Global Information Assurance Certification Paper

Security in a Microsoft AccessÒ Database
(is it possible?)
A Step-By-Step Tutorial

GIAC Certification
GIAC Security Essentials (GSEC)
Practical Assignment
Version 1.4b

Prepared By
Diana Ralston

Submitted
February 18, 2004

Abstract

Databases store information, and many types of information stored in databases are of such a nature that they need to be kept secret from prying eyes. Financial data, customer lists, proprietary product information, payroll, and human resource records all fall within this category. One very popular, though sometimes belittled, database management product is Microsoft Access® (referred to as MS Access hereafter).

While no security measures can ever be guaranteed 100 percent of the time, the issue this paper will examine is the question of whether or not an MS Access database can be configured in such a way that it could be trusted with confidential data. Steps to improve MS Access security will be provided, along with examples of how to retrieve and update the secured data (using Microsoft Visual Basic code).

Overview

Several adjustments and methods can be applied to an MS Access database which improve the security significantly:

1. Windows permissions
2. Password protection
3. File encryption
4. Creating an MDE file
5. Startup options
6. File-level changes
7. Hiding tables
8. Workgroup administration

Each of these adjustments will be explained, illustrated, and evaluated separately. Since a complete tutorial on how to use MS Access is not within the scope of this paper, only those aspects dealing with security and the data residing in MS Access files will be included.

This paper is not intended to be an examination of reliability or robustness, nor is it meant to convince anyone that they should use MS Access as their database of choice. The sole focus of this paper is to present suggestions and techniques for improving what security is available with MS Access, for those users who have already made the choice. Also, this paper is not a tutorial on how to create and program an MS Access database; a certain level of competence in this subject is assumed, as is at least a working knowledge of programming methodology (preferably using Microsoft Visual Basic version 6.0).

One word of explanation for a term that will be used repeatedly throughout this paper – "Jet" is the term Microsoft uses for its version of what's referred to as a

"database engine", which is the software that performs the actual storage and retrieval of data for a database management system (DBMS).

Windows Permissions

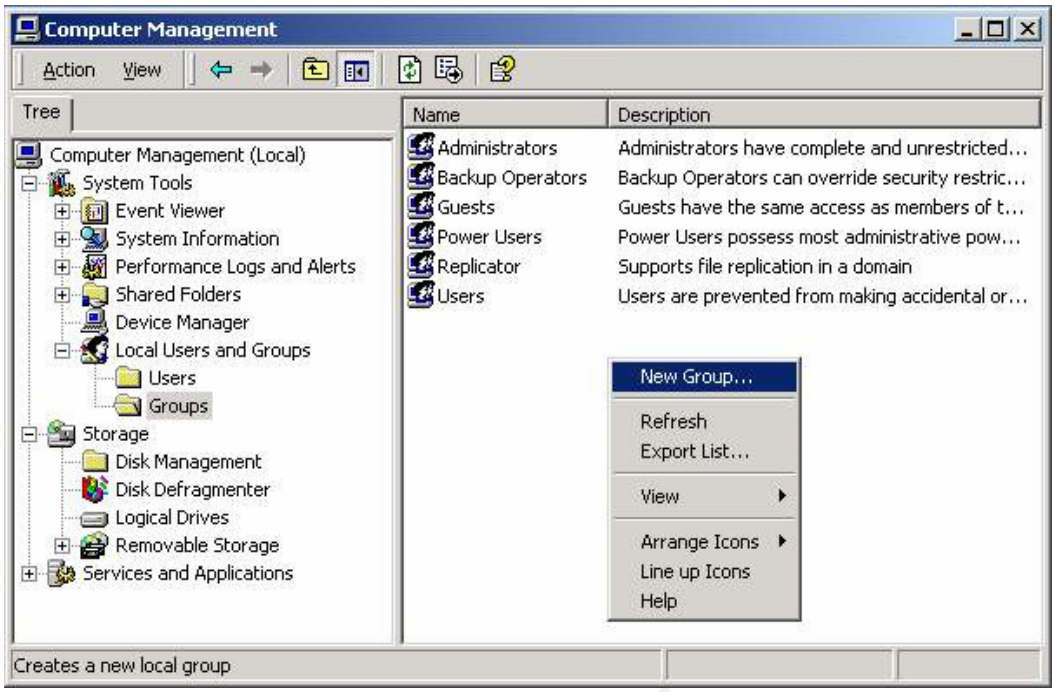
The best place to start in discussing database security is at the top. And that means to secure data at the operating system level.

Within Windows 2000 and Windows XP (if the hard drive is an NTFS partition) is a layer of security – folder permissions. What this means is that these operating systems allow you to limit which users have access to a folder and its contents and what exactly these users can do with the files in the folder. In this way, the security built into Windows can be leveraged to strengthen a database file's security.

