MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

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

(617) 253-6016

To:
Bisbey, R.
Burner, W.
Clark, D.
Clingen, C.
Daley, R.
Feiertag, R.
Gintell, J.
Hunt, D.
Jordan, D.
Roach, R.
Schell, R.
Schroeder, M.
Spitzer, R.
Thomas, E.
Van Vleck, T.
Vinograd, D.
Voydock, V.
Webber, S.
Whitmore, J.

From: J. H. Saltzer

Date: February 25, 1974

Subject: Revision 6 of the list of Multics security holes.

Enclosed is the sixth revision of the list of known Multics security holes. This list was compiled and checked by Doug Hunt, and contains all security problems which have been reported to him (or me) and verified as of February 25, 1974, and have not yet been fixed by installation in the M.I.T. standard system (system 23.4) as of that date.

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.

Multics Security Holes List, Revision 6

February 22, 1974

Compiled by D. H. Hunt

This list of Multics security holes is revision 6, replacing revision 5 of February 9, 1973. 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 are described separately, in RFC's 46 and 47.

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 lifted directly from the previous list. They are a description of the problem, followed by suggested fixes. The fourth part is the current status of the problem. In the case of fixed security holes, 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 which 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.

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.

Metering gates expose everything

Categories: This may result in unauthorized disclosure of information.

Description: The entry point used for reading out system performance meters allows readout of any ring zero program or data base including, for example, typewriter input buffers, which may contain typed passwords.

Remedy: What is needed is a separate entry point which reads out only legitimate system performance meters. General peek gate then does not require such a long list of persons authorized to use it.

Status: A fix for this difficulty, proposed in MTB's 11 and 30, is being implemented.

IPC event channel loop bug

Categories: This is a denial-of-sevice 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 may solve this problem.

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 diffucult 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 and in MTB 35, will isolate the effects of such bugs.

Mailbox is open bug

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

Description: The current mail command implementation requires that permission to send mail to a user must be coupled with permission to read and delete any mail in the user's mailbox.

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

Status: The revised mail command will use message segments. The design is described in MTB 6.

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.

GIM data base bug

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

Description: The GIM creates a data base by a call to "makeseg" rather than append branch. As a result, it will use any segment which is already around and which has the right name. The user can then overwrite the GIM data.

Remedy: Fix the GIM to call append branch rather than makeseg.

Status: The GIM still calls "makeseg," but after doing so, it replaces the segment ACL and ring brackets by ones of the form "rw user_process_id 0,0,0" so that in case there had been a segment of the same name, the user can no longer access it. This current strategy has a deficiency which leads to sabotage. If a malicious user creates a link from his process directory to some appropriate target segment, he can cause the GIM to replace the ACL and ring brackets on that segment, denying access to authorized users. Any privileged program creating per-process segments for a user in this manner could be exploited by a malicious user.

CPU hogging bug 3

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

Description: The entry point hcs_$list_connect of the Gioc Interface Module (GIM) will, if called with the proper argument, set the "interaction switch" on, giving the user credit for an interaction the next time he calls block. By calling every few seconds, one can stay forever in the highest scheduling queue, and completely deny service to lower queue users.

Remedy: Review the PIM design (GIM replacement) to make sure it does not have the same hole.

Status: The entry "list_connect" has been removed from hcs_ and added to phcs_. This implies that the use of the GIM is temporarily restricted to require system programming privilege.

Call limiters on gates

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

Description: Although the 6180 hardware can interpret the call limiter field in a segment descriptor, the storage system does not recognize the call limiter field as an attribute of a branch. Consequently, it is not possible for a user to set the call limiter on any gate.

Remedy: The storage system must be extended to recognize the call limiter field as another attribute of a branch. To facilitate obtaining the proper call limiter value for a gate segment, language translators and the binder may need modification as well.

Status: Call limiters are now being set on hardcore gates, including the procedure restart_fault. The call limiter field, however, is being set in an ad hoc fashion for hardcore gates, and this same procedure cannot be used for user-defined gates.

Bad ring brackets on gate linkage segments

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

Description: When linking to a new-style gate, the associated linkage segment, gate.link, is assigned incorrect ring brackets. For each gate segment referenced by a process, the linker will create an associated linkage segment and assign to it the same ring brackets which are on the gate segment. A malicious user can exploit this situation by executing a "callsp" instruction which references some offset in the segment gate.link, thereby transferring control to the target ring, but to a segment which is not intended to be executed.

Remedy: The call bracket for new-style gate linkage segments should be set equal to the execute bracket. Furthermore, execute permission should be removed since new-style gate linkage segments contain no executable instructions.

Status: This security hole will be corrected as part of the project of moving the linker out of ring zero. At present, it still exists.

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 notify the user by some other means -- such as sending mail -- of the name of the card-image segment.

Status: This difficulty still exists.

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. These comments apply to any 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.

Status: Presently, over 20% 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.

Eligibility hogging

Categories: This can result in denial of service.

Description: The tape DCM uses the "wait" traffic control primitive for events which may be of long duration. As a result, several conspiring users making a certain request of the tape DCM can tie up the available eligibility slots. Suppose there are N eligibility slots. User 1 calls the tape DCM, requesting a forward-space one file operation on a 2400-foot reel. (With careful planning, this can take about four minutes.) Users 2 through N+1 call into the DCM requesting the same operation. Processes 2 through N+1 will wait (call pxss$wait) until process 1 has completed its operation. In the meantime, all N available eligibility slots are occupied, since a waiting process does not lose eligibility.

Remedy: An assumption regarding the "wait" primitive is that it be used in conjunction with "system events:" those guaranteed to complete within a short length of time. For such events, retaining eligibility is reasonable; for longer events, it is not. A strategy of blocking in the user ring, such as used with the teletype DIM, would be more appropriate.

Status: This problem still exists.

Communications line takeover

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

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.

Remedy: The basic design of the interface between the hardcore teletype DIM and the answering service is thought to be sound. There is apparently an implementation error which results in a race condition, which should be found and fixed.

Status: This problem is still outstanding.

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" with 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. Note that this sort of sabotage can be performed only on a system with two or more processors.

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.

Audit trail tampering

This section deals with a secondary issue; it is a matter of concern only if an unauthorized user has been able to read or modify information stored in a segment. The date-time used (DTU) and date-time modified (DTM) attributes of a segment indicate approximately the last time that a segment has been used or modified, respectively. These attributes are maintained implicitly by the file system, so that accesses to segments generate a "DTU" or "DTM" audit trail. Although a penetrator, with some effort, may be able to modify DTU or DTM by patching, he can modify these attributes simply by calling the set_dates entry in hcs_, providing he has "modify" access to the containing directory. He can, for example, reset DTU and DTM to the values they had prior to the unauthorized access to the segment, thereby covering his tracks.

The presence of this entry in hcs_, while not a security hole, can aid a system penetration effort. Since the reason for the set_dates entry in hcs_ is to allow certain backup programs to be run in processes with less than SysDaemon privilege, the particular programs can be modified, and then the set_dates entry

can be removed from hcs_.

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.

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.