

Bunker Hill

501 Delancey Street
San Francisco, CA 94107
(415) 512-0689
www.bunkerhill.com

Microsoft Access Applications and Sarbanes-Oxley Compliance

*By Ron Tornambe, CTO
Allen Dutra, CFO/CPA*

Date 9/24/2006

Contents

Introduction	2
Problem Statement	2
Options	2
Bunker Hill Solution	3
Implementation	3
Summary	6

Introduction

This paper presents a practical and expedient approach to converting Microsoft Access applications to comply with Sarbanes-Oxley regulations.

Problem Statement

The widespread use of Access applications in most organizations presents management with considerable operational and security challenges.

The Sarbanes-Oxley Act has far-reaching implications on the use and perhaps future role of Microsoft Access applications in the organization.

Data stored in Microsoft Access Jet databases is inherently unsafe.

Even the most secure Access databases, those that employ user-level security and encryption, are easily compromised. There are a number of companies that offer inexpensive software that retrieves “lost or forgotten” Access user and password information, even for encrypted Access databases, effectively ‘cracking’ them open.

Organizations dedicated to SOX compliance are faced with the task of identifying all Microsoft Access applications that contain sensitive data and taking corrective action to prevent unauthorized use.

Options

Since Access data cannot be protected, it must be stored in a secure DBMS repository, like DB2, Oracle or SQL Server. There is no way around this requirement.

The following is a non-exhaustive list of alternatives for securing Access applications.

Date

Bunker Hill

Find a Replacement

Purchase software from third party vendors that meet the functional and security requirements of the existing Access applications.

Redevelop the Application

In some cases it might make sense to just “bomb, bulldoze and start over” as Buckminster Fuller once remarked when asked about what he would do to improve Los Angeles.

.Net or Java Conversion

Convert the Access applications to IT standard technologies, like Java or .NET and Oracle or SQL Server, while retaining the basic database design and front-end interface.

Retain the Access Front-End

Convert the MS-Access database to a secure DBMS platform and modify the application front-end to operate with the DBMS and conform to SOX compliance standards.

Bunker Hill Solution

This paper will focus on the last option for the following reasons:

Benefit 1

Saves time and resources; requires far less development time than any other alternative.

Benefit 2

Least disruptive to normal day-to-day operations; involves no user retraining or learning curves nor major shifts in development staffing or technologies.

Benefit 3

Functionality that implements SOX compliance provisions can be added to Access applications as easily as ones developed using other frameworks, like .Net or Java.

Implementation

I. MS-Access Database Conversion

The first step is to convert the MS-Access database to a secure DBMS.

The following is a high-level overview of the conversion process.

The Access design is replicated using the target DBMS Data Definition Language constructs and data is loaded. To optimize performance, Access queries may be converted to views and/or procedures on the target DBMS.

In order to retain the functionality of the original Access application front-end, the DBMS tables are “linked” into Access using ODBC, replacing the original Access tables. Any converted queries are integrated into the new Access application, preserving the look-and-feel and operation of the original GUI.

Some may argue that using ODBC is less secure than ADO/OLEDB since some connection information is stored in the Windows registry and can be viewed using the ODBC Administrator. The only pertinent saved connection information is the Server Name which is not much of a clue for those attempting a break-in. For those who remain unconvinced, the ODBC DSN registry entries can be created by the application program just before the initial connection is established and removed directly afterwards.

Bunker Hill

II. Define a Security Strategy

For an application to be secure, it must ensure that all data it owns can only be changed using its GUI. A complete audit trail of all changes made by users of the application must be recorded.

The security approach presented here is based on the concept of “Application Users” (AU) and the “Application Supervisor” (AS).

The security strategy can be applied to any application and is not limited to Sarbanes-Oxley.

Note that if the application connects to the DBMS over an extranet, a corporate VPN or other DBMS provided secure socket layer must be utilized to protect passwords and data from “sniffers” (programs that inspect data traveling over unsecured links).

It is assumed that only the DBA or Password Management group has full control of assigning User-IDs and Passwords and that measures that provide accountability are in place.

Application Users

The Application Users (AU) group are assigned database User-IDs with roles that do not grant any privileges to application related objects (tables, views, etc.). They are only assigned “Execute” privilege on a single stored procedure.

Application Supervisor

The Application Supervisor (AS) account is setup specifically for use

with an Access application. It should never be used outside of the application. A role that grants all privileges required to operate the application and update the database is assigned to the AS.

The AS Password (PWD) is stored using modern cryptography algorithms supplied by the DBMS vendor that are specifically designed to prevent hackers from deciphering data.

Since the application retrieves the AS PWD at runtime, the PWD can be changed as frequently as desired to minimize the risk of intrusion.

Application User Login Form

A new “Application User Login” form is added to the Access application and is designated as the form that opens when the application is started.

The form accepts the UID and PWD of Application Users. After the AU is authenticated, the program calls the aforementioned stored procedure that returns the AS UID and decrypted PWD.

