TO: MSPM Distribution
FROM: Karolyn Martin
DATE: August 22, 1968
SUBJECT: BX.8.02, BX.8.03

These two sections are being re-issued to reflect the following changes in the ACL commands:

1) arrays of varying strings are no longer used as arguments because the shell to be installed does not handle them, and they are highly inefficient. The entry argument in all the commands has been reduced to a single value.

2) References to gate lists (long outdated) have been removed.

Published: 08/20/68
(Supersedes: BX.8.02, 04/10/67)

Identification

Modify the access control information for a file
setacl, delacl, settrap, set_protection
E. Bjorkman, P. Smith, C. Marceau, K. Martin

Purpose

The access control commands enable the user to replace or modify the access control list (ACL) of segments in the file system hierarchy. Setacl modifies access control modes. Delacl deletes access control names. Settrap provides trap procedures for access control lists. Set-protection provides protection lists. This section assumes that the reader is familiar with the conventions for access control described in BG.9 and BX.8.00.

Discussion

In order to modify the ACL for a segment, the user must have the read, execute, and write attributes on in the directory which contains the segment. If the user attempts to modify the ACL of a segment, given a link to it, he must have, beside the above access in the directory containing the segment, the execute attribute on for the directory containing the link and for all directories containing intermediate links.

Access control lists

An access control list (ACL) for a segment is a list of access control names; with each name is associated a mode and a protection list which define the access to the segment for that name. The common access control list (CACL) of a directory controls access to all segments in the directory. When the user attempts to access segment B in directory A, access control checks first the ACL of B, and then (if the pertinent access control name is not found in the ACL) the CACL of the directory A containing B.

The ordering of access control names is discussed in BX.8.00, as is the notion of mode. This section contains a brief discussion of access control names and of protection lists.

Access Control Names

As explained in BG.9.00 and BX.8.00, access to a segment is governed primarily by access control name, i.e. combination of personal name and project id. A further refinement allows for discrimination between one process-group of a user and other process-groups of the same user. Each process-group of a user has a two-character instance tag (see BQ.3.00) which distinguishes it from other process-groups of the user: thus it is possible for a user to protect his segments from himself, for example, for debugging purposes. A typical "user name" in an access control list might be:

```
John_Doe.MAC.zq
```

where "zq" is an instance tag. Normally the third component of an access control name is *, meaning "any instance tag".

Protection Lists

The protection list defines the access bracket, the rings in which the user's access to the segment is determined by his effective mode and the call bracket, the rings (other than the access bracket rings) from which the user can only transfer to the segment (the call bracket is always outside the access bracket).

The protection list consists of a list of numbers where the first number is the low bound of the access bracket and the second number, which must be greater than or equal to the low bound, is the high bound of the access bracket. A protection list consisting of one number defines an access bracket which is equal to that ring number. The third number in the protection list, if present, is the high bound of the call bracket. The low bound of the call bracket is not specified since it must be one greater than the high bound of the access bracket. (i.e., it is not permitted to say that lower rings are less privileged than higher rings).

Direct access to the segment from rings lower than the access bracket is controlled by the user's effective mode except for the use of the execute attribute. An attempt to execute the segment from a lower ring (given the execute attribute on for the user) results in a ring crossing (BD.9.00). Direct access to the segment from the rings in the access bracket is controlled by the user's effective mode and attempted execution of the segment (given the execute permission) does not cause a ring crossing.

Direct access to the segment from the rings in the call bracket is denied but attempted execution at specific entries (gates) of the segment (given the execute permission) is allowed and causes a ring crossing. All access from rings higher than the call bracket is denied.

The * Convention in Access Control Commands

As defined in BX.8.00, the * convention can be used to denote a set of access control names in an ACL. Thus

.MAC.

includes both John_Doe.MAC and Susie_Q.MAC.

However, to change the access of John_Doe.MAC to a segment, it is not necessarily enough to change the access of *.MAC as that access may be overriden by the access of John_Doe.MAC. i.e., "John_Doe.MAC" and "*.MAC" are two independent names in the access control list, and access for one name is modified independently of access for the other name.