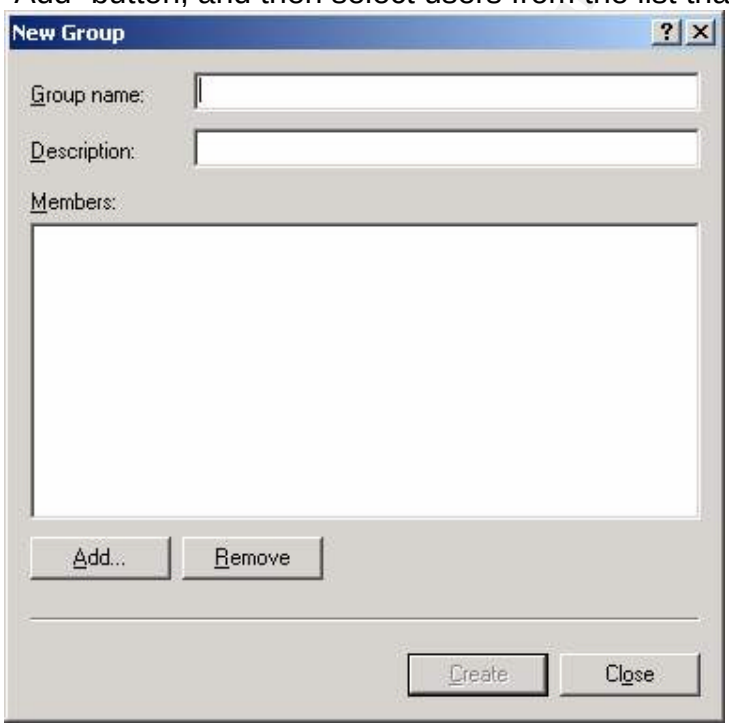
The way that folder permissions work in Windows is that users can be assigned to groups. Once groups of users are established, they can be assigned to the folder's list of who has permission to read or write or modify files in the folder. Assigning these permissions is a multi-step process which begins with setting up a group of users to access the database file's folder. Following is an illustration of how this process might be performed on a Windows 2000 system.

First, establish a user group, by following these steps:

1. Right-click on the "My Computer" icon located on the desktop
2. Click on "Manage"
3. Click on "System Tools"
4. Click on "Local Users and Groups"
5. Click on "Groups"
6. Right-click in an empty portion of the "Groups" area (right-hand side)
7. Click on "New Group…" (see below)



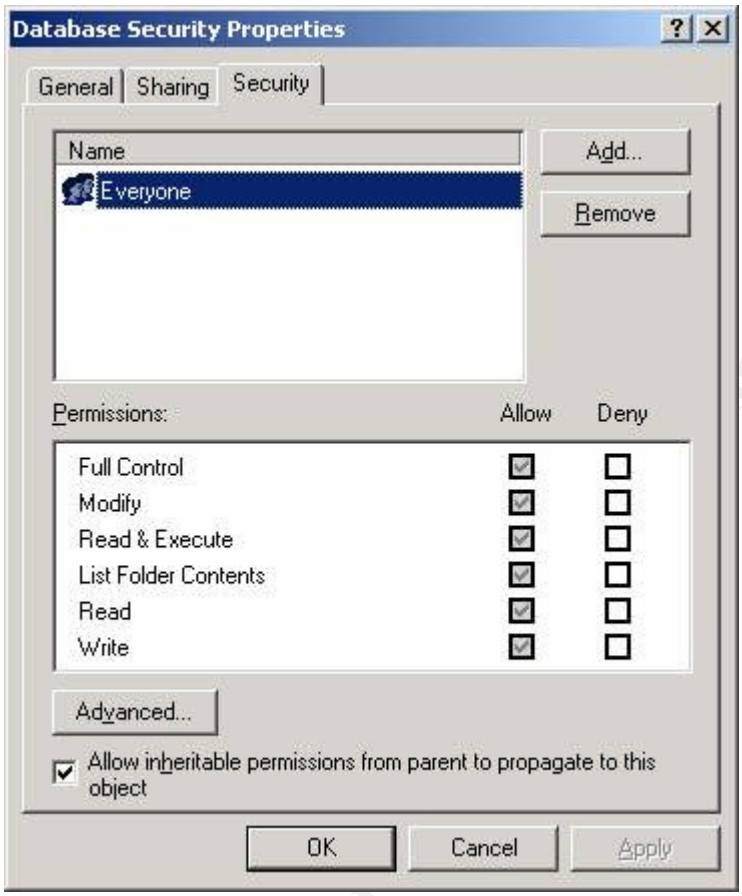
The next form that appears simply requires you to give the new group a name and then to add appropriate users to the group. The procedure for adding users to the new group is quite straightforward, requiring only that you click on the "Add" button, and then select users from the list that appears.



After setting up a group with appropriate users, the next task to be performed is to establish folder security and to add this newly-created group to the folder's permission list. In this example, the name of the folder being secured is "Database Security". Security setup is accomplished by doing the following:

