Microsoft Access Security

Last Updated:

Purpose

The purpose of this document is to explain the Microsoft Access security model. The hope is that if the user understands the intricacies of security, and is conscientious, he will be able to safely and successfully use Microsoft Access security in his databases.

Examples of scenarios that have caused difficulty for people in the past have been set apart under the heading of **Gotcha**. These are intended to give practical examples of common misunderstandings, and to explain why Microsoft Access security works the way it does.

This document is applicable to Microsoft Access versions 1.0 and 1.1, as well as to the Microsoft Access Distribution Kit (ADK). Features new to Microsoft Access 1.1 and the ADK are called out with **New for 1.1**.

Disclaimer: There are several places in this document where the Microsoft Access sytem tables (those MSys) are referred to. In spite of this, these tables are officially "undocumented", and are subject to change in future versions. Any attempts to read from or especially to write to these tables will most almost certainly fail future versions of Microsoft Access.*

Overview

Microsoft Access security consists of 2 parts which are stored in different places. Information regarding the permissions that users and groups have on the objects in a database is stored in the database itself. This way the permission information travels with the .mdb file in which the objects exist. All other security information is stored in the SystemDB specified in msaccess.ini. The default SystemDB is system.mda. Information stored in the SystemDB includes: which Users & Groups exist; which Groups each User belongs to; and User logon passwords. In Microsoft Access parlance, a SystemDB defines a "Workgroup".

Note: Retail Microsoft Access looks for the file msaccess.ini to find it's settings, including SystemDB. Applications created with the ADK can specify any file for these settings (e.g. myapp.ini.) For the purposes of this document, msaccess.ini and myapp.ini are interchangeable.

Logging On

Each User and Group has associated with it a Security ID (SID). The SID is a binary string that uniquely identifies the User or Group. When a user logs on, Microsoft Access looks in the MSysAccounts table of the SystemDB for a user of the same name (case-insensitive). If a user with the same name is found, it then validates the password (case-sensitive). If the password matches, the SID of the user is retrieved and saved in an internal structure. The password is only used to validate the user when he logs on. Other than validating the identity of the user when he logs on, it has no effect on security.

By default, Microsoft Access first attempts to logon as the user Admin, with a blank password. If this logon fails, the user is presented with the Logon dialog. If a user name and password were specified on the command line (using the /USER and /PWD flags), Microsoft Access first tries to logon using that user name and password. If this logon fails, the user is presented with the Logon dialog. Once logged on, the user's SID is retrieved. This SID is used for all subsequent security operations within Microsoft Access.

Users and Groups

Each user can be a member of 1 or more groups. Users & Groups share the same namespace. This means that you can't have both a group and user with the same name.

Microsoft Access defines 3 default groups: Admins, Users, and Guests. Whenever a user is added to the SystemDB, they are automatically given membership in the group Users, and they cannot be removed from the Users group. There is one exception to this rule---the user Guest CAN be removed from the Users group. The Admins group must always have at least one member.

Microsoft Access defines 2 users: Admin and Guest. The user Admin is a member of the Admins and Users groups. The user Guest is a member of the Guests group (only).

The pre-defined groups (Admins, Users, and Guests) cannot be deleted. The user Guest cannot be deleted, though the user Admin CAN if there is at least one other member of the Admins group. Chapter 25 of the Access Users Guide recommends deleting the Admin account. The only reason for this recommendation is so that you don't accidentally create any objects using this account. Objects created by Admin cannot be secured.

Each user and group has a SID associated with it. The SIDs of the Users and Guests groups, and of the Admin and Guest users, are the same in all Microsoft Access installations. The SID of the Admins group, however, is unique across all Microsoft Access installations. This prevents someone who is in the Admins group of one SystemDB from being able to have the permissions of the Admins group of any other SystemDB.

> **Gotcha:** If you delete the user Admin, and then create another user named Admin, the SID of the new Admin will NOT be the same as the SID of the Admin that exists in an unsecure system. This is because the Admin user that you create will have a different PIN that the Admin user that exists in a freshly-installed SystemDB.

If you setup from the Microsoft Access disks, the SID of the Admins group is a function of the serial number of the Setup disk and the User and Company names given (case-insensitive). Therefore, it is critical that the disks used to install Microsoft Access be kept in a safe place, and that the User and Company names be recorded.

