibility is to place a link called "start_up.ec" in the home directory of a user not already having such a segment.

Remedy: A solution to this problem is not to create a link, but rather to keep the deck in a user accesible place with a known name and appropriate access.

Status: This difficulty still exists. However, the proposed new card input facility, in keeping with the Access Isolation Mechanism (AIM) standards, maintains card pools of the different access classes to hold read in card decks which can be copied by the user. The reader is referred to MTB 143 for details of the

proposal.

Validating pointers from outer rings

Categories: This problem may result in unauthorized disclosure or modification of information.

Description: The code generated for manipulating pointers by both version 1 PL/1 and early revisions of version 2 PL/1 used the instruction sequence "ldaq, staq," in some instances. If any hardcore procedure has been compiled using one of these earlier compilers, then it may not be validating pointers correctly. A pointer in an argument list from an outer ring, for example, is copied by these instructions without any interpretation of the ring field. After a "ldaq, staq" copy, the ring field of the pointer may not represent the highest ring which could have influenced the value of the pointer; defeating the hardware validation mechanism. Thus the caller can associate a more privileged ring number with the pointer.

The current PL/I compiler generates improper code for copying pointers, thus bypassing validation of arguments in the following cases:
(a) Any pointer in a structure, if the entire structure is being copied.
(b) A pointer array, if the entire array is being copied.
(c) A pointer in an area, if the area is being copied.
(d) Any of the above where the pointer is a format, file, entry, or lablel variable.
The problem here lies in the fact that the compiler optimizes such moves by use of EIS move instructions and not the appropriate "epp" and "spri" instructions for the pointers.
These comments apply to all inward calls.

Remedy: More recent versions of the compiler use the correct instructions ("epp--" and "spri--") for copying pointers. These instructions make use of the ring validation hardware. The best insurance against discovering additional validation bugs caused by this problem is to verify that all ring zero and ring one PL/1 procedures have been compiled by a recent version of the compiler. This does not fix the second set of problems where it is obvious that appropriate code should be generated by the compiler. As a temporary measure one could include single element copying for pointers after a structure has been copied, or ban structure copying from the language.

Status: Presently, some of the ring zero procedures are still compiled in version one. This situation would be acceptable only if an audit were performed, verifying that the remaining "old-style" procedures never made use of pointers passed in from the user ring. A system imposed (but not enforced) rule could be established that programmers avoid using the above mentioned

constructs. Such a proposal has been made and is part of MTB [198].

AST thrashing

Categories: This may result in denial of service.

Description: A malicious user can degrade system service by forcing the system to activate and deactivate entries in the Active Segment Table (AST) at a high rate. This AST "thrashing" can be caused by requiring a large number of directories to be active. A number of parallel chains of directories can be created, rooted in the user's home directory. The lowest directory in each chain can be linked together by links. Then performing a "status" witn the chase option enabled will require that all directory chains be activated. With the current directory depth limit of 16 and link-chasing limit of 10, a single status request can activate about 130 directories. On the MIT system, four cooperating users could effectively tie up the 4-page AST pool, which currently has 528 entries. One user of the MIT system, if willing to create more than 64 directories larger than 16K (so that they require 64K page tables), can crash the system with this technique since there are 64 of the 64K AST entries at the present time.

Remedy: The system should have a graceful way of aborting such resource-consuming requests. For example, if a substantial number of AST entries were requested by a single process invoking a hardcore primitive, the primitive could abort the operation and return an error code, or terminate the process.

Status: This situation still exists.

del dir tree race

Categories: This may result in denial of service.

Description: The hcs_ entry point, del_dir_tree, can be used to delete a subtree of the hierarchy by specifying the root node. If several other users are cooperating and adding new branches to this subtree, they may be able to cause the deleting process to remain in ring zero for an indefinitely long time.

Remedy: A solution to this problem is to remove the del_dir_tree entry from ring zero. The "walk subtree" part of the del_dir_tree function can be done in the user ring.

Status: This situation still exists.

Experiments on the KST

Categories: By performing experiments on the Known Segment Table (KST), a user can discover the existence of directory trees to which he does not have access.

Description: If a penetrator suspects that there is a directory >A>B>C, for example, he can discover this by giving the command "initiate >A>B>C>D." If the directory >A>B>C exists, then entries for >A, >A>B, and >A>B>C will be constructed in the KST, regardless of whether or not D exists. The penetrator can initiate one of his own segments before the experiment and another one after, bracketing the segment numbers assigned to the directories. If there is a gap between the segment numbers assigned to the user's bracketing segments, then some of the directories in the pathname exist. Moreover, by using the "list reference names" command, the user can display the reference names of the intervening directories.

