Securing Access databases using Active Directory

Introduction

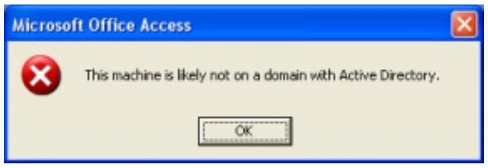
The new ACCDB format for Access databases was first introduced with Access 2007. This format is based on the new ACE database engine. Databases in MDB format (using the JET database engine) are still fully supported, but ACE seems to be the way forward. The new database format has many advantages, but it also left behind some features that were problematic with MDBs, chiefly among them there is no support for Replication, and no support for User-Level Security (ULS).

In my view the omission of ULS is a blessing. It is difficult to apply correctly, just check some messages in the microsoft.public.access.security newsgroup, and it was never very secure, just search the internet for password crackers. Since there is no replacement for ULS in ACCDB, we are free to design our own. Here I am presenting one solution. This solution will work with MDBs just as well. In fact the accompanying sample project is an MDB project so users of Access 2000 and up can use it as well. The code works unchanged in ACCDB. It is not meant to replace ULS and it does not attempt to duplicate its functionality, but it does address a common scenario, which is:

1. The application is deployed in a network with a Domain and Active Directory.
2. It is deployed as an ACCDE or MDE.
3. Users have their own workstation and user account.

Deployed in a Domain

The first item specifies that you must be on a network with a Domain and Active Directory (AD). That is because we will use AD to set up groups that users can be members of. If you are in a Workgroup or on a Standalone machine, this article is not for you. If you don't know if you are on a domain or not, run the sample application that is part of this application, and it will tell you:



This machine is likely not on a domain with Active Directory.

ACCDE or MDE

It is common practice for Access applications to be deployed as a compiled version. For the solution presented here to have a modicum of security it is required. Therefore you have to use Ribbon > Database Tools > Make ACCDE to turn your ACCDB into a compiled version. MDBs have a similar procedure. Compilation replaces the source code with compiled machine instructions that humans can't read. This is important so adventurous users cannot edit or bypass the security code we will be writing. Keep your ACCDB in a safe place so you can continue to make changes, each time deploying as ACCDE.

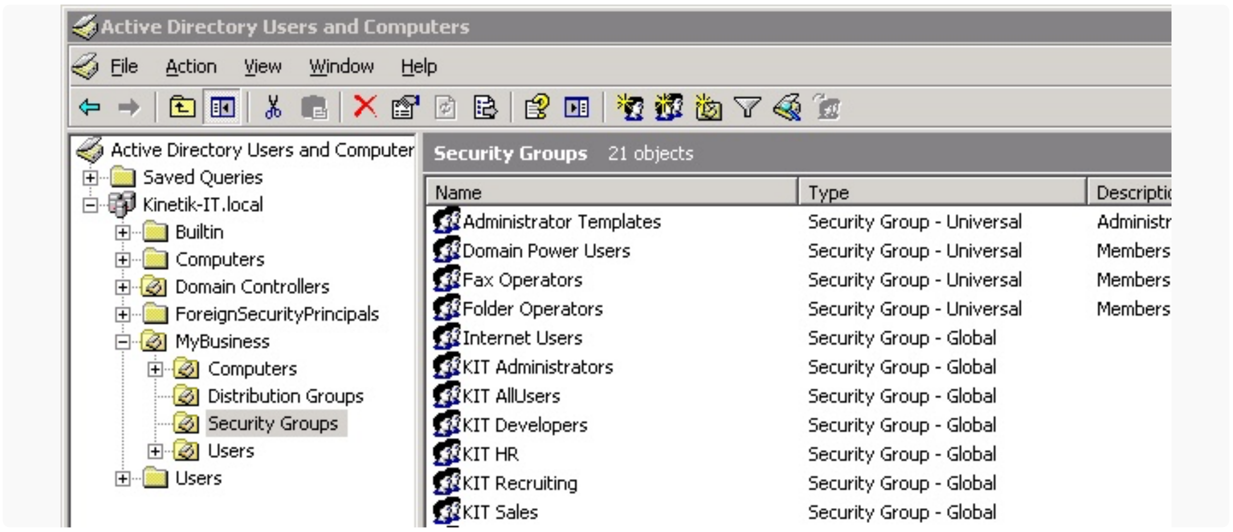
Own workstation and user account

The solution presented here depends on a Windows user's membership in various Active Directory groups. We will ask Windows who is logged in to the computer, and then ask Active Directory which groups this user is a member of. If several users with different security levels share the same computer in a common area, or if several users with different security levels login on several computers as the same Windows user, this solution is not for you.

Implementation

If you are still with us, your situation conforms to the three requirements outlined above. We can now use Active Directory to setup security groups for our application. For example Accounting people may have different access than Shipping people, except for Joe Smith who works in both departments. The main server in your domain, also called the Primary Domain Controller, typically hosts the Active Directory database. In large organizations it may be different and the IT people would know where it is hosted. If you don't know the name of your AD computer, run the sample application and it will display it.

