Table of Contents

- Evolution of Data Security
- New Challenges
- Securing both code and data
- Row Level Security
- Generic example
- Adventure Works
  - Example 1
  - Example 2
- Conclusion
The evolution of data security is not over

- Complex architectures
  - 1970 – Democratization of the Mainframe
  - 1980 – 2 tier applications relying on Mainframe
  - 1990 – 3 tier applications with multiple databases (introduction of EAI middlewares)

- First IT generation
  - First IT professional with 100% of carrier in IT are retiring now

- New public / end users with new channels
  - Human to Human
  - Kiosk and Voice Response Unit to Humans, Corporate servers to others (APIs)
  - Internet to Humans
  - Phones, etc to Humans
New challenges are still coming

- Shared infrastructure
  - Mainframes to multiple users
  - Multi-tiers applications create Identity Management challenges

- Data accessed by Internet Users
  - First, public information to the public (ex: corp websites)
  - Private information to customers / patients (ex: MyUHC.com)
  - Private information to public (ex: mypace)

- Outsourced IT can create risks for corporations
  - Longer lifespan of applications
  - High turnover of IT professional
Securing code and data

- First, Security is at the application level
  - Introduction of RACF and ACF2 limits access to screens

- Second, secured object oriented coding
  - Security is at the object level
  - Users and systems have access to objects

- Third, secured data repositories
  - System IDs limit read / write to entire tables, even columns

Do we need more data Security?
Well what if…

- We could limit access at the data level?
- We could have a solution that leverages existing and simple database protocols?
- We did not need to add another complex layer of security that will require resources to administer?
- We could use a technique that will be impermeable to IT Staff changes and won’t slow down upgrade projects?
What is Row Level Security?

- One definition
  - A method of providing another level of access security in a database by exploiting existing business data

- Row Level Security is not new.
  - Oracle provides RLS as a feature (Labels Security)
  - PeopleSoft has embedded features for RLS
  - Business Objects has numerous white papers

- This presentation explores a generic way of implementing RLS by
  - Restricting user access to data based on data in the row,
  - Keeping the content of business tables unchanged
  - Not affecting application or presentation developers regardless of how users access the data.
Available approaches don’t meet our requirements. For example:

- “Implementing Row Level Security in SQL Server Databases” by Narayana Vyas Kondreddii recommends addition of user id as a column on secure tables.
  - [http://vyaskn.tripod.com/row_level_security_in_sql_server_databases.htm](http://vyaskn.tripod.com/row_level_security_in_sql_server_databases.htm)

- Rask, Rubin, and Neumann offer on the Microsoft Technet site a solution based on defining views that again requires base table modifications.

- Kemal Erdogan presents a promising solution based on lookup tables. That doesn’t require base table changes but leaves the tables unsecured in the case of direct user database access.
Provisos and Quid Pro Quos

- SQL Server database (MS SQL Server 2000, 2005, or 2008)
- An attribute exists in common to all tables to be secured that makes sense as a determinant of who sees what data (in the example, department id)
- Application calls passed to the database are secured by individual user id, not by a single admin user id
- We’ll show only Select security; the concept can be extended to cover Update and Insert statements
- The solution presented is not optimized
  - Performance in your environment will depend on its unique characteristics
Generic Example: SQL Server Table Definition (Slide 1 of 3)

- Overall Approach: add a cross reference table that links userids to the security attributes.

<table>
<thead>
<tr>
<th>PK</th>
<th>UserID</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PK</th>
<th>OrderID</th>
<th>CustomerName</th>
<th>OrderTotal</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PK</th>
<th>Department</th>
<th>ParentDepartment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creating an RLS function step 1: Protect data with Table Valued Functions requiring

```sql
CREATE FUNCTION [adhoc].[u_GetOrderSummary] ()
RETURNS TABLE
AS
RETURN
(
    SELECT OrderCount, Receipts
    FROM dbo.GetOrderSummary(Current_User)
)
```

The problem: the user could key any user’s id as a parameter to circumvent security
Generic Example: SQL Server Table Definition (Slide 3 of 3)

- A solution: Prevent user logins to the application database, but enable them to a separate database that contains table valued functions that call those requiring user ids as parameters, as follows:

```sql
CREATE FUNCTION [adhoc].[u_GetOrderSummary] ()
RETURNS TABLE
AS
RETURN
(
    SELECT OrderCount, Receipts
    FROM dbo.GetOrderSummary(Current_User)
)
```
**RLS in a reasonably complex database:**

**The Adventure Works Examples**

- The database Adventure Works is shipped in every MS SQL server application as an example.
  - It represents a company called Adventure Works
  - Business processes are all modeled and include (and is not limited to):
    - Sales
    - Production
    - HR
    - Ordering

- Two examples of Adventure Works RLS have been developed:
  - A sales person can only sell in his/her territory
  - HR professionals can only see data for employees in their assigned departments
A sales person can only sell in his/her territory (Slide 1 of 4): The Problem and Strategy

- **What we are trying to solve:**
  - Right now, all Sales resources perform a sale in every territory.
  - The new rule is that one can only sale in its own territory.