Remedy: One solution to this problem is to have a more effective cleanup strategy in handling the KST. For example, the initiate primitive could clean up after itself in the case that the user does not have access to the entry specified in the pathname. Another approach is to make use of per-ring segment numbers; for instance all segments initiated in ring 1 (as well as all superior directories) would have segment numbers from 1000 to 1777, and so forth.

Status: This situation still exists. A proposed per ring name space manager solves this problem by allowing the user to get a pointer to directories which may not exist, and still return the error code. The user will not know, by looking at the KST, whether the name was put there because it really existed or because he was not allowed to know that it did not exist. The reader is referred to MTB 154 for further details. A MAC TR is expected to be published on the proposal.

Lost "hangups" in the DN355

Categories: This may result in unauthorized modification and/or release of information, as well as denial of service.

Description: Under certain circumstances, apparently on a heavily loaded system, a user may find his terminal attached to someone else's process after dialing the system. This problem may be due to a synchronization difficulty. Real-time events (e.g. interrupts from a given line) handled by the hardcore teletype DIM, and scheduled events (e.g. maintaining the state of a line in a table) handled by the answering service may not be occurring in the expected sequence. It may also be possible that the DN355 loses hangup signals. Thus the next user to dial into that tty channel will find himself connected to the previous

users process. The results are identical to the High Speed line carrier detect problem, but this is apparently a software error.

Remedy: The problem is probably due to unexpected timing errors. One possiblity for correction is a simplification of the tty code and careful auditing with the problem in mind.

Status: A new tty package is currently being tested prior to installation in the standard system. It is hoped that the restructuring and some simplification will aid in diagnosing the problem should it occur in that system.

SCU Data Validation

Categories: Incorrect SCU restarting may result in unauthorized modification and/or release of information, as well as denial of service.

Description: The problem lies in the method used for restarting after a fault that occurred in an outer ring. The method employed is a call to the procedure "return_to_ring_zero_" with new machine conditions (SCU data). "restart_fault_" is then called which must then validate the SCU data presented to it. The machine conditions consists of 288 bits of information which must be carefully looked at to insure that a user is not restarting machine conditions which would cause the CPU to allow unauthorized access to information or programs. To help in this effort a copy of the machine conditions are kept in ring 0 in the PDS. Thus restart_fault may tell which bits have been changed by the user, enabling checking to be easier. Two problems arise from this situation. A programming error which allows illegal machine conditions to be restarted due to the method of signalling the illegal_restart condition is currently in the system. Secondly the fact that machine conditions are kept in ring 0 draws our attention to possible overflow of the area where the machine conditions are kept, the PDS, or ring 0 stack. By appropriately filling up the area then calling a ring 0 procedure or creating some other invocation of a ring 0 program such as through a fault, the PDS will overflow with possible disastrous results such as the system crashing. One can also take advantage of this problem by similary filling up the area then calling a lower ring (not necessarily ring 0) procedure. The lower ring then proceeds to lock locks and the like and would eventually call some ring 0 primitive which would overflow the PDS. This causes termination of the process without the chance for the lower ring to clean up before terminating.

Remedy: What is needed is a cleaner CPU design with a simpler restarting operation.

Status: As of this writing the bug is in the system. The restart_fault software is new and if it could be made to work

correctly will solve many older SCU validation problems.

Special Casing Ring 0 Unnecessarily

Categories: These problems can result in denial of service.

Description: There are 2 known problems of special casing ring 0 now:

(a) Preempt and IPS interrupts are deferred in ring 0. Thus by forcing one's process to remain in ring 0, the processor will not be given up thus causing a denial of service. This can be accomplished by constructing an indirect chain for an argument to ring 0 which requires more pages than core can hold (one ITS pointer in a page will suffice). This will cause looping in ring 0, without taking a lockup fault, due to the interspersed page faults.

(b) Since record quota overflow is ignored in ring 0, one may cause the supervisor to assign an unlimited amount of quota to any directory. The procedure for doing so automatically is as follows:
Upon detection of a record quota overflow, simply call some inner ring procedure to touch the page in question. The fault will occur in ring 0 and will be ignored. One could potentially use up all disk records in this manner, thus causing a system crash when the system attempts to allocate a record with no disk space.

Remedy: Ring 0 should not be blindly considered exempt from all interrupts at all times. If the integrity of a database is at stake, systematic calls to "defer_interrupts_" and "accept_interrupts_" could be made. Conditions such as record quota overflow should be honored in most cases.