Note that ACL's are so ordered that the file system finds and uses access information for specific users before access information concerning a group of users. (For weighting of access control names, see BX.8.00.)

The * convention is interpreted in the usual way for entry names. i.e., changing the ACL for *.epl means changing the ACL for "a.epl" as well as "b.epl".

Arguments to Access Control Commands

The order of arguments to these commands must be preserved. To specify a null argument, a user must type `'` (null literal string). See BX.8.00 for a full description of the entry argument to the commands described here. That section also contains description of the "omit" option, which can be used in conjunction with entry.

Usage

To add an access control name to the ACL or the CACL of a set of entries or to change the mode for an access control name, the user invokes the command

setacl entry mode acname1 --- acnamen

entry is an entry, i.e. path name relative to the working directory or to the root directory. Entry may define a set of segments, according to the * convention. If this argument is null, the ACL's of all segments in the working directory are modified.

mode is any combination of the letters TREWA, which stand for the attributes Trap, Read, Execute, Write, and Append, respectively. If mode is null, i.e. `', acname1 thru acnamen will have no access rights. (i.e., will be explicitly denied access to the segment. Thus Susie_Q.MAC may be denied access even though *.MAC has read access.)

acname1 --- acnamen are the access control names. Each name must have three elements, as described above. If acname1 --- acnamen are omitted, the name of the user (personal name plus project ID) who invoked setacl is assumed, with the third element (instance tag)*.

Comments

Setacl modifies the ACL for the segments defined by entry. If the user has no right to modify the ACL for some segment, setacl prints an error comment. If a name in acname1 --- acnamen appears in the ACL of a segment, setacl modifies the mode for that name. If a name does not appear in the ACL of a segment, setacl enters the name, with mode, into the ACL, and prints a comment at the console. (The comment is under control of the brief option.)

For each name it adds to the ACL, setacl supplies a protection list. The protection list supplied by setacl consists of exactly one number, and limits access to the segment to the ring from which the user invoked setacl (i.e. the validation ring at the time the user invoked setacl). The user may change the protection list of a segment by using the command set_protection (see below).

Setacl does not supply a trap procedure when it sets the trap attribute for a segment. If no trap procedure is in the ACL, and the trap attribute is on, access control will ignore the trap attribute and take the effective mode from the attributes TREWA. To specify a trap procedure for a trap attribute, the user should invoke settrap (see below).

To set the CACL of directory b with pathname a>b, the user specifies entry name "a>b>". The final ">" signifies that the CACL of b is to be modified rather than the ACL of the branch which points to b. I.e., the access to all segments in directory b is to be modified, rather than the access to b itself. A later version of setacl will set ring brackets as well as mode.

Delacl

A second access control command, delacl, can be used to delete access control names from the ACLs or CACLs of a group of segments and directories:

delacl entry acname1 --- acnamen

entry is an entry, i.e. path name relative to the working directory or to the root directory. Entry may be a set of entries defined by use of the * convention.

acname1 --- acnamen are the access control names. Each name must have three elements. Omisson of all access control names causes an error comment.

Comments

Delacl deletes all access control information on each segment defined by entry for the specified access control name and deletes the name from the ACL. If a specified access control name does not appear in the ACL, delacl notifies the user of this fact and processes the next name. (Notifying the user--by printing a message at the console---is under the control of the brief option.)

To delete access control names from the CACL of directory b with pathname a>b, the user specifies as an entry "a>b>". The final ">" signifies that the CACL of b is to be modified, rather than the ACL of the branch which points to b. I.e., the access to all segments in directory b is to be modified, rather than the access to b itself.

Settrap

To supply a trap procedure for an access control name in the ACL of a group of segments, the user invokes

settrap entry trap `arglist' acname1 --- acnamen

entry is an entry, i.e., path name relative to the working directory or to the root directory. Entry may define a set of segments, according to the * convention.

trap is a procedure name. If this parameter is null, the command deletes the trap procedure for each access-control name in acname1 --- acnamen with respect to entry.

arglist is a literal string representing the arguments to trap. trap and arglist are concatenated to form a command line which is interpreted and executed when an access control trap occurs. If trap is null, arglist is ignored.

acname1 --- acnamen are access-control names. If these arguments are omitted, the name of the user (personal name plus project ID) invoking settrap is assumed, with the third element (instance tag) *.

