Lab Answer Key: Module 9: Managing SQL Server Security

Lab: Managing SQL Server Security

Exercise 1: Managing Server-Level Security

Task 1: Prepare the Lab Environment

1. Ensure that the 20462C-MIA-DC and 20462C-MIA-SQL virtual machines are both running, and then log on to 20462C-MIA-SQL as **ADVENTUREWORKS\Student** with the password **Pa$$w0rd**.
2. In the D:\Labfiles\Lab09\Starter folder, right-click **Setup.cmd** and then click **Run as administrator**.
3. Click **Yes** when prompted to confirm that you want to run the command file, and wait for the script to finish.

Task 2: Verify the Authentication Mode

1. Start SQL Server Management Studio, and connect to the **MIA-SQL** database engine using Windows authentication.
2. In Object Explorer, right-click the **MIA-SQL** instance and click **Properties**.
3. In the **Server Properties – MIA-SQL** dialog box, on the **Security** page, verify that **SQL Server and Windows Authentication mode** is selected. Then click **Cancel**

Task 3: Create Logins

1. In SQL Server Management Studio, expand the **MIA-SQL** instance, expand **Security**, and expand **Logins** to view the existing logins in the instance.

2. Open the **CreateLogins.sql** script in the D:\Labfiles\Lab09\Solution folder.
3. Review the script, noting that it creates the following logins.
    - [ADVENTUREWORKS\Database_Managers]
    - [ADVENTUREWORKS\WebApplicationSvc]
    - [ADVENTUREWORKS\InternetSales_Users]
    - [ADVENTUREWORKS\InternetSales_Managers]
    - Marketing_Application
4. Click **Execute**. Then, when the script has completed successfully, in Object Explorer, right-click **Logins** and click **Refresh** to verify that the logins have been created.

Task 4: Manage Server-Level Roles

1. In SQL Server Management Studio, under **Security**, expand **Server Roles** to view the existing roles in the instance.
2. Open the **ServerRoles.sql** script in the D:\Labfiles\Lab09\Solution folder.
3. Review the script, noting that it performs the following actions.
    - Creates a new server-level role named **application_admin**,
    - Adds the **[ADVENTUREWORKS\Database_Managers]** login to the **application_admin** role.
    - Grants ALTER ANY LOGIN and VIEW ANY DATABASE permissions to the **application_admin** role.
4. Click **Execute**. Then, when the script has completed successfully, in Object Explorer, right-click **Server Roles** and click **Refresh** to verify that the new role has been created.

Result: After this exercise, the authentication mode for the MIA-SQL SQL Server instance should support the scenario requirements, you should have created the required logins and server-level roles, and you should have granted the required server-level permissions.

Exercise 2: Managing Database-Level Security

Task 1: Create Database Users

1. In Object Explorer, expand **Databases**, expand the **InternetSales** database, and expand its **Security** folder. Then expand the **Users** folder and view the users currently defined in the database.
2. Open the **CreateUsers.sql** script in the D:\Labfiles\Lab09\Solution folder.
3. Review the script, noting that it creates the following users.

    - Marketing_Application
    - WebApplicationSvc
    - InternetSales_Users
    - InternetSales_Managers
    - Database_Managers

4. Click **Execute**. Then, when the script has completed successfully, in Object Explorer, right-click **Users** and click **Refresh** to verify that the new users have been created.

Task 2: Manage Database Roles

1. In Object Explorer, expand the **Roles** folder, and then expand the **Database Roles** folder and view the database roles in the database.
2. Open the **DatabaseRoles.sql** script in the D:\Labfiles\Lab09\Solution folder.
3. Review the script, noting that it performs the following actions.

    - Creates database roles named **sales_reader**, **sales_writer**, **customers_reader**, **products_reader**, and **web_application**,

- Adds the **Database_Managers** user to the **db_securityadmin** fixed database-level role.
- Adds the **InternetSales_Users** and **InternetSales_Managers** users to the **sales_reader** role.
- Adds the **InternetSales_Managers** user to the **sales_writer** role.
- Adds the **InternetSales_Users**, **InternetSales_Managers**, and **Marketing_Application** users to the **customers_reader** role.
- Adds the **InternetSales_Managers** and **Marketing_Application** users to the **products_reader** role.
- Adds the **WebApplicationSvc** user to the **web_application** role.
- Creates an application role named **sales_admin**.

4. Click **Execute**. Then, when the script has completed successfully, in Object Explorer, right-click **Database Roles** and click **Refresh** and expand **Application Roles** to verify that the new roles have been created.

