To: Distribution

From: Lee Scheffler

Date: July 25, 1974

Subject: The Protection Audit Mechanisms for Multics

This is one of a series of design memos describing enhancements to Multics access control, as outlined in MTB 047. This MTB uses the new terminology described in MTB 100.

I. Why Audit Mechanisms

In environments where access to information must be carefully controlled, it is important to keep a history of protection-related aspects of system operation. Records of events such as accesses granted and denied to protected information, denied uses of controlled information paths, and logins and logouts of users, form an "audit trail" that serves several useful purposes:

1. It allows a general review of the use of protection mechanisms.

2. It allows easy assessment of the extent of compromise or damage in instances of unauthorized disclosure or destruction of information.

3. The mere existence of a record of denied access deters attempts to bypass protection mechanisms.

For these reasons, particularly with the addition of new administrative access controls to Multics, we need to provide the capability of maintaining an audit trail of events relating to information protection. The mechanisms will use existing system logs to store character string audit messages for auditable protection events (significant instances of the use of protection mechanisms). Each time software detects an auditable event, an audit entry containing all relevant protection-related information is made immediately in the appropriate log. The logs are then available for perusal or analysis by log summary tools.

II. Objectives

The following objectives are necessary for the protection audit mechanisms, and are in keeping with the general Multics design philosophy.

1. Minimize the impact of audit mechanisms in installations where they will not be used. Only those who need to use the mechanisms should have to pay for them. Defaults should make the audit mechanisms invisible.

2. Minimize the number and size of audit messages. This implies providing a high degree of selectivity over which protection events do in fact produce audit entries.

3. Audit logs should be directly readable, and should be easy to analyze. Audit entries should be easily distinguishable from other entries in system logs; brevity should not compromise readability; and audit entries should have sufficient structure to make analysis by program simple.

4. Auditing should be efficient to minimize its time/space cost.

5. Because invalidating audit log contents would allow penetration attempts to go unrecorded, audit log buffers and permanent logs need to be protected from sabotage, by restricting access to them, by copying log buffers into permanent logs frequently, and by printing hard copies of logs frequently.

6. The ability for users to determine if they are being audited should be installation-controllable.

III. Proposed Mechanism

Two logs currently maintained in Multics can be augmented to contain protection audit data:

1. The "system" log, maintained in ring 4 solely by the system-control process, now records events concerning entry to the system and operations performed on behalf of other processes.

2. The "syserr" log, maintained in ring 0, records exceptional events concerning the actions of already existing processes.

Both these logs and the mechanisms for maintaining them will be enhanced in several ways:

1. Some entries currently made in these logs will include additional details concerning access authorizations of processes and terminals and access classes of branches.

2. New log entries will be made for events relating to protection.

3. Administrative controls will be added to specify which of many possible protection events cause audit log entries for which persons, projects and branches.

4. Mechanisms for taking installation-changeable special action (such as notifying the operator) will be added.

5. Mechanisms must be added to prevent loss of log information due to log buffer overflows.

6. New control files will be created to make use of log summary tools currently available to produce periodic protection audit summaries. Slight enhancements of these tools may prove useful to do some processing that cannot be done at audit message logging time.

The following sections describe the proposed mechanisms in more detail.

III.1 Auditing in the System Log

Three classes of events relating to protection are currently audited in the system log:

normal system entry and exit (successful logins and dials, and logouts)

denied system entry (unsuccessful logins and dials)

installations of SATs, PNTs and PDTs

Log entries for normal system entry and exit and for denied entry currently include person and project IDs, hardware channel name, terminal type, and terminal ID (from answerback), plus several other fields. For normal system entry and exit, these will now include a character string representation of the created or destroyed process' access authorization. This string will take the form

L:CCC...C

where "L" is a decimal sensitivity level number and "CCC...C" is an octal representation of the access category bits. This representation can be converted to an installation-specific character string (e.g., "Sensitive,Personnel,Budget") by log summary and analysis tools. For denied system entry, the log entry will include the access authorization of the terminal used (in the above format) and the text of the login line as entered by the user. (The terminal authorization indicates the maximum access class of data the user might have been able to access if he had successfully logged in. The login line helps in determining just what the user was trying to do.)

Two events of denied system entry may require special action at some sites:

- too many bad passwords for this personid since last good password

- personid maximum authorization is less than terminal authorization

At some sites, the "too many bad passwords" event may indicate an attempt to gain unauthorized entry to the system. In this case we want to notify the operator, perhaps giving the room number and location of the terminal. At other sites, this event may be of no special importance, and notifying the operator would be an unnecessary annoyance. The message to be sent to the operator for this event will be put in the installation-changeable "as_error_table_". If the message is not null, the system will notify the operator via phcs_$ring0_message, giving the text of any per-channel descriptive information found for this channel. If the message is null, the operator will not be notified.

