PROJECT MAC June 6, 1973
Computer Systems Research Division Request for Comments No. 27

SOME SMALL IMPROVEMENTS TO ACCESS CONTROL IN MULTICS

by Michael D. Schroeder

This RFC combines some suggestions made by Rick Gumpertz and Jerry Saltzer with some of my own into a proposal to simplify and make more flexible the access control mechanisms in Multics.

The Default ACL Term Problem

Whenever a new segment (or directory) is created, the first ACL for the segment currently is created in a three step operation:

1) An ACL with the single term "*.SysDaemon.* rwe" is created.
2) The appropriate initial ACL from the containing directory is added on top of the ACL created in step 1.
3) A default term with an access id of "<person>.<project>.*" and a mode supplied as an argument to the supervisor, is added on top of the ACL created in step 2. The access id comes from within the supervisor when hcs_$append_branch is used to create the segment, and is supplied by the caller as an argument when hcs_$append_branchx is used. In the second case, "<person>.<project>.*" is almost always supplied by the caller.

The initial ACL provides ACL terms reflecting the directory in which a segment is created while the default term reflects the principal identifier of the creating process. Both functions are necessary.

The terms reflecting the location of the new segment in the hierarchy are quite flexible, as the content of an initial ACL is user-specifiable (with appropriate permission). The term reflecting the creating process is very inflexible. The user has no control whatever if hcs_$append_branch is used, and effectively has no control when hcs_$append_branchx is used. (Almost all standard callers of the latter gate supply the access id "<person>.<project>.*"). This inflexibility can cause difficulty in some cases. For example, the access id "<person>.*.*" would

---

be useful in the default term of a segment created by a user that was a member of multiple projects; or the access id "*.<project>.*" might be just the thing within a project of closely cooperating users. RFC No. 21 points out a more severe problem caused if user-specified instance tags are implemented.

In RFC No. 21 a correction for this inflexibility was suggested that involved a special ring 0 repository for an arbitrary access id portion of the default ACL term. This repository was to be initialized at login depending on the principal id of the new process. A default term, with this access id, would be used as in step 3 presented earlier. This strategy partially cures the inflexibility. Rick Gumpertz pointed out, however, that because the first ACL is constructed from two sources it is still difficult for a user to understand exactly what ACL will first appear on a new segment. He suggested a simple extension that will allow initial ACL's to assume the function performed by the default ACL term, thereby removing the need for the latter. The extension is to allow the variable "-p" to appear as any component of any term in an initial ACL. When a new segment is created, all such variables encountered in the relevant initial ACL will cause the corresponding component of the principal identifier of the creating process to appear in that position in the ACL of the new segment or directory. For example, the initial ACL:

```
-p.-p.*          <mode>
*.SysDaemon.*    <mode>
*.-p.*           <mode>
```

would cause the following ACL to be associated with a new segment created by a process of principal "Schroeder.CompSys.a":

```
Schroeder.CompSys.*   <mode>
*.SysDaemon.*         <mode>
*.CompSys.*           <mode>
```

This facility seems to remove the need for any other sort of default ACL term, thus eliminating step 3 in the creation of the first ACL for a new segment (or directory). Its useful flexibility also seems greater.

Given the above modifications, a normal initial ACL would contain the term:

-p.-p.* <mode>

If the initial ACL were in a directory containing sensitive information accessible only under a particular instance tag (as described in RFC No. 21), then the initial ACL would contain the term:

-p.-p.-p <mode>

Users with special needs could use the "-p" variable in other ways, as well.

The Initial ACL Modes Problem

Note that the above modifications seem to remove the use for the mode argument sppplied to hcs_$append_branch and hcs_$append_branchx. This mode argument performs an important function, however, being the only mode information that accurately reflects the intended use for a segment. The mode bits in initial ACL terms are insensitive to the intended use of a segment. For example, in order to give a particular principal general permission to read and/or execute, as appropriate, all segments created in some directory, that principal must be matched by an initial ACL term which gives read and execute access to all segments created in a directory. Thus, the principal ends up being able to execute data segments. A related problem is that there is no convenient method for changing the mode bits in all terms of a segment's ACL after the segment is created to reflect a change in the intended use of that segment.

A combination of ideas suggested by Rick Gumpertz and Jerry Saltzer appears to alleviate these problems. The modification will be presented in two stages. To start, we define a new attribute for segments (and directories) -- the standard mode. This attribute specifies the normal access mode (some combination of rwe for segments) for the segment. For example, the standard mode for a data segment would be rw, while for a pure procedure segment it would be re. To determine the access privileges available to the process of a particular principal, the standard mode of a segment is ANDed with the mode from the matching ACL term. Thus, the mode in an ACL term is really a mask for the standard mode.

The standard mode for a segment initially would be specified by the mode argument of the append_branch call. Thus, the mode provided by the procedure calling append_branch would modify the effect of those provided by the initial ACL terms. Because the ANDing of the standard mode with an ACL term mode occurs every time access of a principal is verified from the branch, changes in the standard mode after segment creation will be immediately effective.

The proposal, as suggested so far, has one drawback -- it is not possible to grant access in excess of that specified in the standard mode. Occasions may arise where certain principals, e.g., those of SysDaemon's, require more than standard access. This can be met by introducing absolute mode bits, as opposed to the mask mode bits discussed above. Absolute mode bits (represented with capital letters) in an ACL term do not require a match in the standard mode to grant access. Thus, the standard mode can be extended by an ACL term mode. For example, if the standard mode for a segment were re, then the ACL:

```
Schroeder.CompSys.*    rwe
*.SysDaemon.*          RW
*.*.*                  re
```

would give read/execute access to "Schroeder.CompSys.*", read/write to all SysDaemon's, and read/execute to all others. It is anticipated that absolute mode bits would not be used very often. Absolute mode bits would also appear in initial ACL's.

The standard mode of a segment would initially be set from the mode argument of the append_branch call, and would always be updated to reflect current usage. For example, a compiler would set the standard mode of an object segment to rw when writing code into it, then reset it to re.

A user wishing to share procedure and readable data segments would give re permission to sharees via ACL terms. Pure procedure segments with a standard mode of re would be readable and executable by authorized borrowers; and during recompilation, when the standard mode was rw, borrowers would have only r access. Data segments with standard modes of r or rw would be only readable by such borrowers. Notice that compilers, editors, etc., no longer manipulate ACL's, but only set the standard mode.

If some trusted principal other than the creator or manipulator of a segment is to be given all standard access to a segment, then an ACL term mode with all mode bits set would cause the access of that trusted borrower to be exactly the dtandard mode. A principal with permission to modify an ACL can force access, independent of the standard mode, by using "capital letter" mode bits in the matching ACL term.