New for 1.1: You can also create a SystemDB by running the Change Workgroup utility. Using Change Workgroup, you are able specify a Personal Identification Number (PIN) in place of the disk serial number, as well as any User and Company names. This PIN can be any alphanumeric string, up to 20 characters long. The the SID of the Admins group will be determined by these values (also case-insensitive).

In the event that the SystemDB has to be recreated, you must either have the same installation disks, or have recorded the PIN, User name, and Company name used to create it. Because the Change Workgroup functionality removes the connection between the original disks and Access security, this is the recommended method of creating a SystemDB.

The SIDs of users and groups that you create are a function of the user or group name (case-sensitive) and Personal Identification Number (PIN) that you specify. For this reason it is critical that the user and group names and their PINs be recorded. If there is ever a need to recreate the SystemDB, you will need the name and PIN of each user and group that was in the SystemDB. Note that by using different PINs, it possible for users and groups in different SystemDB's to have the same name, but they will actually be different accounts as far as Microsoft Access security is concerned because they have different SIDs.

Gotcha: When you setup Microsoft Access, the User and Company names are written to Disk #1. If you then give anyone else your disks to setup from, Microsoft Access doesn't prompt for the User and Company name. The Admins group in the SystemDB that is created will have the same SID as the Admins group in your SystemDB. This means that anyone who uses that new SystemDB will have all the permissions that you do, in any databases that you have permissions. While this functionality wasn't intended as a form of copy protection, it is a good incentive to abide by the License Agreement. It is also a good reason to create your SystemDB using the Change Workgroup utility in Microsoft Access 1.1

Permissions

There are 2 types of permissions: Explicit and Implicit. Explicit permissions are those given directly to a user. When they are granted, no other users are affected. Implicit permissions are those granted to a group. When they are granted, all users who are members of the group get the permissions of the group. Implicit permissions belong to the group, not the users. If a user is removed from the group, he no longer has the permissions of the group. If a user is added to the group, he gets all the permissions assigned to the group. If the permissions of the group are changed, all users in the group are affected.

When a user attempts to do something, he will get the least restrictive permissions of: 1) all the groups in which he is a member (his Implicit permissions), and 2) those given directly to him (his Explicit permissions.)

In the Microsoft Access Permissions dialog, only the Explicit permissions are displayed. This is a limitation of v.1.x. To view the Implicit permissions, view the permissions of each group that the user is a member of.

The Admins group *of the SystemDB in use when the database was created* will always have permission to change permissions on all objects in the database. This permission cannot be taken away by anyone!!! This permission remains even when all permissions have been revoked from the Admins group, and isn't displayed in the Permissions dialog. It should therefore be obvious that it is very important to know and keep track of which SystemDB was in use when the database was created. To stress this point, italics will be used throughout this document where it comes up.

The Users group initially has Full permissions on all objects created in the database. The Guests group has Read Definition and/or Read Data permissions on them, as appropriate for the object type. Basically, Guests can read all objects and data, but can't change anything.

Permissions on an object can be changed by anyone having permissions to do so. By default, the following users have this permission: members of the Admins group *of the SystemDB in use when the database was created*; the creator of the database (it's owner); the creator of the object (it's owner); and anyone who has Full permissions on the object, either explicitly or implicitly.

Gotcha: Jane gives Joe a database that she created. Joe is a member of the Users group, and no other groups. He is using a different SystemDB (the SID of the Admins group of his SystemDB is different from the SID of the Admins group in Jane's SystemDB). Jane's database is "unsecure", meaning that the Users group has Full permission on all objects. Joe revokes Full permissions from the Users group on a Table1. From that point forward, Joe can no longer change permissions on that Table1. If he also revoked Read Data permissions, he has prevented himself from being able to read Table1, because he no longer has permission to do so. For both these cases, he just revoked his own permissions on the Table1. Because Joe isn't a member of the Admins group *of the SystemDB in use when the database was created,* and because he didn't create Table1, he's completely locked out of Table1 until Jane (using her own SystemDB) gives him permissions again.

All users always have permission to open any database and create objects in it. Microsoft Access relies on operating system security to prevent a user from being able to open a database.

