TO: C. T. Clingen
F. J. Corbató
J. W. Gintell
N. I. Morris
J. H. Saltzer ✓
M. D. Schroeder
M. Weaver
S. Webber

FROM: V. L. Voydock, R. J. Feiertag

DATE: June 7, 1971

SUBJECT: Some proposals concerning ACL's, CACL's and rings

Enclosed are two documents. The first discusses the interaction between ACL's and rings. The second proposes the elimination of the CACL and describes a new construct to replace it. Please try to read them by 6/14/71. As soon as all comments have been received a design review of our proposals will be scheduled.

R. J. Feiertag, V. L. Voydock

Proposal for ACL's

This memo considers problems relating to Access Control Lists. At present ACLs on nondirectory branches and directory branches have different meanings, so they will be considered separately.

First the case of non-directory branches will be considered. It is assumed that the reader is familiar with the meaning of ring brackets on non-directory segments and how they affect the rewa attributes. Modifications to the meanings of the ring brackets were considered, however, none of these changes were considered to make rings easier enough to use or understand to warrant the extensive changes **necessary to** both software and proposed hardware.

The other main topic of consideration with respect to ACLs on non-directory segments is to determine when a process has the right to modify an ACL and what the ACL should look like. It had been proposed that only one set of ring brackets be associated with a segment instead of a set of ring brackets per user per segment. Although the ability to have several sets of ring brackets associated with a segment is not widely used, in those cases where it is used it is very handy. Secondly, since most users will have the same ring brackets on all ACL entries of a particular segment they will effectively have only one set of ring brackets per segment. In other words the user must become more sophisticated only if he wishes to use the added features.

Finally, we consider who has the right to modify a non-directory branch ACL. For this purpose, what we believe to be the present algorithm is quite adequate. This scheme can be stated by the following simple rules:

1. The first ring brack, r1, of an entry to be placed on an ACL must be greater than or equal to the process's current validation level. This guarantees that the user cannot place an executable segment in a lower ring.
2. The validation level of the process must be within the write bracket with respect to the segment whose ACL is being modified. Recall that the write bracket of a segment with respect to a particular process is the range 0 to r1 inclusive where r1 is derived from the ACL entry pertaining to this process. In other words the validation level must be less than or equal to the r1 found in the ACL entry of the segment that applies to this process.
3. The process must have write permission in the directory containing the segment whose access is to be modified.

Must be some [illegible] if defining a gate.

These rules provide sufficient access checks so that no improper modifications can be made to an ACL. Also, the rules are simply stated and relatively easy to understand with proper foundation. Rules 2 and 3 should also be used to determine if a process has the right to modify any of the other segment attributes, namely names and bit count.

The necessity of Rule 3 is obvious. Rule 1 **assures that no process** running in a particular ring can place a segment in a more privileged ring, e.g. a process running in the user ring cannot give himself 0, 0, 4 ring brackets on one of his own segments thereby allowing it to run in the hardcore ring. Rule 2 is the most interesting and has the most consequences and will now be discussed in detail.

Why is Rule 2 necessary? It assures that a process cannot modify its access rights to a segment in such a way that it can give access to information in a more privileged ring. Consider the case of a stack. The ring 1 stack should have ring brackets of 1, 1, 1 to assure that no less privileged rings can access it. If we applied only Rules 1 and 3 it would be legal for a process running in ring 4 to modify the ring brackets on the ring 1 stack to 4, 4, 4. Rule 2 prevents the modification of an ACL if the segment resides (with respect to this process) in a more privileged ring than the one in which the process is currently running.

Rule 2 is meaningful because assuring that the process is running within the write bracket of the segment means that as far as the ring mechanism is concerned the segment can potentially be modified by this process, therefore, the process should also be able to modify the segment attributes. For example consider the following ACL:

Feiertag.Multics.*	rewa	1, 1, 1
Voydock.Multics.*	rewa	4, 4, 4
Gintell.Multics.*	rewa	1, 4, 4

If Feiertag is running in ring 4 he cannot modify this ACL because he is not in the write bracket of the segment. Voydock running in ring 4 can modify the ACL because he can modify the information in the segment, therefore, he might as well be able to change its attributes. If it is not desired that Voydock be able to modify the segment attributes then his write permission should be deleted from the directory containing the segment. Note that it is possible for Voydock running in ring 4 to change Feiertag's ring brackets to 4, 4, 4. Therefore, a simple statement indicating how well protected a segment is within the ring mechanism is that a segment is no more secure than its highest rl. In the above example the segment is secure for ring 4 because no process in a ring greater than 4 can modify it.

This feels wrong

Now consider what rights the process Corbató.Multics.a has with respect to this segment. In order to determine the rights of this process it is necessary to define what a process's effective access is to a segment if it does not appear on an ACL. Clearly the process does not have any of the rewa attributes. The question then becomes in what ring does it not have these attributes. Since the process has null access the ring brackets are not important to the protection mechanism. The only place where they are important is to Rule 2 above. There are really two possible choices. Either not appearing or an ACL means the process's access is null, 0, 0, 0 or null 7, 7, 7. If we choose the former we have the problem that if someone deletes all entries from an ACL then no one can ever touch that segment again. It cannot even be deleted. This situation is not acceptable. Therefore, a user not appearing on an ACL must imply that his access is null 7, 7, 7 to the segment. This creates a small problem for protected segments such as message segments. If Feiertag creates a message segment in his directory then its ACL would consist of:

How does this help?

Feiertag.Multics.* rewa 1, 1, 1

This would mean that any other process could modify this ACL from ring 4 since all other processes have null 7, 7, 7 access. Therefore, protected segments must fully specify their ACLs. This means that the ACLs of protected segments must contain a *.*.* entry. For message segments such an entry would be *.*.* rewa 1, 1, 1. This concludes the discussion of Rule 2 and ring brackets with respect to non-directory segments.

The rules concerning ring brackets on directories exist primarily for compatibility with the rules concerning ring brackets on non-directory segments. Directories have three ring brackets, but only two of these are currently meaningful. It is not even clear that directories should have ring brackets or how many they should have or what they would indicate. For this reason we propose the elimination of ring brackets on directories. Since it is believed that no one makes use of directory ring brackets, their elimination should cause no waves. Later, if it is decided to make use of ring brackets on directories they can be added back in. Deleting them now and then adding them back in later will produce fewer compatibility problems then not deleting them now and trying to change their meaning later when users have found some exotic applications for their current interpretation.

~~A~~ Should also remove the attribute REWA for directories

L S M ~~A~~
~~A~~
~~A~~
A