To: Distribution

From: Lee Scheffler

Date: 9/13/74

Subject: More on Protection Auditing

The design review of MTB-103 ("The Protection Audit Mechanisms for Multics") raised several issues that required further study. Most of the concerns expressed dealt with the performance impact of protection auditing, particularly in installations not using auditing and/or the access isolation mechanisms. This memo gives further details concerning the expected performance cost of auditing, and re-visits several other design issues.

Location, Size, and Execution Cost of Auditing Calls

Potentially auditable directory control events and faults (those which require a test to see if auditing is necessary) occur on the order of 5 times a second. Thus, is it important to minimize the number, code size, and execution cost of conditional sequences and calls for auditing in these areas. The following performance-sensitive modules will contain auditing calls in the initial implementation.

dir_control_error - All directory control programs (should) call here when a process has insufficient access to complete some operation. A 3-instruction conditional and a 14-instruction audit call will be inserted just before the return, to audit instances of denied access.

fim - Some slight restructuring and a conditional audit call are necessary to audit illegal procedure faults (not including legal EIS instructions or out-of-bounds) and access violation faults (not including ring alarms, out-of-bounds, or illegal ring order). Audit checking and calling sequence adds about 30 instructions to fim.

Current plans for a second-stage implementation will involve audit sequences in two additional performance-sensitive modules.

makeknown - when the proposed KST changes (MTB-104) are implemented, access information in the KST will make it possible to audit successful initiations of protected segments and directories (those with access class greater than

Multics project internal working documentation. Not to be reproduced or distributed outside the Multics project.

system-low) in makeKnown. At the point where a segment or directory is actually added to the KST, about 60 instructions of auditing code will be inserted. The auditing of successful making known of protected directories, normal protected segments, and "system" (ring 1 multi-class) segments will be governed by separate audit select flags. For segments with system-low access class, a 3-instruction test will skip the auditing code.

find_ - When the recursive making known of parent directories fails due to insufficient authorization to search further, it is desirable to log the full pathname of the segment. At present, find_ is not structured well enough to do this conveniently. A re-write of find_ will allow this type of auditing, and will probably result in a worthwhile improvement in system performance.

It is expected that, in a system not performing any auditing, the execution cost of the additional auditing checks will be around 20 instructions/second, quite negligible.

Performance Costs of Auditing

An actual auditing call represents somewhere between 500 and 1000 instructions, including the call to protection_audit_, the calls it makes to acquire all the information needed for the log, the call to syserr, and the later invocation of syserr_logger to copy the syserr wired buffer into a paged buffer. Thus, in normal system operation (i.e., no user purposely generating extra auditable protection events), auditing of all protection events for all processes can represent as much as about 5 milliseconds overhead per second of real time. This is a significant cost, but is not unreasonable if one bears in mind that only installations using auditing will pay this cost.

Given the cost of .5 - 1 millisecond per audit log entry, it is possible for a user to cause a great deal of system overhead by repeatedly causing auditable protection events. In installations wishing to detect "penetration" attempts, this overhead will certainly be justified. Future enhancements of protection auditing will probably include detection of significantly abnormal auditing frequency, probably with an exponentially-weighted per-process auditing frequency meter in the pds.

Message Segment Auditing

The new message segment design (see MTB-101) marks message segments as "multi-class" because they may contain messages of many access classes, and assigns the message segment itself a

class equal to the highest class of message it can contain. Because message segments are manipulated only by ring 1 primitives, and because the actual protected objects are the individual messages themselves, initiations of message segments are not very interesting security events. The eventual need is for some finer-grained auditing of the use of message segments. To prevent flooding of the syserr audit log with innocuous messages for dprint and absentee request submissions, a separate pds audit select flag will control the auditing of ring 1 multiple class segment initiations. makeKnown will check for this case when determining whether to audit the initiation of a protected segment. This allows initiations of normal protected segments to be audited without requiring "system segment" initiations to be audited.

Audit Selectivity Classes

Mainly for performance reasons, the audit selectivity classes defined in MTB-103 have been redefined. There are two groups of flags (in the left and right 18 bits of a word in the pds), one group for events audited directly in ring 0, and one group for events audited via a gate from ring 1.

ring 0 audit selectivity classes:

- initiations of protected non-directory segments (those with access class greater than system-low), not including ring 1 multi-class segments
- making known of protected directories (These are separated out because many directories may be made known as by-products of other operations. Knowledge of the making known of a protected directory is generally of less importance than the initiation of a protected non-directory segment.)
- initiations of ring 1 multi-class segments
- access denied to any segment or directory due to improper authorization, ACL mode, or ring validation level
- ring-related access violation faults (except ring alarms and illegal ring orders)
- ACL mode related access violation faults (except out-of-bounds)
- illegal procedure faults (except legal EIS faults and out-of-bounds)

- attempted IPC send-downs
- assignment of system privileges to a process (either blanket privileges, or privileged initiation of single segments)
- rejected device attachments (tape and disk drives)
- protection related actions of the SSA (performed by ring 0 primitives) such as segment downgrading and turning off security-out-of-service

ring 1 audit selectivity classes:

- rejected media (tape and disk) mounts
- message segment protection events (such as overflows)

Per-Segment Audit Selectivity

Being able to audit the use and attempted misuse of specific segments for all processes allows the use of selected protected information to be monitored without requiring the auditing of all use and attempted misuse of all protected information for all processes.

The initial design of this capability used a bit in the branch to cause auditing on all initiations, access denials, and access violation and illegal procedure faults. The cost of checking a bit in the branch during fault processing is far too high for the marginal value of auditing such faults, so no per-segment auditing will be done on faults.

As per-segment audit flags will change infrequently, and since such changes will not need to become effective immediately, this flag can be copied into the KST entry when the segment is made known, to save directory locking.

As per-segment audit selectivity is not an immediate necessity, the only change will be to reserve a bit in the branch and a bit in the KST.

Metering Plans

To provide concrete data on questions of protection auditing performance, some meters will be added:

faults - No one seems to have any good idea of the frequencies of various kinds of faults. A simple aos instruction in fim will provide a histogram of faults that reach fim. The histogram will go into the unused space in wired_hardcore_data. These meters

may not be permanent.

auditing_checks - Some temporary meters will be useful to determine how often audit checks are made. At each place where an audit select flag is checked in ring 0, a counter in active_hardcore_data will be updated. Since these meters will definitely reference an additional page each time an audit flag is checked, they will not be permanent.

auditing_time_and_paging - Some permanent meters maintained by protection_audit_ will record in active_hardcore_data the average time and page faults spent logging audit messages.

(END)