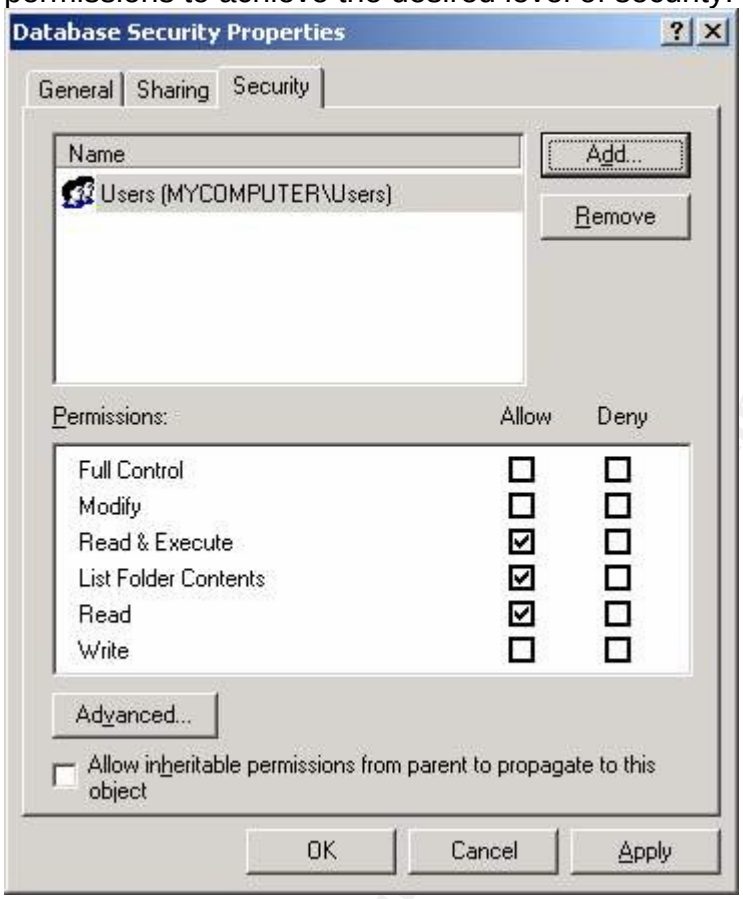
1. Right-click on the database file's folder
2. Click on "Sharing…"
3. Click on the "Security" tab

When setting up folder security for the first time, a form similar to the one below will appear. As shown, default access is set to allow all users to have full access to the folder.



Before adding the appropriate user group to this folder's permission list, the current permissions need to be cleared. To do this, uncheck the "Allow inheritable permissions to propagate to this object" option, which will cause the following form to appear.

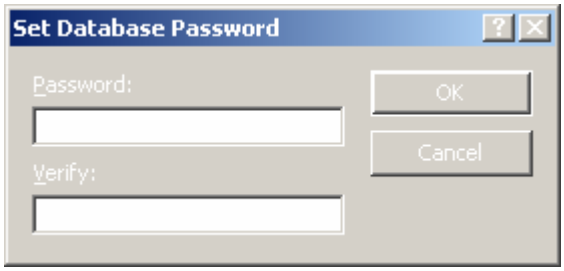
Click on the “Remove” button to clear all permissions for the folder. Then at this point, clicking on the “Add” button will allow addition of the new user group. The default permissions for a new user group is to allow “Read & Execute”, “List Folder Contents”, and “Read” access. Checking only the appropriate permissions to achieve the desired level of security.



One item that should probably always be selected as “Deny” is the “List Folder Contents” permission. If a user (or potential hacker) cannot see what files are in a folder, unauthorized access becomes much more difficult.

Password Protection

Once the database file’s folder has been secured at the operating system level, the files themselves need to be secured as well. The simplest level of Access database security is to set a password. To set a database password, open an existing database file for “exclusive” use, then click on the following in the toolbar: Tools → Security → Set Database Password. At this point, the screen below appears:



After entering a password for a database file, users will be prevented from "directly opening" the file without supplying the correct password. The meaning of "directly opening" (which also reveals the major weakness with exclusively utilizing this security method) is that users are prevented from opening the file and reading its contents "directly" in the MS Access program or from within a higher-level programming language using normal database-access methods.

The weakness of relying solely upon passwords is that files can still be opened and read using a text-editor program. Although data cannot always be directly read in this manner, it *can* be interpreted with a bit of work. Fortunately, this weakness can be overcome and will be explained in the next section.

As mentioned previously, opening a password-protected MS Access database directly requires that the correct password be provided before a user will be granted access. When opening a file from the MS Access program itself, the user will be required, after selecting the file, to immediately enter the password before proceeding any further.

Many times, however, rather than using the MS Access program to read and write data in a database file, an actual program will be written to perform these functions. The reason for creating a program is because a lot more can be done in a program compared to what can be accomplished directly in MS Access.

Satisfying the password-requirement from within a true programming language such as Visual Basic requires a bit more work. Two separate coding examples will be provided to illustrate different techniques – one for ADO (ActiveX Data Object and one for DAO (Data Access Object). As a short note of explanation for the uninitiated, ADO and DAO provide the programming framework for directly accessing and manipulating database objects.

Keep in mind that what these coding examples are meant to show is how to open a password-protected MS Access database file from within a program, *not* how to circumvent a database password (a simple text editor can accomplish that feat). Visual Basic and other languages still require you to provide the correct password in order to open a password-protected file, and the password must be presented in a very specific way – these coding examples are meant to show how to do that. Note: Sometimes the following coding methodologies can be difficult to find. In fact, one book claimed that it's actually impossible to open password-protected files using ADO, which is not true.

ADO (ActiveX Data Objects)

```
Dim db As ADODB.Connection
Dim ConnectString as String
On Error Resume Next
ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51” & _
                   “;Data Source=filepath\filename.mdb;Jet OLEDB” & _
                   “:Database password=password”
Set db = New ADODB.Connection
db.Open ConnectionString
If Err.Number > 0 then
   MsgBox “Unable to open database file”, vbExclamation
Else
   MsgBox “Database file was opened successfully”, vbInformation
End If
db.Close
Set db = Nothing
```

DAO (Data Access Object)

```
Dim db as Database
Dim ConnectString as String
On Error Resume Next
ConnectString = “;DATABASE=filepath\filename.mdb” & _
                “;PWD=password”
‘Open in non-exclusive mode and pass connection string
Set db = OpenDatabase(“ “, True, False, ConnectString)
If Err.Number > 0 then
   MsgBox “Unable to open database file”, vbExclamation
Else
   MsgBox “Database file was opened successfully”, vbInformation
End If
db.Close
Set db = Nothing
```

One final comment about using passwords, if you plan to store any passwords on the computer, do NOT store them in plain text anywhere, not even the Registry. A better solution is to encrypt the password before saving it on disk.