EIS Zero Length String Bugs

Categories: This problem can cause denial of service.

Description: The PL/I compiler will generate zero length string moves for various reasons. Taken by themselves, these moves produce no problem but unfortunately the hardware faults on such operations. If such a fault should occur while a procedure is using the PRDS, the system will crash. A second but related problem is the attempted fix for outer ring programs. There exists the "PL/I machine mode" switch which is used by the fault handler to determine what to do when receiving a "zero length move" fault. If the switch is on the fault is ignored, if it is off the fault will be signalled. This switch setting affects all rings, however, enabling an outer ring procedure to affect an inner ring by the setting of this switch. This could cause an

unexpected fault to be signalled in the inner ring.

Remedy: The obvious remedy is to fix the hardware, but a per ring switch could be used in the interim.

Status: Attempts have been made to fix the hardware resulting in large field installed wiring changes. As of yet these changes have not completely eliminated the problem.

Automatic System Crashing Bug

Categories: This is a denial of service bug.

Description: Both the Fault Interceptor Module (FIM) and the Interrupt Interceptor (II) check the segment number of Sp at the time of a fault or interrupt. If Sp points to the PRDS and the fault was taken in ring 0, the system is made to crash. The operation of the Call instruction only sets Sb, thus if the gate takes a fault before touching Sp, and a user has set Sp to point to the PRDS, the system will crash.

Remedy: Check the validation level of Sp before testing its value. If the validation level is not 0 ignore the value.

Status: The suggested fix has been implemented and has been submitted for installation.

Traffic Controller Argument Copy Bug

Categories: This problem can lead to denial of service.

Description: The traffic controller (PXSS) has to copy its arguments into a place where it will not fault when using them later. Therefore it copies its arguments into "pds_$arg_temp", a temporarily wired segment. The problem arises when PXSS takes a fault while it is copying its arguments (such as a page fault) which causes another call to PXSS. This new invocation similarly attempts to copy its arguments, overlaying the previous invocation's arguments. It has been suggested that this is the cause of the "lost wakeup" symptom, occasionally appearing.

Remedy: When PXSS finishes copying the arguments it should check to see if any faults had been taken in the interim. If so, an attempt should be made to re-copy them again. A retry count could be maintained to prevent infinite looping.

Status: A new version of PXSS has been submitted, for auditing by Honeywell.

Bypassing the Linker Bug

Categories: This could result in denial of service.

Description: One may fabricate a pointer through which to do a "callsp", for a gate entry, thereby transferring to the gate without having its linkage combined by the linker (a linkage fault would not be taken). This would cause a "reference through null pointer" fault when the inner ring attempted to reference its linkage section through the lot. This would be most harmful if the inner ring had locked a lock prior to taking this fault.

Remedy: A quick solution would require all gates to check that their linkage sections had been combined by checking the value of "Lp" they pick up from the "lot".

Status: This problem still exists but a proposed fault as explained in references for fixes to the above mentioned "linker bug", would cause invocation of the per ring linker, thus establishing the correct environment for the inner ring. This fault would effectively take place at the time the inner ring picks up its linkage pointer from the lot. This fault has come to be known as the "ISOT" fault.

Multiple reference to Arguments Bug

Categories: This problem can cause unauthorized release and/or modification of information as well as denial of service.

Description: The idea behind the problem is twofold. The protected procedure must reference its arguments more than once and the argument pointer must be changed between these references. On Multics the latter can be done by using the tally feature of indirect pointers. Now all one must do is to find a procedure which acts along the following lines: An argument is picked up and verified for correctness. Later the argument is picked up again and used. This seemingly harmless act results in disaster when the argument is a process id of the process to stop (and destroy).

Remedy: A simple remedy is to have procedures copy their arguments into automatic storage and use it there.

Status: Although the problem has been pointed out, all places where a problem could exist have not been fixed yet. The reader is referred to RFC 59 for further details. A copy of the summary of results presented there is included here for completeness.

Number of entry points examined in hcs_	170
Number of entry points with multiple references	51
Classification of multiple references:	
Type 1 -- Probably O.K.	23
Type 2 -- Fragile, but probably O.K.	8
Type 3 -- Don't know, lack of information	3
Type 4 -- Hole without obvious exploitation	8
Type 5 -- Hole with known exploitation	9
Untested entry points	3

Bad Ring Bracket Checks

This section simply lists bugs that fall under the category of bad, or no, ring bracket checking.

(1) "chname" does not check the extended ring brackets of a directory whose name is to be changed. This allows a caller to change the name of a directory in a lower ring.