Task 3: Assign Permissions

1. Open the **DatabasePermissions.sql** script in the D:\Labfiles\Lab09\Solution folder.
2. Review the script, noting that it grants the following permissions.

- SELECT on the **Sales** schema to **sales_reader**.
- INSERT, UPDATE, and EXECUTE on the **Sales** schema to **sales_writer.**
- SELECT, INSERT, UPDATE, DELETE, and EXECUTE on the **Sales** schema to **sales_admin**.
- SELECT on the **Customers** schema to **customers_reader**.
- SELECT on the **Products** schema to **products_reader.**
- EXECUTE on the **Products** schema to **InternetSales_Managers**.
- INSERT on **Sales.SalesOrderHeader** to **web_application**.
- INSERT on **Sales.SalesOrderDetail** to **web_application**.
- SELECT on **Products.vProductCatalog** to **web_application**.

3. Click **Execute**.

> **Result**: After this exercise, you should have created the required database users and database-level roles, and assigned appropriate permissions.

Exercise 3: Testing Database Access

Task 1: Test IT Support Permissions

1. Minimize SQL Server Management Studio and open a command prompt.
2. In the command prompt window, enter the following command (which opens the sqlcmd utility as ADVENTUREWORKS\AnthonyFrizzell):

```
runas /user:adventureworks\anthonyfrizzell /noprofile sqlcmd
```

3. When you are prompted for a password, enter **Pa$$w0rd**.
4. In the SQLCMD window, enter the following commands to verify your identity:

```
SELECT suser_name();
GO
```

5. Note that SQL Server identifies Windows group logins using their individual user account, even though there is no individual login for that user. **ADVENTUREWORKS\AnthonyFrizzell** is a member of the **ADVENTUREWORKS\IT_Support** global group, which is in turn a member of the **ADVENTUREWORKS\Database_Managers** domain local group for which you created a login.
6. In the SQLCMD window, enter the following commands to alter the password of the **Marketing_Application** login.

```
ALTER LOGIN Marketing_Application WITH PASSWORD =
'NewPa$$w0rd';
GO
```

7. In the SQLCMD window, enter the following commands to disable the **ADVENTUREWORKS\WebAplicationSvc** login.

```
ALTER LOGIN [ADVENTUREWORKS\WebApplicationSvc] DISABLE;
GO
```

8. Close the SQLCMD window and maximize SQL Server Management Studio.
9. In Object Explorer, right-click the **Logins** folder and click **Refresh**. Then right-click the **ADVENTUREWORKS\WebApplicationSvc** login and click **Properties**.
10. In the **Login Properties -ADVENTUREWORKS\WebApplicationSvc** dialog box, on the **Status** page, select **Enabled** and click **OK** to re-enable the login.

Task 2: Test Marketing Application Permissions

1. In SQL Server Management Studio, click **New Query**.
2. In the new query window, enter the following Transact-SQL code to impersonate the **Marketing_Application** login.

```
EXECUTE AS LOGIN  = 'Marketing_Application'
GO
SELECT suser_name();
GO
```

3. Click **Execute** and verify that the connection is executing in the context of the **Marketing_Application** login.

4. Enter the following Transact-SQL code under the previous code:

```
USE InternetSales;
SELECT * FROM sys.fn_my_permissions(‘Customers.Customer’,
'object');
GO
```

5. Select the code you just entered and click **Execute** to view the effective permissions for **Marketing_Application** on the **Customers.Customer** table.

6. Enter the following Transact-SQL code under the previous code:

```
SELECT * FROM Customers.Customer;
GO
```

7. Select the code you just entered and click **Execute** to verify that the user can query the **Customers.Customer** table.

8. Enter the following Transact-SQL code under the previous code:

```
UPDATE Customers.Customer
SET EmailAddress = NULL
WHERE CustomerID = 1;
GO
```

9. Select the code you just entered and click **Execute** to verify that the user does not have UPDATE permission on the **Customers.Customer** table.

10. Enter the following Transact-SQL code under the previous code:

```
SELECT * FROM Products.Product;
GO
```

11. Select the code you just entered and click **Execute** to verify that the user can query the **Product.Products** table.

12. Enter the following Transact-SQL code under the previous code:

```
SELECT * FROM Sales.SalesOrderHeader;
GO
```

13. Select the code you just entered and click **Execute** to verify that the user does not have SELECT permission on the **Sales.SalesOrderHeader** table.

