To: MTB Distribution

From: Jim Gray

Date: March 17, 1981

Subject: The New MRDS Security Approach.

Send comments by:

Multics mail on System M to JGray.Multics

Telephone to HVN 341-7463 or 602-249-7463

Continuum meeting mrds_sec, link to transaction 388.

1. INTRODUCTION

This MTB describes the new approach to providing security for the Multics Relational Data Store (MRDS), and Logical Inquiry and Update System (LINUS). This new approach has had a long history of of design evolution. The following references can answer many questions on the motivation for the approach taken.

MTB-360 : The Multics Data Base Manager Security, J. Jagernauth

MTB-431 : Changes to Design of MRDS Security, C. Tavares

Multics Relational Data Store: A Case Study, M. Good, et. al., MIT (see Section 8: Security)

MTB-DRAFT : Security via Submodels for MRDS, L. Spratt (>udd>m>lls>mtb>ss.mtb)

Continuum meeting, mrds_security, the MRDS group

2. PAST RELATION LEVEL SECURITY APPROACH

In MR7.0, and earlier releases of MRDS (version 3 databases), the approach taken was to provide an extended set of MRDS acl's (store(s), modify(m), delete(d), retrieve(r)) that were translated into appropriate Multics acl's on a relation basis.

Three commands were provided for this purpose, mrds_set_acl, mrds_delete_acl, and mrds_list_acl. Their job was to take a MRDS extended acl for a relation, say store for foo, and map it into Multics acl's on the database structure, in this case write access on the foo vfile multi-segment file, and possibly some vfiles used in the invert_dir directory containing data about attributes that were declared as secondary indexes.

3. CURRENT RELATION LEVEL MRDS SECURITY

In the MR8 release of MRDS (version 4 databases), security is still provided on a relation basis, but due to the new architecture, there is no need for a set of MRDS acl commands.

Instead, the user can set Multics acl's on segments and multi-segment files under the database directory, having names related to the relation name. Thus to be able to store into relation foo, you set "rw" access on the segment foo.m and the msf foo. In both this and the past version of MRDS, "rw" access was needed on the concurrency control segment, and "r" access on the common database model segment.

4. DESIRED ATTRIBUTE LEVEL SECURITY

There was a desire to provide security on an attribute, rather than relation basis, as first proposed in MTB-360. Serious work on this proposal was prompted by the Executive Office of the President contract that was first discussed in 1979. Reviewing the original proposal showed serious problems with it, as noted in MTB-431. This led to a new proposal on how attribute level security would be provided. (see the MIT study and MTB-DRAFT). This approach has further evolved into the approach to be described in this paper. (see the mrds_security continuum meeting)

5. PROPOSED ATTRIBUTE LEVEL CONTROL

The new approach is called attribute level control (ALC) rather than attribute level security (ALS). This is because the

providing of complete security, that does not allow subversion by using a means other than accessing the data by the data base manager, will require putting MRDS code and databases into a lower ring. This will make use of all the security that Multics is currently capable of providing. This will not be accomplished in the next release (MR9), but is planned for a future release.

The mechanism for providing (ALC) without breaking existing applications on existing databases, and to allow a user to do without security if he so desires, is to "turn on" security via a new command called secure_mrds_db.

In order to make use of the security he has turned on, a user must have created a new (version 5) submodel that has been extended to provide access specification on both a relation and an attribute basis. The things the user can specify are, null or append_tuple (i.e. store), and/or, delete_tuple on a relation. Also null or, read_attr and/or modify_attr on an attribute within a relation. Note that attribute and relation access are orthogonal.

Once this ALC type submodel has been created, for security purposes, it must be placed under the database directory in a new inferior directory "secure.submodels". The command create_mrds_dsm has a "-install" option that creates this new directory, if it is not present, and creates the submodel under this directory, instead of the working directory.

Now that the user has a secure submodel, and a secured database, he can open the database through this submodel view, and be provided by database manager controlled attribute level security.