(2) When the branch entries of "append" create a directory branch, they automatically assign extended ring brackets of 7,7 rather than allowing the caller to specify them.

(3) All entries in "set" except "set$bc" and "set$bc_seg", do not compare the write ring of the segment whose directory entry is to be modified, to the current validation level. This allows modification of a directory entry of an inner ring segment.

(4) Several entries in "acl" allow a caller to set ring brackets to 4,4,4 even if the current validation level is greater than 4. An internal procedure of "acl", "check_rb", compares proposed ring brackets to 4 rather than to the current validation level.

Verifying Existence of Directories and Segments

This section, similar to the previous section, lists those bugs which allow a user who has no access, to find out the existence of directories and segments.

(1) It is possible to use the branch entries of "append", callable through "hcs_", to verify the existence of entries in directories to which the caller caller has no access, because a name duplication error can be returned even though the caller does not have access to the parent directory.

(2) Callers of gate entries that call a "find_" entry and a "dir_control_error" entry can utilize an inconsistancy in error codes returned by these procedures to verify the existence of a segment under a directory to which the caller has no access.

(3) Through use of a design error, it it possible to verify the existence of a directory in which a link exists that points to a segment that one has access to. This is due to the decision that access to an object is to be associated with that object, not with "indirect pointers" to it. Thus one could initiate the object pointed to by the link, by using the "secret" directory as part of the pathname. If the intiation was sucessful, it is known that the directory exists.

Potential Problems

This section lists known bugs that may or may not be exploitable but are considered potential problems.

(1) Several gates call procedures or entries that do not exist.

absentee_test_$destroy_proc calls deact_proc$destroy_proc

hphcs_$quota_get_reset calls quota$gget_reset

hphcs_$set_dir_ring_brackets calls level_0_$set_dir_ring_brackets

tape_admin_$delete_bvds calls tape_path_name_$delete_bvds

admin_gate_$tdcm_attach calls tdcm$tdcm_attach

admin_gate_$tdcm_mount_bit_set calls tdcm$tape_mount_bit_set

admin_gate_$tdcm_promote calls tdcm$tdcm_promote

system_control_ calls reconfigure$del270
erf
reconfigure$add270

mseg_vl_temps_$vl_check_salv_bit_ptr calls
queue_mseg_$q_check_salv_bit_ptr

(2) There exist several gate entries which are exact duplicates of other gate entries. Although this situation is not a problem, removal of the procedure at a later time may result in problem (1) above.

hcs_$append_link and hcs_$appendl call append$link

hcs_$chname and hcs_$chname_file call chname$cfile

hcs_$ex_acl_delete and admin_gate_$ex_acl_delete
call ex_acl$adelete

hcs_$ex_acl_list and admin_gate_$ex_acl_list call ex_acl$alist

```
hcs_$ex_acl_replace and admin_gate_$ex_acl_replace
                    call ex_acl$areplace

hcs_$fs_get_call_name and hcs_$fs_get_ref_name
                    call fs_get$call_name

admin_gate_$ips_wakeup and hphcs_$ips_wakeup
                    call pxss$ips_wakeup

admin_gate_$admin_ring_zero_peek and phcs_$ring_0_peek
                    call ring_0_peek$ring_0_peek

hcs_$stop_process and hphcs_$stop_process
                    call stop_process_$stop_process

hcs_$try_to_unlock_lock and admin_gate_$try_to_unlock_lock
                    call try_to_unlock_lock$try_to_unlock_lock

hcs_$cpu_time_and_paging_ and hcs_$get_usage_values
                    call vclock$cpu_time_and_paging_
```

(3) Several privileged gates do not check the number of arguments passed-- those using the "fgate" (fast gate) pseudo op in the macro assembler "mexp".

(4) "fs_search$get_rel_segment" references a parameter "a_bit_count" which is not defined for this entry.

(5) "hcs_$get_seg_count" specifies one too few parameters to be checked by this gate.

(6) In "quota$qset", "sum$dirmod" is called twice for the same directory.

(7) The entry point in "metering_ring_zero_peek_" is defined with an "entry" instead of a "segdef" pseudo op. This would partially defeat any call limiter which may at some time be specified.

(8) Although hardly feasible, one could potentially cause overflowing of inner rings linkage section by invoking the linker on all possible entries. One may use y=addr(ring_1_segment_entry) to do so without actually calling the procedure.