File Encryption

Which brings up the subject of encryption. Since an MS Access database can be opened and read with a plain text editor even with the advantages of password protection, another level of security should be added for truly sensitive data. Fortunately, another helpful tool is provided within the MS Access program. This

second tool is for data encryption. As the name suggests, using this tool will encrypt all data within an MS Access database.

To encrypt an existing database file click on the following in the toolbar: Tools → Security → Encrypt/Decrypt Database. The next screen that appears allows you to find and select the file you wish to encrypt. Once you select a file, another screen will appear requesting that you enter a new name for the file. At this point, you can either (a) enter a new file name, delete the old file, and rename the new encrypted file to the old file's name, or (b) you can enter the original filename and the program will then ask if you want to replace the existing file. Select "Yes" and your file will be encrypted.

The type of encryption used by MS Access is an RC4-encryption algorithm with a 32-bit key from RSA Data Security Incorporated. The probable reason that Microsoft chose such a short key was to comply with United States export laws, which allow a key length of less than 40 bits. Because of this less-than-ideal key length, however, MS Access encryption has been cracked numerous times. With that said, it is still sufficiently secure against all but a determined and experienced individual. Nonetheless, following are some recommendations and cautionary characteristics of using MS Access encryption:

- Since the Jet database engine will automatically decrypt data, whether a file is opened directly in MS Access or whether it's opened in a program, encryption should always be used in conjunction with a file-level password. This combination will, for most uses, alleviate the weaknesses associated with each method used separately.
- To protect extremely sensitive data such as social security and credit card numbers, a more robust encryption method could be used to encrypt individual data fields *before* submitting the data to Jet. This technique can even be used along with file-level passwords and MS Access-native encryption to provide an additional layer of security. One especially useful benefit of this technique is that these doubly-encrypted fields cannot even be read by opening the file in MS Access, even *with* the file-level password. In order to add this level of security, programming is required. Again, using a Visual Basic coding example, the following two lines show an example of how to programmatically encrypt then decrypt a single field:

  ```
  rs!CreditCardNumber = EncryptStr(ReadableCCNumber)
  ReadableCCNumber = DecryptStr(rs!CreditCardNumber)
  ```

  In this example, EncryptStr( ) and DecryptStr( ) represent programmer-created functions in which the appropriate code for performing the chosen procedures would be included (see Appendix A for an example of how these functions might be created).
- To explain why it would be beneficial to write a program to access data and manually encrypt data, rather than just using the MS Access program, consider these advantages for doing so:
  a) You can completely control the level and method of access because it's all determined by how you write your program.

b) You can choose the encryption technique which provides the degree of security you desire, rather than accepting what Microsoft provides.

Unfortunately, in spite of the benefits, there is also one small problem with encrypting individual database fields – it becomes more difficult to sort records based upon the contents of those fields. What this means is if, for example, you encrypted a payroll file's social security number in the manner just described, you wouldn't be able to use SQL to easily sort and display records in order by social security number. This type of deficiency can become more of a problem with large tables.

- Even though encryption is a valuable tool in the MS Access database security model, three drawbacks exist:
  a) Encrypted database files will be slower to open, read, and write (15 percent according to MS Access help).
  b) The file created will be larger, making a larger footprint on the disk.
  c) One final problem with MS Access' native encryption is the inability to compress the data using a tool such as WinZip.

Creating an MDE file

The subject of this section doesn't deal directly with protecting MS Access-stored data, although some articles and books seem to indicate otherwise. But since MDE files do have an impact on the overall concept of MS Access security, a brief explanation will be provided. An MDE file is, in essence, a compiled version of the database, or more specifically the source code portion of the database (i.e., forms, reports, and modules). Through the process of creating an MDE file the source code is converted to what's referred to as Pcode, which is a computer-readable set of instructions. Also during this process, the VBA (Visual Basic for Applications) which comprises the user-readable source code is then removed, which prevents users from modifying forms, reports, and modules.

To convert an existing database file to the MDE format click on the following in the toolbar: Tools → Database Utilities → Make MDE File. The next screen that appears allows you to find and select the file you wish to convert. Once you select a file, another screen will appear requesting that you specify the name for the converted file.

As previously mentioned, converting a database file to the MDE format does nothing to improve security for the underlying data in a file. But since it does "hide" any included functions to manipulate and display that data, it may have some worth as part of the overall security changes. If your database does not include any forms, reports, or modules, however, do not bother with this modification.