In many organizations security groups (we are NOT talking about Distribution Groups in this article) are already setup, and you can use them as-is. Or you can create new groups specifically for this application and add users to those groups. On the AD computer there will be tools to review the Active Directory, create groups, assign users, etc. Depending on the operating system you may find these tools under Control Panel > Administrative Tools.



Name	Type	Descriptic
Administrator Templates	Security Group - Universal	Administr
Domain Power Users	Security Group - Universal	Members
Fax Operators	Security Group - Universal	Members
Folder Operators	Security Group - Universal	Members
Internet Users	Security Group - Global	
KIT Administrators	Security Group - Global	
KIT AllUsers	Security Group - Global	
KIT Developers	Security Group - Global	
KIT HR	Security Group - Global	
KIT Recruiting	Security Group - Global	
KIT Sales	Security Group - Global	

Once you have identified the groups needed for your application, you are ready to implement security. Typically security takes one of these forms:

1. some forms and reports are off-limits to some groups
2. forms are available but read-only
3. fields are visible or not

Off-Limits

To prevent a group from opening a form or report, we write code in that object's Open event. We take advantage of the Open event's Cancel property: if you set Cancel to True then the form or report will not open. Using this event rather than for example a Button_Click event is much better because you have to write the code only once. Regardless how your form is opened, the security check will happen. Here is an example:

```
Private Sub Form_Open(Cancel As Integer)
    Cancel = True  'Nobody can open this form.
End Sub
```

There is one somewhat unfortunate side-effect of setting Cancel to True: the calling routine will get a runtime error 2501 = The OpenForm action was canceled. We handle that with an error handler that ignores the error if it is number 2501:

```
Private Sub cmdForm2_Click()
    On Error GoTo Error_Handler

    DoCmd.OpenForm "frmTest_NoAccess"

Exit_Handler:
    Exit Sub

Error_Handler:
    If Err.Number = 2501 Then
        'Error 2501 = The OpenForm action was canceled.
        'Not really an error. Just indicates we were unable to open the form, which
is normal if we have insufficient rights.
    Else
        MsgBox Err.Description, vbCritical
    End If
    Resume Exit_Handler
End Sub
```

Read-only Access

To make a form read-only, Access has some form-level properties: AllowAdditions, AllowDeletes, and AllowEdits. By default these properties are set to True, indicating the user can add records, delete records, and edit records. In our code we can set these properties to False for users that don't belong to certain groups:

```
Private Sub Form_Open(Cancel As Integer)
    Dim blnHasAccess As Boolean
    Dim strGroupWithReadWriteAccess As String

    strGroupWithReadWriteAccess = "Administrators;Accountants"

    blnHasAccess = IsMemberOfSecurityGroups(strGroupWithReadWriteAccess)
    Me.AllowAdditions = blnHasAccess
    Me.AllowDeletions = blnHasAccess
    Me.AllowEdits = blnHasAccess
End Sub
```

The function IsMemberOfSecurityGroups in line 7 we will discuss in more detail below. For now it suffices to know that the function returns True if the user is a member of Administrators or Accountants.

Invisible Fields

To set individual fields to be visible or not, we again test for membership in certain groups, and if not, the controls' Visible property is set to False. See the sample code for an example. You have to be a bit careful with this. For example a control that has the focus cannot be set to invisible. In that case you first have to set the focus to a control that is always visible.

Code Reference

To start using the sample code in the attached project, copy these modules to your application: modAPI, modSecurity, and modUtilities. Also set a reference (code window > Tools > References) to "Active DS Type Library" (activeds.tlb). Choose Debug > Compile to make sure the code compiles.

modSecurity has several public functions that are used to apply security. The first one we already briefly saw in action above:

IsMemberOfSecurityGroups

Public Function `IsMemberOfSecurityGroups` (ByVal strGroupName As String) As Boolean

This function takes a groupname or a string with semicolon-separated groupnames, and returns if the user is a member of any of them. We typically call this function in the Form_Open or Report_Open event to check if the user is allowed to open the form or report.

GetSecurityGroups

Public Function `GetSecurityGroups` () As String()

This function returns an array with groups the user is a member of. This is the function that was used to fill the listbox on the Main Form of the sample application.

CacheSecurityGroups

Public Function `CacheSecurityGroups` () As Boolean

This function initializes an array in modSecurity with all groups the current user is a member of. Subsequent calls to IsMemberOfSecurityGroup or other security functions simply use this array, rather than calling Active Directory again. This is a performance optimization. You are encouraged to call CacheSecurityGroups in your startup code, but if you don't it will be called behind the scenes the first time you call one of the other functions.

GetADComputer

Public Function `GetADComputer` () As String

This function returns information about the computer that is hosting Active Directory.

Summary

In this post we presented a security framework for Microsoft Access applications (either MDB or ACCDB) based on security groups in Active Directory. The attached sample program provides all the needed source code and some usage examples.

AD_sample.zip

http://www.accesssecurityblog.com/post/Securing-Access-databases-using-Active-Directory.aspx