To enforce submodel views as the only means of secure access to the database, the secured database will not be allowed to be accessed (by a non-DBA, see next section) via the primary database view, or submodels not in the secure.submodels directory.

6. ADDING THE DATABASE ADMINISTRATOR CONCEPT

If anyone were allowed to create a submodel and place it in the submodel directory, then with sufficient Multics acl's on the MRDS database, he could subvert attribute level security. If he could access the database via the main model, rather than a submodel, the security is also subverted.

For this reason the Data Base Administrator concept will be added to MRDS. This DBA will be defined as someone that has "sma" access on the database directory. He will automatically be given any other access he would ever need, in order to make future

access requirements invisible to him, and to ease documentation and understanding of what a DBA really is. For a secured database, users will be divided into non-DBA's, that can only access the database through a submodel view, and DBA's that may access the database model view. For an un-secured database, some things will still be restricted to a DBA, such as the commands AMDB, QMDB, and CMDSM using the -install option.

Only a DBA (once the database has been secured) will be allowed to create secure submodels (i.e. submodels that are installed in the secure.submodels directory). Further, he will be the only one that can access the database via the main model view.

A non-DBA will be restricted to what he can do through his submodel view of the database with all MRDS commands and subroutines. This means that subroutines such as dmd_, that can only access the main model view, will not be available to him.

7. DATA ACCESS VIOLATION DETECTION

The time at which an attempt to violate an ALC access restriction is detected is at the time of attempting to set scope (concurrency control modes). This is because scope must be set before data may be accessed.

A few access violations will still be detected by MRDS at data selection time. This is because concurrenty control (scope) is still on a relation level, while security is being extended to attribute level. For example, a user may set scope of modify_attr on a relation, if he has modify_attr access on any attribute within that relation. However, he does not need modify_attr access on all attributes, so an attempt to modify an attribute which does not have this access permission will be detected by MRDS at data selection time.

If the database is not secured, then only Multics acl's will be checked against the requested scope modes. These modes (append_tuple, delete_tuple, read_attr, modify_attr) indicate desired operations, thus imply needed access to perform that operation. (e.g. append_tuple requires "rw" access to the relation data)

Once the database has been secured, MRDS access modes will also be checked against the implied access needs of the scope request. Thus append_tuple scope requires append_tuple MRDS access.

The vfile containing the relation data will not be attached and opened until after this access checking has been done, in order to avoid having vfile detect access violations of Multics acl's.

8. MODEL ACCESS VIOLATION DETECTION

The attempt to access unauthorized parts of the database model will be left strictly to Multics acl's. To make best use of ALC, the database directory, and submodel directory should have null access for all but the DBA.

The relation and database model segments should have acl's set in accordance with the view access definitions. Thus for a view with relations r001 and r002 out of four in the database, the view user needs at least "r" access to the segments r001.m, r002.m, and db_model, with null on r003.m and r004.m segments.

The non-DBA should only have "r" access to those submodels that were specifically created for him. He would have "n" access on either the database directory, or the secure.submodels directory. Thus his view of the database model and data is entirely restricted to only that provided by the submodel view.

9. DETAILED INTERFACE CHANGES

The details of the changes to the various MRDS subroutine and command interfaces, in terms of the proposed ALC security approach, and the DBA definition are contained in the MTB's listed below.

MTB-496 : Proposed Changes in the MRDS Submodel Interface, N. Davids

MTB-502 : Effects of Security on the MRDS Interface, J. Gray

MTB-503 : Changes to the MRDS Command Interface, J. Gray

MTB-504 : Changes to the MRDS dsl_ Subroutine Interface, J. Gray

MTB-505 : Changes to the MRDS dmd_ Subroutine Interface, J. Gray

MTB-506 : Extension to the create_mrds_dsm and display_mrds_dsm Commands for MRDS Security, N. Davids