(9) The current PL/I compiler can generate code that will take a lockup fault. Currently the CPU operation for taking interrupts causes interrupts not to be sampled after a trasfer instruction, nor after execution of the even instruction of an even/odd pair. The unwary programmer could program a loop, or a series of "if" statements that could conceivably result in code that when executed, would take a lockup fault. This code running on the

PRDS will cause a system crash if the lockup fault is taken at that time.

(10) "initiate_search_rules" does not take an error exit if the search rule number fails a bounds check.

(11) As discussed above in SCU Data Validation, there exists the general problem of validating machine conditions before restarting. It appears that this always results in a "battle of wits" situation as long as the current hardware is used.

(12) All inner ring programmers are faced with the problem of correctly passing results back to the outer ring. On input, arguments are first copied and only the local copies are used. This prevents any problems when referencing those arguments during critical pieces of code. It it likewise important to reference output arguments only after critical sections of code have been executed (i.e. after all locks have been unlocked). In the case where a user passes an area as an argument to ring 0 (such as star_), ring 0 must be prepared for all eventualities that may arise when outputting results in this area, including an area where the area package may not work properly (possibly because a user has deliberately destroyed area information).

Outright Bugs

This section lists those bugs which are simply programming errors, but result in improper operation or even a system crash.

(1) Procedure "try_to_unlock_lock$check" does not only check the lock, but attempts to unlock it.

(2) The initial acl handling of "asd_" does not check that "a_dirname" concatenated with "a_ename" is less than or equal to 168 characaters. This may result in setting an acl on the wrong directory.

(3) Calling "hcs_$status_minf" with a directory name of ">" and an entry name of "" will cause a system crash due to bad cleanup handling that causes an attempt to unlock an unlocked lock.

(4) "fs_get$pathname" does not initialize "kstp" (a pointer) before using it. This may result in a ring 0 loop.

(5) "link_man$get_count_link" uses "make_seg" to create combined linkage segments in the process directory. This could result in picking up an existing segment with incorrect access or ring brackets. The proposed linker mentioned in a previous section under "linker bug" would fix this problem.

Traffic analysis experiments

This final section deals with a class of problems, rather than a specific one. It is often possible for a penetrator to obtain useful information by observing the level of activity of another user's process. There are some sorts of activity which others cannot observe, such as the rate of transmission to or from the terminal attached to a process. Some other indicators of activity are readily accessible, such as the information displayed in the "who table:" all "listed" users have corresponding entries in the who table whenever they are logged in. (A user can be "unlisted", but this is not the default case.) Two cooperating users can send information based upon the presence of who table entries.

Information can be obtained by the receiving process whether or not the sending process is cooperating. In a military system, where the normal information paths between processes of different levels are constrained, information may still be transmitted using traffic analysis techniques. If the process which is the subject of the experiment is not cooperating, but is merely being observed, then the information obtained by traffic analysis may contain more noise. Nonetheless, it may be sufficient: if the penetrator knows a way to crash the system, he may choose to do so only when a certain user name appears in the who table.

A more informative sort of traffic analysis experiment is to observe the number of records of storage used by another user, by examining quota. Storage usage is reflected up to the lowest containing directory with nonterminal quota; if the penetrator has access to that containing directory, he can observe storage usage. To obtain information with the least noise, the penetrator would perform the experiment at a time when the observed user was the only one operating within the terminal quota subtree. Note that separate accounting segments, as proposed for the accounting system hierarchy scan problem, would help in this case.

In light of the Access Isolation Mechanism (AIM) standards being implemented on Multics "sneaky signalling" through use of quota remains an unsolved problem. The AIM system attempts to compartmentalize users into "categories" such as army classification levels of "secret", "top secret", etc. Thus directories become classified accordingly. It was evident that a directory of lower classification could not be placed in a directory of higher classification or else one would have to look in the more classified parent directory for information about the child directory. This goes against the entire classification system. Thus only directories of equal or higher classification may be placed in any directory. The outstanding problem is signalling from the user of a more classified directory to a user of the less classified parent directory by the use of the quota mechanism. In the current system quota is reflected up the

heirarchy until a directory with terminal quota is met. Thus top secret information can be transmitted to a less classified user. As of yet this problem is not solved , and may not be solved unless one begins implementation of a system as described by L. Rotenberg in his thesis "Making Computers Keep Secrets."

Other sorts of traffic analysis experiments include the observation of segment number usage, paging rate, and response time. In each of these three cases, an estimate of the level of activity of another process is made by measuring the performance of one's own process; consequently the noise in these measurements increases with the number of active processes.

These are examples of a class of problems which need to be considered if information paths between processes are to be controlled. Information paths of sufficiently low bandwidth might be considered acceptable.