The event of a person using a terminal with a higher authorization than his own maximum authorization may be of vital importance at some sites, and totally irrelevant at others. (In military applications, terminals with high authorizations are always physically located in controlled areas with the same authorization. A less authorized person using such a terminal has therefore broken physical security.) Again, the operator message for this event will go into as_error_table_, and the operator will be notified only if the message is not null.

With the addition of person, project, and user authorizations to PNTs, SATs and PDTs (respectively), changes to these authorization fields need to be detailed in the log as well. The system-control process is in the best position to do this at table installation time. It constructs a new table from the current table using a template given it by a System Security Administrator (for PNTs and SATs) or by a Project Administrator (for PDTs). It will create a list of updates of authorization fields as it builds the new table. If the installation is performed (no errors), the list of authorization changes is put into the system log. Attempts to install SATs and PNTs by persons who are not SAs or SSAs will also be logged in the system log. Attempts to install tables by persons who are not SAs, SSAs, or PAs will be audited in the syserr log by monitoring accesses denied to "proj_admin_seg" (where all persons allowed to perform table installations are cataloged). (See section III.3.)

III.2 Auditing in the syserr Log

The syserr log mechanism is appropriate for auditing protection events that can occur in an arbitrary process. It resolves locking problems associated with simultaneous log entries from processes on different processors, and is available in ring 0 where most protection events will occur or be discovered. A new module (let's call it "protection_audit_" for now) will be added to ring 0. All auditing will be done via this module. protection_audit_ will not interpret audit messages, but will simply enforce some formatting rules and syserr control argument use. (We don't want auditing crashing the system.)

The events to be logged in the syserr log fall into several classes, depending on the ring in which they occur.

The ring 0 events that will be auditable by direct calls to protection_audit_ in ring 0 are:

- initiations of protected segments and directories (those with non-null access class)
- access denied to any segment or directory (due to improper authorization, ACL mode, or ring validation level)
- access violation faults
- illegal procedure faults
- attempts to "send down" an IPC message to a process of improper authorization
- salvager turning on "security-out-of-service" bit on a branch (because of an inconsistency in protection

attributes)

- rejected attachments of devices (tapes, disks) to processes (for the future)

- setting and resetting of system privilege bits (to allow trusted system programs to perform necessary operations)

Some of these events occur as a result of calls through gates into ring 0 programs which audit their own operation.

In ring 1, several events may occur that need to be audited:

- message segment overflows (because overflow patterns could transmit information to processes of lower authorization)

- message segment access isolation mechanism violations (i.e., attempts to delete messages of lower access class)

- message segment extended access violations

- refused tape and disk mounts due to improper authorization or ACL mode (for the future)

These events are audited by ring 1 programs through a gate (probably "admin_gate_") to protection_audit_ with ring brackets (0,0,1) and an ACL entry giving "re" to "*".

Two SSA operations performed by specific ring 0 primitives in SSA processes will be audited by direct calls to protection_audit_ in ring 0:

- downgrading the access class of a segment or directory

- turning off the "security-out-of-service" bit on a branch (after resolving inconsistencies discovered by the salvager)

III.2.1 Format and Contents of Audit Entries in the syserr Log

Log sequence number, date and time, and syserr action code are already maintained for each syserr log entry by syserr. New syserr action codes are defined in MTB 071. The new codes use the units digit to control system action and the tens digit to distinguish between classes of log entries. Protection audit entries will occupy class 2 (syserr codes 20 through 29). protection_audit_ will use syserr codes 23 (log the message, write it on the operator's console, sound beeper) or 24 (simply log the message) only.

Audit entries will include the following character string components, in order:

audit mnemonic
process group ID of causing process
access authorization of causing process
identification of object (segment, directory or process)
access class of object
additional information

The audit mnemonic distinguishes between types of protection events. A mnemonic is a character string that appears verbatim in a standard place in all audit messages in the log. It is specified by the program calling protection_audit_. Mnemonics will be short. A limited component-name convention will distinguish different flavors of the same type of event (e.g., different actions of the SSA). The mnemonics will be declared internal static initial in an include file.

The process group ID and access authorization of the causing process (i.e., the process executing when protection_audit_ is called) identify the active agent of a protection event. These items will be read directly from the pds by protection_audit_.

For segments and directories, the object identification is the full pathname of the branch. The program calling protection_audit_ will obtain the full pathname and access class of the branch. In the case of fim, these items will be obtained by protection_audit_.

For the target process of an IPC message, the only easily available object identification on the sending side is the target process ID. (See MTB 067.) Therefore, the target process ID will be logged as an octal character string. (The system-control process will log the process group ID/process ID binding in the system log at process creation time. The translation from process ID to process group ID can be done at log summary and analysis time.) The calling program will supply the process ID and access authorization of the target process.

(A cleaner mechanism for obtaining the target process group ID was rejected due to insufficient justification for the additional ring 0 segment required. A new non-wired table of per-process information, including the process ID/process group ID bindings, could be maintained in ring 0. It would be referenced via APTE pointers. It would not have to be locked because processes would only read from it; all writing would be done by the system-control process at process creation or destruction time. This would have made the process group ID of the target process of an attempted IPC send-down available at audit time.)