Comments

Settrap searches the ACL of entry for the names in acname1 --- acnamen. If an acname is in the ACL, settrap sets the trap attribute on (leaving other attributes unchanged) and enters trap and arglist into the access control list. If an acname is not already present in the ACL, and trap is non-null, settrap enters it in the ACL with mode T, and enters trap and arglist into the ACL. Settrap then prints a message at the console that the new access control name has been added (this message is not printed if the brief option is on).

If the argument trap is null, i.e. `', settrap deletes trap information from the ACL and sets the trap attribute off.

To set a trap procedure in the CACL of directory b with pathname a>b, the user specifies as an entry "a>b>". The final ">" signifies that the CACL of b is to be modified, rather than the ACL of the branch which points to b. i.e., the access to all segments in directory b is to be modified, rather than the access to b itself.

Settrap does not check the existence of the procedure trap or the correctness of arglist. If access control (see BG.9.00) cannot execute trap, for some reason (e.g. no such procedure exists), the other attributes (TREWA) define the effective mode of the segment.

Errors

There are two types of errors which the user may make in using the access control commands described in this section:

1) The user does not have the requisite access privileges. For example, he attempts to modify the ACL of segment b in directory a, but does not have the Read and Write attributes on in directory a.

2) The user supplies incorrect or meaningless arguments to the command. For example, he gives mode as "QZ15((2"; mode must be at most five characters, chosen from the set "TREWA". Or the user specified entry "a>b>", but b is not a directory, hence contains no CACL.

In addition, a third type of "error" may occur: no two users may change access control information in a directory at the same time. If setacl discovers that another user is modifying access control information in the directory it will print an appropriate comment and not attempt to modify the CACL or any ACL in the directory.

Implementation

The ACL for a segment consists of a list of access control names, with access control information for each name, as discussed earlier in this section. An ACL is represented by the following structure:

```
dcl 1 acl (aclct) ctl (aclp),
      2 acname,
        3 name char (24),
        3 projid char (24),
        3 instance_id char (2),
      2 mode bit (5),
      2 plistp ptr,
      2 trapp ptr;
```

acl - an array of access control names with associated information. The number of names is aclct.

acname - an access control name, consisting of three elements: a personal name, a project ID (projid) and an instance ID.

mode - five bits, representing TREWA. If a bit is 1, the corresponding attribute is on.

plistp - pointer to an adjustable bit string containing the protection list.

trapp - pointer to an adjustable character string which represents a trap procedure with arguments.

Each command in this section follows a basic pattern:

1) It considers the segments specified by entry one at a time.

2) For each segment, it write locks the directory containing the segment, then calls readacl (see BG.8.02) to obtain the current ACL for the segment in an acl structure (described above).

3) The command modifies the structure according to the user's wishes, calls writeacl (see BG.8.02) to replace the segment's ACL, then unlocks the directory containing the segment.

4) The command prints error comments and other messages at the user's console.

The reason for write-locking the directory containing the segment is to avoid a race in writing the ACL. Consider the following example: User A decides to modify the ACL of segment x. At about the same time, user B decides to modify the ACL of segment x. User A's command calls readacl to obtain the ACL. Then user B's command calls readacl. Now user A writes the ACL so that it contains the modification he desires. Next user B writes the ACL; A's modifications are lost. To avoid such an occurrence, A first "write-locks" the directory containing the ACL for segment x. When B tries to read x, he (i.e., setacl executing for B) is informed that the directory is locked.

The implementation of setacl is described here in some detail. Modifications of the pattern in other commands are briefly noted.