> **Gotcha:** Even if a user doesn't have an account in your SystemDB, he can use any other SystemDB in which he has an account to logon to Microsoft Access. Then he can open any database that he can get to through the operating system. Once the database is open, he can see which objects exist, and create new objects. There is no method to prevent creation of objects.

It is possible for a user to have permissions on an object in a database, and not be able to see those permissions in the Permissions dialog. This will happen if the user was assigned permissions in the database and then was deleted from the SystemDB. The permissions given to the user are stored in the .mdb, and aren't removed when the user is removed from the SystemDB. The following steps will show these permissions:

1. Set the Show System Objects option to Yes
2. Give yourself Read Data permissions on the MSysACEs table
   If you don't have permissions to do this, you're not a member of the Admins group *of the SystemDB in use when the database was created,* and you didn't create the database.
3. Attach to or import the MSysAccounts table from the SystemDB currently in use
   If you don't have permissions to do this, you're not a member of the Admins group *of the SystemDB currently in use.*
4. Run this query:

```
SELECT DISTINCTROW MSysObjects.Name AS Object, MSysAccounts.Name As
    UserOrGroup FROM MSysACEs, MSysAccounts, MSysObjects, MSysACEs
    LEFT JOIN MSysAccounts ON MSysACEs.SID = MSysAccounts.SID,
    MSysACEs INNER JOIN MSysObjects ON MSysACEs.ObjectId =
    MSysObjects.Id;
```

This will show the names of all users and groups who have permissions on all objects in the database. If the UserGroup field is blank, it means that the user or group doesn't exist in your SystemDB, yet still has some kind of permissions on that object. *Again: The query above works fine in Microsoft Access 1.x. However, the Microsoft Access system tables (MSys*) are officially "undocumented", and are subject to change in future versions.*

Ownership

Ownership is a very important part of Microsoft Access' security model. The SID of the user who creates a database is considered the "owner" of the database. He will always have permission to change permissions on any object in the database, regardless of who creates the object, or what permissions are on the object.

The SID of the user who creates an object is considered the "owner" of the object. He will always have permission to change permissions on the object, even if the database owner has revoked his permissions on the object.

The Admins group *of the SystemDB in use when the database was created* will always have permissions to change permissions on any object in that database.

The only way to transfer ownership of a database or object is to recreate it. For databases, this means doing a File New Database and importing all the objects from the old database. Table relationships will have to be recreated after importing, as they're not preserved when importing or exporting.

For objects, ownership can be transferred by importing or exporting the object, or by copying and pasting

it. These operations essentially create a brand new object, one that is completely unsecured. Note that for both of these situations, the person doing the importing/exporting or copy/pasting must have permission to read the object in the first place.

> **Gotcha:** As mentioned above, the Admin and Guest users have the same SID in all SystemDB's. It is therefore critical to security that the Admin and Guest users NOT create (are the owner of) any objects or databases. If they do, anyone who has Microsoft Access and can get to the database .mdb file can open it and have Full permission on the objects created by Admin or Guest, and all objects in the database if either of these users created the database.

The "Hole"

By default, the Users and Guests groups have Read Data permissions on the MSysObjects table. For various technical reasons, this is necessary to allow the user to do File Load From Query in the Filter window.

The astute reader will see that MSysObjects contains a field named "Owner", and he will figure out that it contains the SID of the owner of each object. The astute reader will also realize that the SID field in the MSysAccounts table of system.mda contains his own SID. Because everyone has Read Data permissions on MSysObjects, it would be possible to read the SID of the owner of any given object, and replace his SID in MSysAccounts with it. The next time he logged on, he would have the SID of the owner of the object, and all rights associated with it. He could change permissions on the objects, and therefore do anything he wanted to with it.

SIDs are binary fields. When MSysObjects is browsed, the binary fields are truncated at the first null terminator (ASCII 0). Additionally, any characters whose ASCII values are below 32 will display as garbage. All of this essentially prevents someone from being able to edit or copy and paste this value directly. However, it would be a relatively simple matter to change your own SID using Microsoft Access Basic code or a query.

Fortunately, the Security Wizard will patch this hole. See the section on the Security Wizard, below.

