MULTICS TECHNICAL BULLETIN MTB-360

To: Distribution

From: J. C. C. Jagernauth

Date: May 5, 1978

Subject: The Multics Data Base Manager (MDBM) Security

INTRODUCTION

The MDBM data base architecture is being reorganized to improve the performance of the relational interface and to allow for future implementation of a CODASYL compliant network interface. This network interface generates and accesses a data base consisting of the same internal structure as one created by the relational interface. This new organization degrades the current level of security provided for relations unless they are contained in unique files. In addition, tuple level security (securing each record occurence) and attribute level security (securing each field within a record-type) are desirable security features for the MDBM. Tuple level security is not considered in this document because the resulting performance degradations can be significant. Attribute level security can be provided either as a Multics feature (e.g. pfile_, documented in MTB-317) or as an MDBM function. The pfile_ concept is not used because security breaches may occur when variable length attributes are accessed. As a result, attributes and relations are secured by the MDBM, while files are secured in the current Multics fashion.

The MDBM security features proposed in this MTB are flexible since the data base administrator (DBA) is allowed to apply these features to security sensitive data only, thus limiting the overhead necessary to secure a data base.

It is assumed that the reader is familiar with relational and CODASYL terminology and has read MTB 359 which describes the proposed enhancements to the MDBM. Please forward all comments and suggestions to the author at:

Honeywell Information Systems
P.O. Box 6000 MS K-28
Phoenix Az. 85005

-or-

Jagernauth.Multics on system M

RELATIONAL SECURITY REQUIREMENTS

With a few exceptions, the proposed performance enhancements to the MDBM allow both the network and relational interfaces to access the same data base. Therefore, the MDBM security features must satisfy the requirements of both the relational and the CODASYL data base interfaces. The DBA of a relational data base is allowed to optionally secure files, relations and attributes. The standard Multics acl mechanism is used to secure files. If attributes or relations are to be secured, the containing files (in most cases) are placed in a lower ring and access controlled by the data base manager.

Retrieve, update and null access privileges are provided for files. Each file in the new data base architecture has a file model segment associated with it. To control "open" privileges, read or null modes are applied automatically to the corresponding file model segment when file access is set. Read access is applied to the file model segment if retrieve or update is required for the file. A data base can be opened by users having non-null access to at least one file of that data base. Write access on data model segments is needed for restructuring purposes and is initially provided for the DBA only. At data base creation time the DBA is granted sma on the data base directory, rw on all multisegment files and rw on every file model segment (i.e., the maximum access rights possible are granted to the DBA).

Retrieve, update, store, delete, modify and null access privileges are provided for relations. Store, delete and modify are autonomous data base operations and are collectively equivalent to the update access mode. Securing a relation causes the containing file to be placed in a lower ring if: 1) specific store, delete, or modify access rights are required, or 2) more than one relation is clustered in the file that contains the relation to be secured, and the relation privileges differ from those of the file.

Retrieve, update and null access privileges are provided for attributes. Securing an attribute causes the containing file to be placed in a lower ring unless all attributes of that file have identical access privileges.

DATA BASE VISIBILITY

It is proposed that the following rules be implemented within the MDBM to prevent unprivileged users from displaying file names, relation names and attribute names.

1. The file-names of a data base can be obtained by users with open privileges to that data base (i.e., non-null access on at least one file). In the "File Security" example described below, note that everyone registered in the FSO project may list the names of all files contained in the data base although they have no access to file1.

2. Relation name visibility requires non-null privilege on the file containing that relation. In the "File Security" and "Relation Security" examples described below, G66 users may list the names of all relations contained within file1, even though their access to the individual relations is limited.

3. Attribute name visibility requires non-null privilege on the containing relation. In the "Attribute Security" example described below, FSO users may obtain the names of every attribute in the sales relation although they may only retrieve volume information.

IMPLEMENTATION PROPOSAL

Access to peripheral devices on Multics is controlled by access control segments (ACSs), which are zero length segments whose acls are used to verify a user's access rights to peripherals. It is proposed that this concept (together with lower ring placement of data) be used in the MDBM to secure relations and attributes. The MDBM security mechanism is designed to allow for dynamic control of changing security requirements. Whenever possible, normal acls are applied to the multisegment files contained in the data base. However, ACSs and extended acls are used in many cases when secure relations and attributes are required. In these cases an ACS directory is created consisting of an ACS_control segment and the access control segments. The DBA is initially granted sma on the ACS directory and rw on the ACS_control segment. In addition, the DBA is granted rw on ACSs as they are created. The following sections describe how files, relations and attributes are to be secured.