The AS UID and PWD are used to supply ODBC connection information required by the Access application.

Note that User-IDs and Passwords are not stored anywhere within the source code or design objects of the Access application or in the Window’s registry or file system.

After the AS connection is established, an entry is added to a table that contains the application’s session-id and the AU UID. The triggers that implement the audit trail will use this information to associate the AU with the transaction details and prevent unauthorized users from updating data.

Bunker Hill

Application Users are then able to operate the application with full privileges.

III. Implement an Audit Trail

An essential component of the security strategy is the implementation of an audit trail that logs all AU database insert, update and delete transactions.

To implement the audit trail, a new table will need to be created for each application table. The new table will include all fields from the associated application table as well as OperationCode, AU UID, Session-ID, and Timestamp. Since the audit tables do not include any keys, indexes, or constraints, inserts will require minimal overhead.

Triggers must be created for insert, update and delete operations on all tables. The triggers ensure that users belong to registered applications and save all data pertaining to the transaction in the associated audit table.

The OperationCode will be set to 'I' or 'D' to denote the type of transaction saved; Insert or Delete. Note that database Update transactions are implemented by adding both Delete and Insert records to the associated audit trail table. The Delete record contains the data before the update is applied and the Insert contains the updated record.

The Application User's User ID associates the audit trail transactions

with the AU responsible for the changes.

An example of a trigger that implements the audit trail follows. This example is implemented using DB2, but is also easily implemented using MS-SQL and Oracle.

After an application connects to the DBMS, it registers itself by adding an entry to the following table that records the session-id of the application and the AU UID. Note this entry is removed from the table when the application is closed.

```
CREATE TABLE NORTHWIND.APPLICATION_USERS
(
  APPL_ID VARCHAR(128),
  USER_ID VARCHAR(40)
)
;
```

The audit trail table associated with the application's SHIPPERS table follows.

```
CREATE TABLE NORTHWIND.AT_SHIPPERS
(
  OP_CODE      CHAR,
  USER_ID     VARCHAR(40),
  APPL_ID     VARCHAR(128),
  TRANS_TIME  TIMESTAMP,
  SHIPPERID   INTEGER,
  COMPANYNAME VARCHAR(40),
  PHONE       VARCHAR(24)
)
;
```

Insert Trigger Example

The triggers serve as gatekeepers of application data. Only transactions from currently registered application users are accepted.

```
CREATE TRIGGER NORTHWIND.AT_SHIPPERS_INS
AFTER INSERT ON NORTHWIND.SHIPPERS
REFERENCING NEW_TABLE AS INS_TABLE
FOR EACH STATEMENT MODE DB2SQL
BEGIN ATOMIC
  DECLARE APPUSERID VARCHAR(40);
  SET APPUSERID = (
    SELECT USER_ID
    FROM NORTHWIND.APPLICATION_USERS
    WHERE APPL_ID=APPLICATION_ID()
  );
  IF APPUSERID IS NULL THEN
    SIGNAL SQLSTATE '75001'
    ('Connection expired due to inactivity.
    Application must be restarted.');
```

Bunker Hill

```
ELSE
  INSERT INTO NORTHWIND.AT_SHIPPERS
  SELECT 'I',
        APPUSERID,
        APPLICATION_ID(),
        CURRENT_TIMESTAMP,
INS_TABLE.*
  FROM INS_TABLE;
END IF;
END
;
```

The trigger fires when an insert operation is issued for the SHIPPERS table. It first attempts to retrieve the USER_ID from the APPLICATION_USERS table by supplying the session-id of the user that initiated the transaction. If the session-id does not exist, then either the user's ODBC connection has expired or an unauthorized user has attempted to tamper with data. In either case, the transaction is canceled; the database is not altered and no audit trail entry is recorded. If the session-id is found, an entry is added to the associated audit trail table.

IV. Secure the Access Application

Although creating an Access MDE file removes all editable source code and disables viewing the particulars of Access Form and Report designs, additional actions must be taken to secure Access applications from unintended and potentially malicious uses.

Most Access applications include built-in menus and toolbars that include items that can be used to compromise security. For example, the link to the "Startup" form can be used to alter options that were set to prevent application misuse. Users

must also be prevented from displaying the Database Window, which exposes database table and query design information that may contain User-IDs or even Passwords.

As a general rule, any built-in menu items or buttons that are not essential to the operation of the application should be removed.

Since ODBC implements connection pooling, connections expire after they have been idle for 60 seconds. The default can be changed by setting the OPTimeout entry of the ODBC driver in the Window's registry.

Although setting the OPTimeout value to a short interval will prevent unauthorized persons from commandeering users' workstations while unattended, timeouts that require users to repeat the logon procedure may be more frequent and prove counterproductive.

Summary

By implementing the steps outlined in this paper, Microsoft Access applications can expediently and cost-effectively be converted to comply with Sarbanes-Oxley regulations.

Since the Access databases are converted to robust DBMS, the additional benefits of scalability, improved disaster-recovery, and manageability are also realized.