Needless to say, this hole will be plugged in a future version of Microsoft Access. *Disclaimer: Again, the Microsoft Access system tables (MSys*) are officially "undocumented", and are subject to change in future versions. Don't count on any information regarding them to be applicable to anything but Access version 1.x.*

The "Run with Owner's Permissions" Query Property

Microsoft Access queries have an property called "Run with Owner's Permissions" (RWOP). This is in the Query Properties dialog. If this property is checked, the permissions of the owner of the query are the only permissions that are considered when attempting to access tables or queries that the query is dependent on.

> **Gotcha:** Joe has Read Definition permissions on the Salary table, but not Read Data. Joe creates Query1, which is based on the Salary table. Joe has left RWOP turned on. Joe is unable to execute the query, because he doesn't have Read Data permissions on the Salary table. This much is intuitive. Jane created the Salary table, and has Full permission on it. Jane tries to execute Query1. She cannot, because the RWOP option of Query1 is turned on, and the owner of the query (Joe) doesn't have Read Data permissions on the Salary table. While this may not be entirely intuitive, it is correct functionality.

Only the owner (creator) of a query may save it if the RWOP property is turned on. Even the database

owner or members of the Admins group *of the SystemDB in use when the database was created* can't save a query created by another user if the RWOP property is turned on. It would be a security breach to allow them to. Note though, that anyone with Modify Definition permissions on the query can uncheck RWOP and successfully save the query. This may lead to other complications if other people were using the query and depending on RWOP being there.

> **Gotcha:** Joe has Read Definition permissions (only) on the Salary table. Jane creates Query1, which is based on the Salary table, which Jane has Read Data permissions on. The query is defined as "Select LastName from Salary WITH OWNERACCESS OPTION;". Joe has Read Data and Read Definition permissions (only) on Query1. Joe can execute Query1 and see the LastName field. He can also open Query1 in design mode and add any field (like the EmployeeSalary field!) to the query output, then execute the query. He cannot save the query, however. This is arguably a bug. The workaround is that if you're using RWOP queries to enforce security, you need to ensure that Joe doesn't have Read Definition permissions on ANY tables or queries in Query1.

If the owner of a RWOP query has been removed from the system (by using the Users dialog to delete the user), nobody will be able to execute RWOP query. This is because Access first checks to see if the owner exists before executing the query. The solution to this is for someone who has Read Definition permission on the query (and all tables/queries in the query) to do a Save As, then delete the original query and rename the new query to the same name as the original one. You could also add the original user (with the same PIN) back to the system.

This functionality extends to the QueryDef object as well. Only the owner of a query can change the query using the "Set Qry.SQL = MySQLString$" syntax if MySQLString$ contains "WITH OWNER ACCESS OPTION".

Encryption

Database encryption has nothing to do with security. It is only used to prevent someone from using a file or disk editor (such as Norton Utilities) from being able to read and write data in an .mdb file. Access reads and writes all data a page at a time. Pages are 2Kb in size. Encryption is done at the page level, not at the data level. That is, at the level that encryption is being done, there is no notion of what is on the page, only that there's 2Kb of data that needs to be encrypted and written, or read and decrypted. This implies that everything in an Access .mdb file is encrypted, including tables, queries, forms, indexes, etc. Access uses the RSA algorithm for database encryption.

Given a database that isn't encrypted, it would be possible for a (very) knowledgable person to use a disk editor to read the SID of the database owner or the SID of the Admins group *in use when the .mdb was created*. Given that SID, they could assume ownership of the database using the methods outlined above, and do anything they wanted with the database. For this reason, it is recommended that if you want to protect against such a person, you should encrypt your database. Because of this, databases secured by the Security Wizard are encrypted.

Only the creator (owner) of a database, or a member of the Admins group *in use when the database was created*, can encrypt or decrypt it.

Due to the overhead of encrypting and decryption, there is a performance degradation of approximately 10-15% in encrypted databases. Encrypted files are also essentially uncompressible (via PKZip, Stacker, DOS 6 DoubleSpace, etc.)

The Security Wizard

The Access Security Wizard is used to secure databases. It allows you to choose which object types to

secure (Tables, Queries, Forms, Reports, Macros, Modules.) The selected object types are secured by revoking all permissions from the Users and Guests groups.