FILES are secured by setting standard Multics access modes (read, write and null) on the containing multisegment files. Read and write are equivalent to retrieve and update respectively.

RELATIONS may uniquely reside in files or they may be clustered, so there are several scenarios for providing secure relations. The following is a description of each:

1. A relation resides in a unique file and requires retrieve, update, or null privilege.

   Applying access modes to the containing file is sufficient to secure this relation and specific access modes should not be set. However, if they are specified, the MDBM recognizes that there is only one relation in the file and specific relation privilege is not set.

2. A relation is clustered and requires retrieve, update, or null privilege.

   The containing file is placed in a lower ring if it is not already there, an ACS is created for the relation to be secured and read, write or null access mode is applied to that ACS. After this is done, the MDBM checks for and eliminates duplicate ACSs. A duplicate ACS is one that has the same access control list as a previously defined ACS. The containing file is not placed in a lower ring if its security requirements are identical to the relation's requirements. All of the processing described above can be avoided if the DBA does not set specific access modes for a relation when every other relation in that file has identical security requirements; because the file access rights apply to all relations contained therein.

3. A relation resides in a unique file and requires store, delete, or modify privilege.

   The containing file is placed in a lower ring, an ACS is created for the relation and extended access modes are applied to it, providing the user has update privilege on that file. The MDBM checks for and eliminates duplicate ACSs. The extended access modes are used by the MDBM to permit or prevent store, delete, or modify operations.

4. A relation is clustered and requires store, delete, or modify privilege.

   The containing file is placed in a lower ring, an ACS is created for the relation to be secured and extended access modes are applied to that ACS. The MDBM checks for and eliminates duplicate ACSs. The user must have update privilege on a file before store, delete, or modify can be provided for a relation contained in that file. Extended access modes are used by the MDBM to permit or prevent store, delete or modify operations.

An ATTRIBUTE is secured by placing the containing file in a lower ring, creating an ACS for the attribute, and applying read, write, or null access modes to that ACS. However, if all attributes in a file have the same security requirements, then the access rights on that file are sufficient to secure the attributes residing therein; so, specific access modes for those attributes should not be set. Similarly, if all attributes in a relation have the same security requirements, then the access rights on that relation are sufficient to secure an attribute contained in that relation; so, specific access modes for those attributes should not be set. Whenever access modes are set for an attribute, the MDBM checks the security specifications of all attributes with those of the containing relation and file. If they are identical for every attribute and the containing file, the ACS is deleted and the file is removed from the lower ring, providing no other relation or attribute in that file is secured. On the other hand, if they are only identical for every attribute and the containing relation, the ACS is deleted and the relation ACS is used to control access to the attribute.

Unnecessary ACSs are not created. But, even if the acl of one attribute in a file differs from the other attribute acls, that file has to be placed in a lower ring and an ACS created. Therefore, the ACS_control segment is designed to identify the access control list that must be used by the MDBM to determine the access rights of a secured relation or attribute. It would be extremely inefficient for the data base manager to verify security privileges via the storage system access checking primitives everytime a secured item is accessed. So, at file ready time, the MDBM obtains the user's access rights to all relations and attributes for the file, if it is a lower ring. This information is stored in a per-process bit array, which is called the current access vector (CAV); and access rights verification is accomplished via the CAV until the data base is closed. This means that once the user has opened a data base, the DBA is not allowed to change the user's access rights until that data base is closed. In this manner, overhead is kept to a minimum because the storage system access checking primitives are used only once for each user session.

PERFORMANCE CONSIDERATIONS

The security mechanism degrades MDBM performance only when files are placed in a lower ring. Accessing lower ring files requires an extra call to a gate segment and some additional processing in order to verify the access rights to secured relations and attributes. The additional processing required is negligible since the CAV is used for access verification purposes. So, only the extra call to a gate segment is significant and its exact impact on performance can be easily measured when this security mechanism is implemented.

EXAMPLES

A department store data base is used in this section to illustrate the functionality of the proposed MDBM security mechanism. This data base is identical to the one discussed in Section 2 of the Logical INquiry and Update System (LINUS) Reference Manual, Order No. AZ49. It contains the following relations:

emp (name emp_no dept mgr sal comm)
sales (dept item vol)
supply (supplier item vol)
loc (dept loc)
class (item class)

The emp, sales, and supply relations are assumed to be clustered in file1, while the loc and class relations reside in file2 and file3 respectively.