- **What we are going to do:**
  - We create a function that links user ID to the Territory.
  - We create a view to prevent the user from inserting a different user ID than his
A sales person can only sell in his/her territory (Slide 2 of 4): The Data Model
A sales person can only sell in his/her territory (Slide 3 of 4) The Function

-- based on Sales.vSalesPersonSalesByFiscalYears

TABLE VALUED FUNCTION
CREATE FUNCTION [Security].[ufnGetSalesTotals]
(@UserId VARCHAR(20))
RETURNS TABLE
AS
SELECT
    pvt.[SalesPersonID],
    pvt.[FullName],
    pvt.[Title],
    pvt.[SalesTerritory],
    pvt.[2002],
    pvt.[2003],
    pvt.[2004]
FROM (SELECT
    soh.[SalesPersonID],
    soh.[SubTotal],
    c.[FirstName],
    c.[LastName] AS [FullName],
    e.[Title],
    st.[Name] AS [SalesTerritory],
    YEAR(DATEADD(m, 6, soh.[OrderDate])) AS [FiscalYear]
FROM [Sales].[SalesOrderHeader] soh
    INNER JOIN [Sales].[SalesPerson] sp
    ON sp.[SalesPersonID] = soh.[SalesPersonID]
    INNER JOIN [Security].[SalesTerritory] st
    ON sp.[TerritoryID] = st.[TerritoryID]
    INNER JOIN Security.SalesAccess sa
    ON sa.TerritoryID = st.[TerritoryID]
    AND sa.UserId = @UserId
    INNER JOIN [HumanResources].[Employee] e
    ON soh.[SalesPersonID] = e.[EmployeeID]
    INNER JOIN [Person].[Contact] c
    ON e.[ContactID] = c.ContactID
) AS soh
PIVOT
( SUM([SubTotal])
FOR [FiscalYear]
IN ([2002], [2003], [2004])
) AS pvt
The Secure View

CREATE VIEW [Security].[vsSalesTotals]
AS
SELECT [SalesPersonID], [FullName], [Title], [SalesTerritory], [2002], [2003], [2004]
FROM Security.ufnGetSalesTotals(USER)
HR professionals can only see data for employees in their assigned departments (Slide 1 of 4): The Problem and Strategy

- **What we are trying to solve:**
  - Right now, all HR employees have access to all employee data.
  - We want to limit them and assigned them to specific departments

- **What we are going to do:**
  - We create a function that links user ID to the Territory
  - We create a view to prevent the user from inserting a different user ID than his
HR professionals can only see data for employees in their assigned departments (Slide 2 of 4): The Data Model.
HR professionals can only see data for employees in their assigned departments

(Slide 3 of 4) The Function

-- based on HumanResources.vEmployee

CREATE FUNCTION [Security].[ufnGetEmployeeData]
(
    @UserId VARCHAR(20)
)
RETURNS TABLE
AS
RETURN

SELECT
e.[EmployeeID],
c.[Title],
c.[FirstName],
c.[MiddleName],
c.[LastName],
c.[Suffix],
e.[Title] AS [JobTitle],
dpt.Name AS [DepartmentName],
shr.UserID,
c.[Phone],
c.[EmailAddress],
c.[EmailPromotion],
a.[AddressLine1],
a.[AddressLine2],
a.[City]
FROM [HumanResources].[Employee] e
INNER JOIN [Person].[Contact] c
    ON c.[ContactID] = e.[ContactID]
INNER JOIN [HumanResources].[EmployeeAddress] ea
    ON e.[EmployeeID] = ea.[EmployeeID]
INNER JOIN [Person].[Address] a
    ON ea.[AddressID] = a.[AddressID]
INNER JOIN [Person].[StateProvince] sp
    ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
    ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
INNER JOIN HumanResources.EmployeeDepartmentHistory edh
    ON edh.EmployeeID = e.EmployeeID
    AND edh.EndDate is null
INNER JOIN Security.HRAccess shr
    ON shr.DepartmentID = edh.DepartmentID
    AND shr.UserID = @UserId
INNER JOIN HumanResources.Department dpt
    ON dpt.DepartmentID = edh.DepartmentID
CREATE VIEW [Security].[vsEmployee]
AS
SELECT  
[EmployeeID] , [Title]
,[FirstName] , [MiddleName]
,[LastName] , [Suffix]
,[JobTitle] , DepartmentID
,[DepartmentName] , UserID
,[Phone] , [EmailAddress]
,[EmailPromotion] , [AddressLine1]
,[AddressLine2] , [City]
,[StateProvinceName] , [PostalCode]
,[CountryRegionName] , [AdditionalContactInfo]
FROM Security.ufnGetEmployeeData(USERS)
Summary

- RLS allowed us to add security controls and implement business rules on existing databases
  - The overall structure of the database stays unchanged
  - Cost of developments are low
  - Functions can be reused for future developments

- Other possible enhancements
  - Add a audit functionality: create a log of who tried to access which data and at what time (Sarbox, HIPAA and regulatory requirements)
  - Link to an LDAP like Active Directory for permanent business or security requirements