To install the Security Wizard, copy wzsecure.mda into your Microsoft Access 1.1 program directory. Add the following line to the [Libraries] section of your msaccess.ini file:

```
wzsecure.mda=ro
```

If it doesn't already exist, add a [Menu Add-Ins] section in your msaccess.ini file, and add the following line to it:

```
Security &Wizard==swz_SecureDB()
```

The next time you start Microsoft Access, the Security Wizard command will appear on the Help menu. To secure a database, open it and invoke the Security Wizard by selecting this command.

The Security Wizard will only run on Microsoft Access 1.1, but will work with databases in 1.0 or 1.1 format. Attempting to use the Security Wizard with Microsoft Access 1.0 will result in "Out-of-date database format", followed by "Some library modules could not be loaded" when you start Microsoft Access. Databases created by the Security Wizard are always in 1.1 format.

The Security Wizard can take quite a while to run. For instance, securing nwind.mdb on a 486/50 takes approximately 13 minutes. The About dialog gives more detailed information about what the Security Wizard does.

> **Gotcha:** One of the steps that the Security Wizard takes is to update several internal database security structures to patch "The Hole" (documented above). The only negative side effect of patching "The Hole" is that when people who aren't in the Admins group *of the system.mda in use when the database was created* attempt to use the "Load Filter From Query" or "Save Filter as Query" functionality while in MyApp.mdb, they will get an error that says "No permission for 'MSysObjects'". This problem will be fixed in a future release of Microsoft Access.

Because of the updates to the internal database security structures that the Security Wizard makes, it is the only supported method of securing a database. Any other steps will result in a database that is not fully secured.

Common Scenarios

Securing an Existing Database

You've written an application using Microsoft Access---call it MyApp.mdb. You've created all the objects while you were logged on as Admin (the default.) Now you want to "secure" the database. Here are the steps to follow to do so:

1. Install the Security Wizard using the steps above.
2. Choose Change Password from the security menu and give a password to the user Admin.
3. Create a new user and add him to the Admins group. For this example, the user is Joe.
4. Exit Microsoft Access and restart it, giving Joe as the user when the Logon dialog is given.
5. Open MyApp.mdb
6. Invoke the Security Wizard by selecting the Security Wizard command from the Help menu.
7. Select the object types to be secured.

   The Security Wizard creates a new, encrypted, database, exports all the objects from the current database, and properly secures the selected object types by revoking permissions from the Users and Guests groups. It doesn't change the current database in any way. It recreates attached tables and table relationships in the new db as well.

8. If desired, create any of your own groups and users, and give them appropriate permissions on all the objects in the new database.

The new database is now secure. The only people who can get into the objects in MyApp.mdb are those you gave permissions to in step 8, and anyone who is a member of the Admins group *of the current SystemDB.*

Unsecuring a Secured Database

You've followed the steps under Securing an Existing Database, above. Joe is the owner of the database and all objects in it. The Admins group *of Joe's system.mda* has permissions to change permissions on any object in the database. The goal here is to make the user Admin the owner of the database and all objects in it, and to give the Users group Full permissions on all objects. Remember, the SIDs of the user Admin and the Users group are constant in all system.mda's, so this will effectively give everyone Full permissions on all objects in the database, and make them the database owner as well.

1. Make a backup of MyApp.mdb!!!
2. Boot Microsoft Access, logging on as someone who has Read Data and Read Definition permissions on all the objects in MyApp.mdb.
3. Open MyApp.mdb.
4. Give the Users group Full permissions on all objects in the database.
5. Edit your msaccess.ini file so that SystemDB points to a system.mda in which the original user Admin exists. If you've deleted the Admin account, you cannot recreate it, because you don't know the PIN. You can recreate a clean SystemDB using the Microsoft Access 1.1 Change Workgroup utility. This SystemDB will have an Admin user with the correct SID.
6. Exit Microsoft Access and restart it logging on as the user Admin.
7. Create a new database (call it MyApp2.mdb).
8. Import all the objects from MyApp.mdb into MyApp2.mdb.

Your database is now completely unsecured. Anyone who can get to the MyApp2.mdb file has Full permissions on all the objects in it.

Protecting Your Code

You've written Microsoft Access Basic code that you want to distribute, perhaps as an application or a Library database, but you don't want anyone to be able to read your code. The modules are in MyApp.mdb, and were all created while you were logged on as Admin. Assuming that you're starting from a completely unsecure configuration, simply follow the steps under "Securing an Existing Database", above, choosing the Modules checkbox in step #7.