The following modes are used to specify data base access privileges:

r	retrieve
u	update
s	store
m	modify
d	delete
n	null

File Security

Access rights may be set on the three files without causing data placement in a lower ring. Assume that files 1, 2, and 3 have the following security specifications:

file1:

ru	*.Multics.*
ru	*.NCB.*
r	*.MED.*
r	*.G66.*

file2:

ru	*.Multics.*
r	*.NCB.*
r	*.FSO.*

file3:

ru	*.Multics.*

r	*.HDSA.*
ru	*.FSO.*

It is intended that the users listed above have the specified access rights to all data contained in files 1, 2, and 3 respectively, as long as specific acls are not set on relations and attributes. The following sections show how these acls have to be manipulated in order to secure relations and attributes.

Relation Security

Securing relations may result in files being placed in a lower ring. Assume the following security specifications for the emp relation:

ru	*.Multics.*
r	*.NCB.*
r	*.FSO.*

The MDBM places file1 in a lower ring because the security requirements of the emp relation are different from those of file1. Since emp and file2 have the same requirements, an entry is made in the ACS_control segment to indicate that access to emp is controlled by the file2 acl (see Table 1), but an ACS is not created. Note that FSO users are not listed on the file1 acl. So, the DBA must set retrieve access for *.FSO.* on file1 before attempting to set access modes for the emp relation. When this is done, FSO users are able to retrieve data from every relation in file1, which is not intended. Therefore, null access has to be set for *.FSO.* on the sales and supply relations. An ACS (named GDS) is created for this purpose (see Table 1). The acl on GDS contains:

ru	*.Multics.*
ru	*.NCB.*
r	*.MED.*
r	*.G66.*
n	*.FSO.*

Also, let the sales and class relations have the following specifications:

rsdm	*.Multics.*
rs	*.NCB.*

In this case file3 is placed in a lower ring, an ACS (named STD) is created and extended acls are applied to it. An ACS_control segment entry is made for the class and sales relations (see Table 1). Note that although G66 users have retrieve privileges on file1, they cannot access the supply relation because they were not included in the specific acls above.

Attribute Security

Securing attributes may result in files being placed in a lower ring. Let attribute vol of the sales relation have the following specifications:

ru	*.Multics.*
r	*.NCB.*
r	*.FSO.*

File1 is already in a lower ring, so no ring change is necessary. Since file2 has the same security specifications, an entry is made in the ACS_control segment indicating that the acl of file2 controls access to attribute vol of the sales relation (see Table 1). Again, FSO users are not listed on the acl of the sales relation. So, the DBA must set retrieve access for *.FSO.* on the sales relation. But, this allows retrieve access to every attribute in the sales relation, which is not intended; so the DBA must now set null access for *.FSO.* on the dept and item attributes of the sales relation. (Refer to table1 for the control segment entries.)

Now, let the sal attribute of the employee relation have the following security specifications:

ru	Jag.Multics.*
r	*.Multics.*

In this case an ACS (named GHI) is created to control access to the sal attribute. An entry is also made to the ACS_control segment. It should be noted that everyone in the Multics project can retrieve salary information, but only the user "Jag" may also update salary data. No other user is allowed to access the salary attribute.

Table 1 illustrates in a logical fashion the contents of the ACS_control segment.

Table 1.

Relation/Attribute to be secured	Unique ACS_name	ACS_name or file_name that currently applies (where to look)
file1.emp	XYZ	file2
file1.sales	ABC	STD
file1.emp.sal	GHI	GHI
file1.sales.vol	DEF	file2
file3.class	STD	STD
file1.supply	GDS	GDS
file1.sales.dept	ALM	GDS
file1.sales.item	AMO	GDS

Suppose the security requirements of file2 change to the following:

r	*.Multics.*
r	*.NCB.*
ru	*.MED.*
u	*.G66.*

Then, all references to file2 in the ACS_control segment are removed and an ACS (named XYZ) is created to control access to the emp relation and the vol attribute of the sales relation (refer to Table 2).

If the security requirements of the class relation change to:

r	*.Multics.*
u	*.NCB.*

Then, file3 is removed from the lower ring and its entry is deleted from the ACS_control segment (see Table 2).

The ACS_control segment now contains the following:

Table 2.

Relation/Attribute to be secured	Unique ACS_name	ACS_name or file_name that currently applies (where to look)
file1.emp	XYZ	XYZ
file1.sales	STD	STD
file1.emp.sal	GHI	GHI
file1.sales.vol	DEF	XYZ
file1.supply	GDS	GDS
file1.sales.dept	ALM	GDS
file1.sales.item	AMO	GDS

CODASYL SECURITY REQUIREMENTS

The general format of the CODASYL security mechanism is:

PRIVACY LOCK [FOR ¦¦ (codasyl usage modes) ¦¦] IS { literal-1 lock-name-1 PROCEDURE data-base-procedure-1 } [OR { literal-2 lock-name-2 PROCEDURE data-base-procedure-2 }] ...