Setacl

After checking the validity of mode, setacl weighs each of the names in acname1 --- acnamen (or the user's name if they are omitted) by calling the external function weighter (described below). Each name is entered into the following array:

```
dcl 1 namelist (n),
    2 name char (24),          /* personal name of user */
    2 projid char (24),        /* project ID of user */
    2 instance_id char (2),    /* instance ID */
    2 weight fixed bin (17);   /* weight attached to
                                  access control */
```

Here n is the number of elements in acname1 --- acnamen. The external function orderusers (see below) orders namelist so that namelist (1) contains the "heaviest" access control name and namelist (n) the "lightest" access control name.

Next setacl considers entry. Setacl calls the procedure entryarg (see BY.2.04) to convert the entry argument into a combination of

1) the path name (relative to the root directory) of the directory which contains the entry or set of entries defined by the symbolic entry name;

2) the segment name (not a path name).

If the segment name is null (i.e. entry ends in ">"), then setacl modifies the CACL of the directory returned by entryarg. If the segment name has some component denoted by * or ** (signifying a set of segments), then the entry names it defines are obtained by calling the file system library procedure star (see BY.2.02).

The ACL's for the individual entry names returned by entryarg and star are modified one at a time. For each entry, setacl locks the directory containing the entry. Next setacl calls the Directory Supervisor primitive readacl,

```
call readacl (dir, entry, areap, aclp, aclct, errcode);
```

Dir and entry are a directory pathname and a single entry name (name of one segment in the directory).

Areap, supplied by setacl, is a pointer to an area in which readacl can allocate the acl array of structures. Readacl fills in the array and returns aclp, a pointer to the array, as well as aclct, the dimension of the array.

Errcode indicates any errors which may have occurred.

In the event of an error, setacl unlocks the directory containing the entry, then prints a message at the console and goes on to the next entry (or the next set of entries, depending on the nature of the error - for example, the user does not have Read and Write access to the directory containing a set of entries).

If there is no error, setacl now proceeds to construct a new access control list in the form of a controlled array of structures, newacl, identical to acl except for dimension. To compute the dimension, newaclct, of newacl, setacl compares the names in namelist with the names in acl. Newaclct is equal to n + aclct less the number of matching names. Setacl now allocates newacl and copies elements of acl into it. When a name in acl matches a name in namelist, the new mode is entered for that name. Other information for the name is left unchanged. When a name in namelist has no match in acl, it is added, at the place appropriate to its weight, in newacl (i).

The pointer newacl(i).plist is set to point to a protection list consisting of one number, the number of the validation ring at the time setacl was called (Section BD.9.00 discusses the concept of validation ring). Newacl(i).gatelistp and newacl(i).trapp are set to zero.

When newacl is complete, setacl calls the directory supervisor primitive writeacl to replace the old ACL for the segment.

```
call writeacl (dir, entry, newaclp, newaclct, errcode);
```

where newaclp is a pointer to newacl. Finally, setacl calls setseglock to unlock the directory containing the access control list just written.

Delacl

The implementation of delacl is similar to that of setacl. When a name in the array acl matches a name in namelist, that name is not copied into the newacl array. When there is no match in acl for a name in namelist, the name is inserted in a comment to be printed at the user's console.

Settrap

The implementation of settrap is similar to that of setacl. When a name in the array acl matches a name in namelist, the trap attribute for that name is set on, the trap procedure name and arglist are entered in an adjustable string for that name. If a name in namelist has no match in acl, the name is added to acl with mode T and inserted in a comment to be printed at the user's console. The trap procedure name and arglist are entered in an adjustable string for that name. If the argument trap is null, the T attribute is deleted from the mode for acnames.

Weighter

Weighter is an external function which receives as input an array of access control names, and returns a structure containing the names with their associated weights. The weight of a name is:

$$\sum_{i=1}^{3} 2(i)x(i)$$

where x(i) = 0 if the component of the name is *, and x(i) = 1 otherwise. (Each access control name has three components: personal name, project id, and instance id.)

Orderusers

Orderusers is an external function which orders access control names by weight. Input to orderusers is a namelist array of structures (defined above). The function sorts the weight components so that the name with the highest weight is in namelist (1).