Startup Options

Before leaving the subject of MDE files, if you choose to allow users to access and update data through an MDE file, several additional steps should be taken to prevent those users from venturing into areas that would be dangerous. The way to stop these types of intrusions is to set the MS Access startup options so that users are forced into a specific area and denied access to other areas. Several steps are necessary to set startup options:

1) From the main database window in MS Access, click on the “Modules” tab, the “New” button, then enter the following code in the module window that appears:

```
Option Compare Database
Option Explicit

Function SetStartupProperties()
   ChangeProperty "AllowBypassKey", dbBoolean, False
End Function

Function ChangeProperty(strPropName As String, varPropType As Variant,
varPropValue As Variant) As Integer
   Dim dbs As Database, prp As Property
   Const conPropNotFoundError = 3270
   Set dbs = CurrentDb
   On Error GoTo Change_Err
   dbs.Properties(strPropName) = varPropValue
   ChangeProperty = True
Change_Bye:
   Exit Function
Change_Err:
   If Err = conPropNotFoundError Then  ' Property not found.
      Set prp = dbs.CreateProperty(strPropName, varPropType, varPropValue)
      dbs.Properties.Append prp
      Resume Next
   Else
      'Unknown error.
      ChangeProperty = False
      Resume Change_Bye
   End If
End Function
```

What the above does is to prevent users from utilizing the old Windows trick of holding down the Shift key while the program (MS Access in this case) is loading, which would override the program startup security we’re going to establish. Once the code is entered click on the “X” button in the upper right-hand corner of the module window, click on the “Yes” button when asked if

you want to save the changes, click on the "OK" button in the "Save As" window. Then you'll see a new module of code called "Module1" in the list.

2) The next step is to tell MS Access that there's a set of code to run when the database is first opened. This is accomplished by clicking on the "Macros" tab, clicking on the "New" button, clicking on the down arrow that appears on the first line of "Action" column. Once you do that, a drop-down list will appear; scroll down to find the action "RunCode" and click on it. At this point, an entry for "Function Name" will appear at the bottom of the current window – enter "SetStartupProperties()" in the "Function Name" entry field. This will tell MS Access to call the function shown above when the database first opens. Now, click on the "X" button in the upper right-hand corner of the Macro window, click on the "Yes" button when asked if you want to save the changes, enter a macro name of "AutoExec" on the "Save As" window, then click on the "OK" button. Then you'll see a new macro called "AutoExec" in the list.
3) The final step in defining the startup process is to indicate to MS Access which actions you wish to prevent users from being able to perform. To set the startup options: click on Tools → Startup… When the Startup window appears, click on the "Advanced>>" button so that all options are visible. Enter an application title, select a display form, then uncheck the following options:
   a) Allow Full Menus
   b) Allow Default Shortcut Menus
   c) Display Database Window
   d) Allow Built-in Toolbars
   e) Allow Toolbar/Menu Changes
   f) Use Access Special Keys

   Click on the "OK" button.

After completing these three steps, and creating an MDE file for the database, an application will exist which, when run, will only allow users to enter data in the form you provide. In addition, these users will be unable to perform more intrusive actions on the database.

File-Level Changes

This next suggestion is one designed to thwart only a very casual hacker, but can be a useful in concert with other recommendations. Changing filenames is not an incredibly effective security measure and perhaps could be better categorized as a form of subterfuge. The thrust of this idea is to change a file's extension from the standard ".mdb" to almost anything else. For example, if your database file is named "file1.mdb", you could rename it to "file1.xxx". What this one change will accomplish is that it will not be readily apparent for someone looking at the file in Windows Explorer that the file is an MS Access file. This is important for two reasons:

1. Potential hackers who are confident in their MS Access-cracking abilities may not notice your renamed file and may therefore never attempt to exploit it.
2. Double clicking on the file will no longer open the MS Access program.

Not only can consideration be give to changing a file extension, care should be taken in choosing entire file names. In other words, if your database file is called "EverythingYouWantToKnowAboutPayroll.mdb", it's going to attract attention. Perhaps choosing a more innocuous name would be advisable.

In addition to changing a file's extension, a file or the folder it resides in can be marked to the system as hidden. To "hide" a file or folder, right-click on it, click on "Properties", check the option for "Hidden", and click on "OK" or "Apply". This procedure is somewhat misnamed, however, because hidden files can still be viewed simply by selecting the folder option to "show hidden files and folders".

Hiding Tables and Files

A technique similar to marking a file as hidden can be employed to hide tables (or queries or reports) from being viewed within the MS Access program itself. This process is similar to hiding a file or a folder on disk. And just like files and folders, hidden tables can still be viewed by selecting the appropriate option within MS Access, but a casual snoop might not know how to do that. To hide an MS Access table, open the database file in MS Access, select the "Tables" tab, right-click on the table to be hidden, click on "Properties", check "Hidden", then click OK. The table will then disappear from the database window viewer. This may seem like a very effective technique until it becomes apparent how easy it is to view these hidden tables — click on the following in the toolbar: Tools → Options, then select the View tab and check the option to show "Hidden Objects".

Workgroup Administration

Workgroup administration is a subject similar to Windows permissions in that it describes a process of assigning usage permissions for MS Access database files. The main difference between these two usage assignment methods is that with workgroup administration, permissions are stored in a separate file over which greater control is available. This file, referred to as a workgroup file and created with an extension of ".MDW", is itself an encrypted MS Access database.

As will be discussed further in the next section, MS Access encryption and password protection is far from unbreakable, so it's not unreasonable to expect that workgroup files can be cracked in the same manner. In fact, several programs are available which purport to do exactly that. Nevertheless, assigning file access permissions with a workgroup file adds yet another layer of security to

the mix, in the hopes of creating enough hurdles to prevent all but the most determined and experienced hackers.