14. Close SQL Server management Studio without saving any files.

Task 3: Test Web Application Permissions

1. In the command prompt window, enter the following command to run sqlcmd as **ADVENTUREWORKS\WebApplicationSvc**:

```
runas /user:adventureworks\webapplicationsvc /noprofile sqlcmd
```

2. When you are prompted for a password, enter **Pa$$w0rd**.

3. In the SQLCMD window, enter the following commands to query the **Products.vProductCatalog** view:

```
SELECT ProductName, ListPrice FROM Products.vProductCatalog;
GO
```

4. Verify that the user can query the **Products.vProductCatalog** view.

5. In the SQLCMD window, enter the following commands to query the **Products.Product** table:

```
SELECT * FROM Products.Product;
```

```
GO
```

6. Verify that the user does not have SELECT permission on the **Products.Product** table.

7. Close the SQLCMD window.

Task 4: Test Sales Employee Permissions

1. In the command prompt window, enter the following command to run sqlcmd as **ADVENTUREWORKS\DanDrayton**. This user is a member of the **ADVENTUREWORKS\Sales_NorthAmerica** global group, which is in turn a member of the **ADVENTUREWORKS\InternetSales_Users** domain local group.

```
runas /user:adventureworks\dandrayton /noprofile sqlcmd
```

2. When you are prompted for a password, enter **Pa$$w0rd**.

3. In the SQLCMD window, enter the following commands to query the **Sales.SalesOrderHeader** table:

```
SELECT SalesOrderNumber, TotalDue FROM Sales.SalesOrderHeader;
GO
```

4. Verify that the user can query the **Sales.SalesOrderHeader** table.

5. In the SQLCMD window, enter the following commands to update the **Sales.SalesOrderHeader** table:

```
UPDATE Sales.SalesOrderHeader SET ShipDate = getdate() WHERE
SalesOrderID = 45024;
GO
```

6. Verify that the user does not have UPDATE permission on the **Sales.SalesOrderHeader** table.

7. Close the SQLCMD window.

Task 5: Test Sales Manager Permissions

1. In the command prompt window, enter the following command to run sqlcmd as **ADVENTUREWORKS\DeannaBall**. This user is a member of the **ADVENTUREWORKS\Sales_Managers** global group, which is in turn a member of the **ADVENTUREWORKS\InternetSales_Managers** domain local group.

```
runas /user:adventureworks\deannaball /noprofile sqlcmd
```

2. When you are prompted for a password, enter **Pa$$w0rd**.

3. In the SQLCMD window, enter the following commands to query the **Sales.SalesOrderHeader** table:

```
SELECT SalesOrderNumber, TotalDue FROM Sales.SalesOrderHeader;
GO
```

4. Verify that the user can query the **Sales.SalesOrderHeader** table.

5. In the SQLCMD window, enter the following commands to update the **Sales.SalesOrderHeader** table:

```
UPDATE Sales.SalesOrderHeader SET ShipDate = getdate() WHERE
SalesOrderID = 45024;
GO
```

6. Verify that the user can update the **Sales.SalesOrderHeader** table.

7. In the SQLCMD window, enter the following commands to update the **Products.Product** table:

```
UPDATE Products.Product SET ListPrice = 1999.00 WHERE
ProductID = 1;
GO
```

8. Verify that the user cannot update the **Products.Product** table.

9. In the SQLCMD window, enter the following commands to call the **Products.ChangeProductPrice** stored procedure:

```
EXEC Products.ChangeProductPrice 1, 1999.00;
GO
```

10. Verify that the one row is affected (because the user has EXECUTE permission on the **Products.ChangeProductPrice** stored procedure).

11. In the SQLCMD window, enter the following commands to delete a row from the **Sales.SalesOrderDetail** table:

```
DELETE Sales.SalesOrderDetail WHERE SalesOrderDetailID =
37747;
GO
```

12. Verify that the user cannot delete rows from the **Sales.SalesOrderDetail** table.

13. In the SQLCMD window, enter the following commands to activate the **sales_admin** application role and delete a row from the **Sales.SalesOrderDetail** table:

```
EXEC sp_setapprole 'sales_admin', 'Pa$$w0rd'
GO
DELETE Sales.SalesOrderDetail WHERE SalesOrderDetailID =
```

```
37747;
GO
```

14. Verify that the one row is affected. This is possible because the **sales_admin** application role has DELETE permission on the **Sales** schema.

15. Close the SQLCMD window.

Result: After this exercise, you should have verified effective permissions in the MIA-SQL instance and the InternetSales database.