It is proposed that this format be maintained by the new CODASYL interface except that literals, lock names, or data base procedures be replaced by access identifiers, and the name PRIVACY be replaced by SECURITY to identify the deviation being made from the proposed CODASYL standards.

The proposed general format for the new CODASYL interface is:

SECURITY LOCK [FOR ¦¦ (codasyl usage modes) ¦¦] IS access_id1 [OR access_id2] ...

The CODASYL privacy standards specify various usage modes for schemas, areas, data items, records, members and set types. These usage modes are mapped to the access modes defined (in this document) for files, relations, and attributes. Details of this mapping will be provided when the new CODASYL interface is documented in a future MTB.

IMPLEMENTATION DETAILS

The MDBM security mechanism consists in major part of primitives that perform atomic functions necessary to secure data contained within the new data base architecture. These primitives may then be used by both the relational and network interfaces to satisfy their specific security needs.

Gates are provided to allow the DBA and the MDBM to access lower ring data. The DBA needs access for activities such as dumping the data base, while the MDBM has to verify user access rights and manipulate the lower ring data. Care will be taken during the implementation of gates to ensure that other lower ring data are not inadvertently destroyed.

The following are descriptions of commands required by the data base administrator to secure relational data bases. The mrds_set_acl, mrds_delete_acl, and mrds_list_acl commands are upward compatible with the current versions of these commands. The data base files are not placed in a lower ring if only file access modes are specified. However, this lower ring protection may be required by some DBAs who are concerned that their data bases may be accidentally destroyed by unprivileged users. The smdrb command provides those administrators with the ability to place data bases into a lower ring. The CODASYL mechanism for securing data bases will be described when its interface is documented.

Name: mrds_set_acl, msa

The DBA is allowed to set the access privileges on attributes, relations, and files contained in a MRDS data base. Executing this command can sometimes result in one or more files being placed in (or removed from) a lower ring. In these cases the files affected are forced to a quiescent state before acls are set and ring bracket changes made. Access rights of an active user cannot be modified. Ring bracket changes are not allowed if the smdrb command was used to set specific ring brackets on the data base files.

Usage

```
     mrds_set_acl path mode1 {user_id1}... {moden} {user_idn}
{control_args}
```

where:

1. path

   is the pathname of a relational data base.

2. modei

   is a one- to four-character combination of the following valid modes:

r	retrieve
u	update
s	store
m	modify
d	delete
n	null

   store, delete, and modify apply to relations only. If these modes are specified, then the -attribute and the -file control arguments cannot be used. Store, delete, and modify are collectively equivalent to update. Therefore, the update mode is invalid if used in conjunction with store, delete, or modify.

3. user_idi

   is an access control name of the form Person_id.Project_id.tag. All user_ids with matching names receive modei. (For a description of the matching strategy, refer to the set_acl command in the MPM Commands and Active Functions.) If no match is found and all three components are present, an entry is added to the access control list. If the last modei has no user_id following it, the user's Person_id and current Project_id are assumed.

4. control_args
   can be selected from the following:

   -attribute STR, -attr STR
      specifies attributes with which the acls are to be associated. STR is a list of variables of the form rel_name.attr_name or rel_name. The rel_name is used to refer to all attributes in the named relation. If STR is not specified then every attribute in the data base is assumed. This control argument is invalid if the store, delete, or modify mode is used. A user must have equivalent or greater access rights on the containing relation before attribute privileges can be set. For example, a user who has only retrieve on a relation cannot be given update rights on an attribute contained in that relation. However, update is granted if the user has store, delete, modify, or update privilege on the relation.

   -relation STR, -rel STR
      specifies relations with which the acls are to be associated. STR is a list of relation names and/or file names. A file name is used to refer to all relations in that file. If STR is not specified then every relation in the data base is assumed. A user must have equivalent or greater access rights on the containing file before relation privileges can be provided. For example, a user who has only retrieve on a file cannot be granted update on a relation contained in that file.

   -file STR, -f STR
      specifies files with which the acls are to be associated. STR is a list of file names. This control argument is invalid when the store, delete, or modify mode is specified. If STR is not specified then every file in the data base is assumed. Read access mode is set on a file model segment for the user with non-null access rights to the corresponding file.

Notes

If control arguments are not specified, then the access privileges are applied to all files. All non-null users of a data base are given status rights on the data base directory and read/write privileges on the data base control segment. However, if a user has null access to all files, relations, and attributes, then null access is also applied to the dbc segment and the data base directory.