The following list, taken from a Microsoft support document, explains how to set up a workgroup file:

1. Use the Workgroup Administrator program (Wrkgadm.exe) to create a new workgroup information file. Write down the **Name**, **Organization**, and **WorkGroup ID** strings that you will be prompted for when you create your new workgroup information file and store them in a safe place. If your workgroup information file ever becomes lost or corrupted, you can reconstruct it by using these identical strings, which are then encrypted to create a unique token. Without a valid workgroup information file, you could conceivably be locked out of your database forever. Another reason to save this information is for upgrading an encrypted Access database to a newer version of Access. The recommended path for upgrading databases is to re-create the workgroup file in the new version of Access before upgrading the database itself.
2. The Workgroup Administrator automatically switches you to the new workgroup information file. Start Access, and open any database.
3. You will be logged on as a user named Admin. Use the **Security** menu options to add a password for the Admin user. The Admin user is the default account, and setting its password is what causes Access to prompt for a logon Name and Password the next time that you start Access.
4. Create a new user, which is the account you will use to help protect the database. Add this new user to the Admins group. Write down the strings that you use for the name and PID in case you ever need to re-create your workgroup information file. The PID is not the password-the string used for PID is encrypted, along with the string used for the Name, to create a unique token (SID, or system identifier) identifying the user.
5. Quit Microsoft Access and log back on as the new user account that you created in step 4. You will not have a password for this account yet, (the PID you typed with the name in step 4 is **not** the password), so now is a good time to set one.
6. Remove the Admin user from the Admins group so that Admin is a member only of the Users group. The Admin user account has no administrative powers built into it; they are derived from membership in the Admins group, which does. Although you cannot delete any of the built-in users or groups (Admin, Admins, and Users), you can move users to and from the Admins group and restrict permissions to the Users group.
7. Open the database that you want to help protect and run the Security Wizard. Select the objects that you want to help protect (it makes sense to select them all). The wizard will then create a new database owned by your new user, and will import all of the objects and relationships into it. It will also remove all permissions from the Admin user and the Users group

and encrypt the new database. The original database will not be altered. Note that the Access 2000 security wizard does not create a new database-it simply creates a backup copy of the original. One flaw with this arrangement is that not all permissions to open the database are removed from the Admin user and Users group to open the database, even though they appear to have been removed.

8. Open the new database. Because the Security Wizard removed all permissions from the Users group for the security-enhanced objects, you need to create your own custom groups and assign the level of permissions needed to these groups. Every user is required to be a member of the Users group (otherwise, a user would not be able to start Microsoft Access), so only grant permissions to Users that you want everyone to have. Members of the Admins group have irrevocable power to administer database objects, so make sure to limit membership in the Admins group to only those users who are administrators.
9. Create your own users and assign them to the groups that reflect the level of permissions that you want them to have. Do not assign permissions directly to users because that is extremely hard to administer. Users inherit permissions from the groups they are members of, and keeping track of the permissions assigned to a group is much easier than keeping track of the separate permissions of individuals. If a user is a member of multiple groups, then that user will have all of the permissions granted to any of those groups plus any permissions assigned specifically to the user (this is known as the "least restrictive" rule). There is no way to deny permissions to a user if that user is a member of a group that has been granted those permissions. If you need to create specific permissions for only a single user, create a group for that user and assign the permissions to the group; then, add the user to the group. The reason for this becomes clear when you consider that the user may quit, and you may have to set up permissions for the replacement on short notice.
10. Additionally, you may need manually to remove the **Open/Run** permission from the database container for the Users group through the security menus or through code. This will prevent someone from opening the database by using another workgroup information file or the default System.mda/mdw. In Microsoft Access 97, the User Level Security Wizard is supposed to remove the **Open/Run** database permissions for the Users group, but fails to do so. The Access 2000 Security Wizard removes permissions to the point where they are not visible on the security menus, but testing has revealed that in Access 2000 it is possible to open a database by using the default workgroup information file regardless of the menu settings. The cure for both versions of Access is to create a new, empty database while logged on as a member of the Admins group and import all of the objects from the security-enhanced database. You should take this step before spending too much time helping protect objects

because Access considers imported objects to be "new" and loses the permission information that was stored in the source database. [1]

Exploits

So far, several methods have been described to enhance MS Access data security. Unfortunately, several exploits are already available to circumvent most of these measures.

Passwords, while an important element of database security, are always stored in the same location in an MS Access database file; so even though they're encrypted, MS Access passwords have proven easy to extract. Several programs have already been written which will quickly and easily extract the password from any MS Access database file. One of these programs is free, so even cost isn't an issue for a potential hacker. Following are the web links of four programs which claim to be able to extract MS Access passwords. The only one I personally tested was the first (free) one, and it does work.
http://www.shatterock.com/products/software/dbpwd/
http://lastbit.com/access/default.asp
http://www.lostpassword.com/access.htm
http://www.ozgrid.com/Services/access-password-recover.htm

As could probably be expected, methods of cracking the encryption used in MS Access have also been devised. Reportedly, MS Access uses a very simple algorithm, behaving as a stream cipher where the stream is XORed with the database. With this knowledge available, following is one theoretical method of solving the MS Access encryption puzzle:

- Create a known database which is at least as large as the database you are trying to break
- Encrypt it
- Find the XOR between the known database and its encryption. This is the key stream.
- XOR the key stream against the target database you are trying to break[2]

Keep in mind, however, what was stated previously about encryption not being a huge impediment, as long as you know the password. Therefore, this latest exploit, while informative, is probably unnecessary if a potential hacker can use one of the password-hacks listed previously to open up the database file completely.