The "additional information" field contains any other relevant information about the event being audited (such as the actual instruction that caused an illegal_procedure fault).

III.3 Audit Selectivity

The ability to not audit certain events is important primarily for efficiency reasons, and to prevent the log from being obscured with irrelevant or unimportant messages. The auditing mechanism need not be invoked when it is not needed, and the size of audit logs can thereby be reduced.

The events logged in the system log are mostly of general interest for other than protection audit reasons, and are fairly infrequent. Providing selectivity on events logged in this log would therefore serve little useful purpose.

Selectivity of events audited in the syserr log will be provided by a set of "audit select" flags in the pds of each process. When a program discovers an auditable event, it will check the relevant audit flag. If the flag is ON, a call to protection_audit_ is made; otherwise, no call is necessary. For ring 1 auditable events, the protection_audit_ entry called by the gate into ring 0 will check the relevant audit flags. A gate from ring 1 to ring 0 with "re" to "*" will be provided to allow ring 1 programs to determine what auditing is being done. A new gate from ring 4 to ring 1 will allow specific installations to control who has access to determine if they are being audited.

The pds audit flags will be set at process creation to the union of two sets of audit flags in the SAT (for the project) and the PNT (for the person). The pds flags are not changed during the lifetime of a process. The SSA sets the audit flags in the SAT and PNT. The default for these flags at registration is OFF, so that no auditing is performed unless actively specified by the SSA.

The currently defined audit selectivity classes are:

- initiations of protected branches (branches with non-null access class)
- access denied to any branch due to improper authorization, ACL mode, or ring validation level
- access violation faults
- illegal procedure faults

- attempted IPC send-downs
- rejected device attachments
- message segment access violation attempts and overflows
- other refused ring 1 operations (tape and disk mounts)
- SSA protection operations performed directly by SSA processes

The setting and resetting of system privilege bits are extremely important events from a security viewpoint. Therefore, audit messages will always be logged for these events whenever access isolation mechanisms are in use at an installation (i.e., whenever "system high" access authorization is higher than level 0, no categories). If experience proves that system privilege bit settings are frequent enough to clutter the audit log, and that some system processes are trustworthy enough to operate without auditing their privilege bit settings, an additional audit flag can be added without difficulty.

The ability to audit <u>all</u> initiations or access denials of a particular segment or directory is useful in monitoring the use and attempted misuse of protected information. For example, monitoring "proj_admin_seg" in this way causes all attempts to install PDTs by persons who are not project administrators to be audited.

To implement this selectivity, a bit is reserved in each branch, OFF initially, to control this function. The SSA will have a command to turn this bit on and off via a gate into a ring 0 program (which will audit itself). An audit entry will be made for initiations, access denials, and access violation and illegal procedure faults if this bit is ON, irrespective of pds audit flags for the causing process.

III.4 Log Maintenance and Protection

The mechanism used to maintain the system log automatically creates a new log segment when the current one becomes full. Only Initializer.SysDaemon has "w" access to the system log segments. Therefore, no changes are necessary to guarantee the integrity of the system log.

The syserr logging mechanism uses a wrap-around circular buffer with a maximum size of 256K words in its own disk partition to store syserr messages. With the possibility of a high volume of audit messages, the following mechanism will be used to prevent the loss of audit messages. During system initialization, an event call channel of high priority will be set up for the syserr logging mechanism to cause the system-control process to copy the log buffer. (The channel number will be stored in the syserr log buffer header.) A set of relative thresholds (amounts or fractions of buffer not yet copied) in the syserr log buffer header will specify that wakeups should be sent to the system-control process as each threshold is exceeded. (Multiple thresholds guard against lost wakeups.) The high scheduling priority of the system-control process, the low frequency of event calls to the system-control process, and the high priority of the log copying event call channel virtually guarantee that the syserr buffer will be copied faster than any user process can cause new audit messages. If the buffer does wrap around, a message will be written on the operator's console.

III.5 Audit Log Analysis

It is too early to determine which of the many possible ways of summarizing or analyzing audit logs will prove most useful. Experience with using the audit mechanisms is necessary to discover the kinds and frequencies of auditable events that constitute normal and abnormal system operation. Therefore, no tools for statistical analysis of audit logs will be generated initially.

Some simple tools already in use by system administrators to print daily summaries of system and syserr logs are useful in eliminating unwanted messages from summaries. Basically, these tools (daily_log_process and daily_syserr_process) scan the logs under direction of a control file. The control file selects or inhibits the selection of log entries based on date, time, severity, log sequence number, and occurrence of key character strings.

Audit entries in both system and syserr logs have stylized formats. Syserr audit entries include a character string mnemonic for the type of event. Therefore, these tools can be used without change to produce readable protection audit summaries.

Future enhancements of these tools may include some pre-processing to translate 1) process IDs into process group IDs, and 2) numeric codes for authorizations and access classes into installation-defined character strings.