The -attribute and -file control arguments do not apply to data bases created before release 7.0 of the MDBM.

Name: mrds_list_acl, mla

Access modes are listed for attributes, relations, and files contained in a MRDS data base.

Usage

```
mrds_list_acl path {user_ids} {control_args}
```

where:

1. path
   is the pathname of a relational data base.

2. user_ids
   are access control names of the form Person_id.Project_id.tag. All access entries with matching names are listed. (For a description of the matching strategy, refer to the set_acl command in MPM Commands and Active Functions.) If user_id is omitted the entire acl is listed.

3. control_args
   can be selected from the following:

   -attribute STR, -attr STR
   specifies the attribute acls to be listed. STR is a list of variables of the form rel_name.attr_name or rel_name. The rel_name is used to specify all attributes in the named relation. If STR is not specified then every attribute in the data base is assumed.

   -relation STR, -rel STR
   specifies the relation acls to be listed. STR is a list of relation names and/or file names. A file name is used to specify every relation in the named file. If STR is not specified then every relation in the data base is assumed.

   -file STR, -f STR
   specifies the file acls to be listed. STR is a list of file names. If STR is not specified then every file in the data base is assumed.

Notes

The modes output by this command are the same as described for the mrds_set_acl command. If control arguments are not specified then the acls of every file in the data base are listed.

Name: mrds_delete_acl, mda

Entries are removed from access lists of attributes, relations, and files contained in a MRDS data base. Executing this command can sometimes result in one or more files being placed in (or removed from) a lower ring. In these cases the files affected are forced to a quiescent state before acls are set and ring crossings made. Access rights of an active user cannot be modified. Ring crossings are not allowed if the set_mdbm_database_ring_brackets command is used to set specific ring brackets on the data base files.

Usage

```
     mrds_delete_acl path user_ids {control_args}
```

where:

1. path
   is the pathname of a relational data base.

2. user_ids
   are access control names of the form Person_id.Project_id.tag. All access entries with matching names are deleted. (For a description of the matching strategy, refer to the set_acl command in MPM Commands and Active Functions.) If no user_id is specified, the user's Person_id and current Project_id are assumed.

3. control_args can be selected from the following:

   -attribute STR, -attr STR
   specifies the attribute acls from which deletions are to be made. STR is a list of variables of the form rel_name.attr_name or rel_name. The rel_name is used to refer to every attribute in the named relation. If STR is not specified then every attribute in the data base is assumed.

   -relation STR, -rel STR
   specifies the relation acls to be deleted. STR is a list of relation names and/or file names. A file name is used to specify all relations in the named file. If STR is not specified then every relation in the data base is assumed.

   -file STR, -f STR
   specifies the file acls to be deleted. STR is a list of file names. If STR is not specified then every

file in the data base is assumed. The user's read access mode on a file model segment is deleted when his access rights on the corresponding file are deleted.

Note

If control arguments are not specified then entries are deleted from the acls of every file in the data base.

Name: set_mdbm_database_ring_brackets, smdrb

The ring brackets of files contained in the data base specified can be modified by the DBA. This command provides the DBA with the ability to place files of a data base into a lower ring when the Multics ring mechanism, along with normal Multics acls are the only protection needed for those data base files.

Usage

```
set_mdbm_database_ring_brackets path {args} {control_args}
```

where:

1. path

   is the pathname of the data base created by the MDBM.

2. args

   can be chosen from the following:

   rb1

   is the number to be used as the first ring bracket.

   rb2

   is the number to be used as the second ring bracket.

   rb3

   is the number to be used as the third ring bracket.

3. control_arg

   can be -file STR or -f STR which specifies the files that require ring bracket changes. STR is a list of file names. If this control argument is not specified then ring brackets are modified for every file in the data base.

Notes

This command cannot be used to set the ring brackets of a file that was placed in a lower ring by the MDBM to satisfy security requirements specified via the mrds_set_acl command. Conversely, only the file access modes may be changed via the msa command for data base files that are placed in a lower ring by this command. If rb3 is omitted, the third ring bracket is set to rb2. If rb2 and rb3 are omitted, the ring brackets are set to rb1. If rb1, rb2, and rb3 are omitted, they are set to the user's current validation level. The ring brackets must be in the allowable range 0 through 7 and must have the ordering:

$$rb1 \leq rb2 \leq rb3$$

The user's process must have a validation level that is less than or equal to rb1. Ring brackets and validation levels are discussed in Section 6, "Intraprocess Access Control" of the MPM Reference Guide.