[1] Chipman, Mary, et.al.
[2] Rosen, Mark.

In Appendix A is a program example showing how XOR encryption and decryption work. This section of code could be used as a simple stand-alone encryption mechanism, as a means to perform the first step in double encrypting data (as mentioned previously), to learn how this form of encryption works, or as a starting point to write an improved encryption function.

In Appendix B is an explanation of and an example showing how to use a free RC4 encryption library which can be downloaded from the Internet (http://www.rdg.mirror.ac.uk/sites/ftp.wiretapped.net/pub/security/cryptography/algorithms/rc4/RC4Lib.zip/RC4.dll?extract=true). The RC4 library is a variable key-size cipher written by Ron Rivest for RSA. Variable key size means that, by using the RC4 library, you can choose a larger key size than the 32-bit key selected by Microsoft for Access. Doubtless, other encryption utilities (some probably more secure than RC4) are available for the do-it-yourselfers who want to try encrypting or double encrypting data. This particular library was chosen (a) because it's easy to use and (b) it's free.

Conclusions

Several strategies have been presented to make the best of an imperfect situation, which is to superimpose some measure of security onto a database manager that is not ideally suited for that purpose. Even if all the techniques which have been explained are used, a skillful and determined hacker could probably view everything you're trying to protect.

In situations where data security is an absolute priority, MS Access is not the best choice. Other products are available which can provide much higher levels of security. However, for those who have already made a commitment to using MS Access, the steps outlined in this article can significantly improve your chances of keeping private information hidden.

Appendix A

```
Public Function EncryptStr(strInData As String, strPassKey As String) As String
' ------------------------------------------------------------------------------------
' Routine only encrypts data.  To decrypt data, call the DecryptStr
' routine using the same PassKey you used here.  The data returned
' contains only printable Hex characters.
' ------------------------------------------------------------------------------------
   Dim strHexChar As String, strNewDataStr As String, strAscPassKey As String
   Dim intPassKeyChar As Integer, intInDataChar As Integer
   Dim intNewAscChar As Integer
   Dim i As Long          ' Used to parse the incoming data string
   Dim j As Integer       ' used to parse the converted password key
   ' Initialize variables
   j = 0
   strAscPassKey = ""
   strNewDataStr = ""
   ' If pass key is empty then use the password in reverse
   If Len(strPassKey) = 0 Then
      For i = Len(strInData) To 1 Step -1
         strPassKey = strPassKey & Mid(strInData, i, 1)
      Next
   End If
   ' Parse the password key and convert each character to its
   ' ASCII decimal value and append to a new string
   For i = 1 To Len(strPassKey)
      strAscPassKey = strAscPassKey & Asc(Mid(strPassKey, i, 1))
   Next
   ' Start parsing the incoming data string and encrypting it
   For i = 1 To Len(strInData)
      ' Increment the parsing counter
      j = j + 1
      ' Parse thru the new password key string and convert one char
      ' at a time in the input string to its ASCII decimal value
      intPassKeyChar = Asc(Mid(strAscPassKey, j, 1))
      'If password counter equals or exceeds the length of the temp
      ' password string then reset the counter back to start at the
      ' beginning of the string again
      If j >= Len(strAscPassKey) Then j = 0
      ' Convert one char at a time in the input string to its ASCII decimal value
      intInDataChar = Asc(Mid(strInData, i, 1))
      ' Use the Xor operator to perform logical exclusion on two expressions.  The
      ' Xor operator performs as both a logical and bitwise operator.  A bit-wise
      ' comparison of two expressions using exclusive-or logic to form the result,
```

```
        ' as shown in the following example:
        '  x = 1000 XoR 1110
        '       1000
        '       1110
        '      --------
        '  x = 0110   Same values become "0" else become "1"
        intNewAscChar = intPassKeyChar Xor intInDataChar
        ' Convert the new value to Hexidecimal string
        strHexChar = Hex(intNewAscChar)
        ' If after hex conversion the length is only one
        ' char then prefix it with a zero
        If Len(strHexChar) < 2 Then strHexChar = "0" & strHexChar
        ' Append the new character to the newly encrypted string
        strNewDataStr = strNewDataStr & strHexChar
    Next
    ' Return the newly encrypted string
    EncryptStr = strNewDataStr
End Function

Public Function DecryptStr(strInData As String, strPassKey As String) As String
' ------------------------------------------------------------------------------------------
' Routine only Decrypts data.  Use the same PassKey as in EncryptStr.
' ------------------------------------------------------------------------------------------
    Dim strTmp As String, strHexChar As String
    Dim strNewDataStr As String, strAscPassKey As String
    Dim intPassKeyChar As Integer, intInDataChar As Integer
    Dim intNewAscChar As Integer
    Dim i As Long        'Used to parse the incoming data string
    Dim j As Integer     'used to parse the converted password key
    ' Initialize variables
    j = 0
    strAscPassKey = ""
    strNewDataStr = ""
    ' If pass key is empty then use the password in reverse
    If Len(strPassKey) = 0 Then
        For i = Len(strInData) To 1 Step -1
            strPassKey = strPassKey & Mid(strInData, i, 1)
        Next
    End If
    ' Parse the password key and convert each character to its
    ' ASCII decimal value (numeric) and append to a new string
    For i = 1 To Len(strPassKey)
        strAscPassKey = strAscPassKey & Asc(Mid(strPassKey, i, 1))
    Next
```

```
    ' Start parsing the incoming data string and decrypting it
    For i = 1 To Len(strInData) Step 2
       ' Increment the parsing counter
       j = j + 1
       ' Parse thru the new password key string and convert one char
       ' at a time to its ASCII decimal value
       intPassKeyChar = Asc(Mid(strAscPassKey, j, 1))
       ' If password counter equals or exceeds the length of the temp
       ' password string then reset the counter back to start at the
       ' beginning of the string again
       If j >= Len(strAscPassKey) Then j = 0
       ' Convert one char at a time in the input string to its ASCII decimal value
       strHexChar = Mid(strInData, i, 2)
       ' Convert the Hexidecimal string into an integer
       intNewAscChar = CInt("&H" & strHexChar)
       ' Use the Xor operator to perform logical exclusion on two expressions.  The
       ' Xor operator performs as both a logical and bitwise operator.  A bit-wise
       ' comparison of two expressions using exclusive-or logic to form the result,
       ' as shown in the following example:
       '  x = 1000 XoR 1110
       '       1000
       '       1110
       '       -------
       '  x = 0110   Same values become "0" else become "1"
       intInDataChar = intPassKeyChar Xor intNewAscChar
       ' Append the new character to the newly decrypted string
       strNewDataStr = strNewDataStr & Chr(intInDataChar)
    Next
    ' Return the newly decrypted string
    DecryptStr = strNewDataStr
End Function
```