Your modules are now secured so that nobody except Joe and members of the Admins group *of the current SystemDB* can read the modules. All users will still be able to execute the code---there's no way to prevent someone from being able to execute code.

Moving a Database to Another System.mda

You are Joe, and have written MyApp.mdb and want to give it to Jane who is using a different system.mda. You want Jane to have complete administrative authority over your database. The problem is this: The Admins group *of Jane's system.mda* doesn't have any permissions on anything in MyApp.mdb.

There are 2 variations of this scenario:

Variation #1: MyApp.mdb is completely unsecured (the Users group has Full permissions on everything).

The situation here is as follows... The user Admin created MyApp.mdb and all the objects in it, and is therefore the owner of everything. The Admins group *of Joe's system.mda* has permissions to change permissions on all the objects in MyApp.mdb. The Users group (which

has the same SID in all Microsoft Access installations) has Full permissions on all objects in MyApp.mdb. Jane's system.mda is unsecure (the Admin password is blank, and there are no user accounts or groups.)

To give MyApp.mdb to Jane, and have her get complete administrative authority over it, follow these steps:

1. Make a backup of MyApp.mdb!!!
2. Boot Microsoft Access *using Jane's system.mda.*
3. Create a new database---call it MyApp2.mdb
   This will establish Admin as the database owner.
4. Import all the objects from MyApp.mdb into MyApp2.mdb
   This will establish the user Admin as owner of all the objects, and give the Admins group *of Jane's system.mda* permission to change permissions on all objects created in MyApp2.mdb.
5. If you had any table relationships or attached tables, recreate them in MyApp2.mdb
6. Rename MyApp.mdb to MyApp.old and rename MyApp2.mdb to MyApp.mdb.

Variation #2: MyApp.mdb is secured using the steps under Securing an Existing Database.

To give MyApp.mdb to Jane, and have the Admins group *of her system.mda* get complete administrative authority over it, follow these steps:

1. Follow the instructions under Unsecuring a Secured Database, above
2. Follow the instructions under Variation #1, above.

For all intents and purposes, Jane is now the "owner" of MyApp.mdb and all objects in it, and Joe is just another user. Under both these scenarios, you can include the steps under Securing an Existing Database, above, as necessary to make the database secured under Jane's system.mda.

Additional Guidelines for Shipping a Secure Application

There is an intimate connection between each .mdb file and the SystemDB *in use when the database is created*. Members of the Admins group of that SystemDB will always have permissions to change permissions on any objects in the .mdb file. Thus it is impossible to protect your code from anyone who is in the Admins group of the SystemDB *in use when the .mdb was created*, and has a copy of the SystemDB. For this reason, you should NEVER release that SystemDB to anyone who you don't want to be able to read your tables, forms, modules, etc. Use the Change Workgroup utility to create a new SystemDB.

Miscellaneous Tidbits

The User() function in Microsoft Access Basic returns the name of the current user. This is not the same as the SID. As mentioned under Users and Groups, above, it is possible for 2 users in different SystemDB's to have the same name but different SIDs.

> **Gotcha:** This can cause a problem if 2 people that are using different SystemDB's have the same name but different PINs (and therefore different SIDs). For example, if you are using the User() function to log activity, and both are using your database, the activity logging could be inaccurate. These are actually 2 different accounts, even though their user names are the same. This is another reason that applications that attempt to implement their own security cannot be truly secure.

Changes in Access 1.1 and the ADK

Change Workgroup

The Change Workgroup tool now allows you to create a SystemDB. When creating the SystemDB, it allows you to specify the User and Company names, as well as a Personal Identification Number (PIN).

This completely removed the connection between the disks from which Access was installed and the SID of the Admins group. The PIN can contain any characters, and can be up to 20 characters long. To ensure that your SystemDB (and therefore the SID of the Admins group) couldn't be recreated by someone who has your original Microsoft Access setup disks, it is recommended that you create a new SystemDB before creating any secure databases.

Upgrade Setup

When using the Microsoft Access 1.1 Upgrade disks to upgrade an existing 1.0 installation to 1.1, the SID of the Admins group is not changed. In fact, the SystemDB is not touched in any way. This means that the SID of the Admins group is still tied to your original Microsoft Access 1.0 setup disks. You can use the Change Workgroup utility to change this, if desired.