Appendix B

After installing RC4.dll on your system, open a new project in Visual Basic, click on the following in the toolbar: Project → Reference, then select "RC4 Stream Cipher Library" from the list.

```
Dim RC4 As New RC4
Dim EncryptString as String, KeyString as String
EncryptString = "This is a sample string to encrypt and decrypt"
KeyString = "My test key string"
' Example of encrypting the characters in EncryptString
EncryptString = RC4.Encrypt(EncryptString, KeyString)
' Example of decrypting the string we just encrypted
EncryptString = RC4.Decrypt(EncryptString, KeyString)
Set RC4 = Nothing
```

To explain further what the above-listed code is doing:

1. All program definitions, including one to execute the RC4 library are defined
2. The variable EncryptString is loaded with text that the program will encrypt and then decrypt
3. The variable KeyString is loaded with a value that will be XORed against EncryptString to create encrypted text (note: and this is VERY important, the same KeyString value that was used to encrypt the text must be used to decrypt it again
4. Next, the text is encrypted. After encryption, examining the contents of EncryptString would show the following gibberish string of characters:
   OÇÐ›H½Y‰aø òaùÍôC h³ Kk€3h,QÚ¬ÃÍ+ s¶4HD&ÚÝk ßÄ
5. After encryption, the gibberish text string is decrypted back to the original value
6. Finally, to clean up, our newly created RC4 object is destroyed

Bibliography

Chipman, Mary, Baron, Andy, Bell, Chris, Kaplan, Michael, Litwin, Paul, Torrico, Rudy. "Access Security FAQ." Version 2.42. Oct 2000. URL: http://support.microsoft.com/default.aspx?scid=/support/access/content/secfaq.asp (17 Feb 2004).

Gearhart, Bill. The ASP Emporium. "Nine Reasons NOT To Use MS Access To Power A DB-Driven Website." No date. URL: http://www.aspemporium.com/aspemporium/tutorials/dontusemsaccess.asp (17 Feb 2004).

Hope, Mary H., Sawkins, Julian. University of Derby, Computing Services, Policy and Strategies. "Microsoft Access, when to use it and when not to use it." 19 Jun 2003. URL: http://www.derby.ac.uk/computing-services/AccessSharedApps.doc (17 Feb 2004).

McManus, Jeffrey P. Database Access with Visual Basic 6. Indianapolis: Sams Publishing, January 1999.

No author. "ACC: How Microsoft Access Uses Encryption." Version 2.0. 07 May 2003. URL: http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/q140/4/06.asp&NoWebContent=1 (17 Feb 2004).

No author. "Chapter 14: Securing Your Application." 16 Jan 1997. URL: http://www.microsoft.com/accessdev/articles/bapp97/chapters/ba14_1.htm (18 Feb 2004).

No Author. "Gauging your security needs, alternatives to Access/JET security." No date. URL: http://www.tek-tips.com/gfaqs.cfm/pid/181/fid/3893 (17 Feb 2004).

No author. "Why Choose Microsoft Access database for your office automation needs?" No date. URL: http://www.blueclaw-db.com/microsoft_access.htm (17 Feb 2004).

No author. "Database Engine." No date. URL: http://www.webopedia.com/TERM/D/database_engine.html (17 Feb 2004).

Redondo, Alvaro. "RC4 Stream Cipher Library 1.00." No date. URL: http://www.rdg.mirror.ac.uk/sites/ftp.wiretapped.net/pub/security/cryptography/algorithms/rc4/RC4Lib.zip%5Bpeek%5D (17 Feb 2004).

Robinson, Garry. "Real World Microsoft Access Database Protection and Security." New York, New York: Apress, 2004.

Rosen, Mark. "Thoughts on the Next Target." 06 Jul 1997. URL: http://www.privacy.nb.ca/cryptography/archives/cryptography/html/1997-07/0033.html (17 Feb 2004).