Lab 9 – Database Security

Deliverable

Word or PDF File containing your work

Set up

In this lab, you will apply the concepts learned in this week's lectures and readings. You'll need access to the SQL Server database created for you on the iSchool SQL Server to perform these tasks. You will also need a partner for this lab. You will be granting them the ability to add data to and read data from a table using stored procedures and views.

Steps

Create a blank document to record your answers to the questions called out below. Ensure your name is at the top of the document!

NOTE: This lab was written to be run on the iSchool server. If you are running your own server, you will need to create a SQL Server login on your server. This is outside the scope of this lab, but you can find the specifics here: https://msdn.microsoft.com/en-us/library/aa337562(v=sql.105).aspx

If you do create your own login, for the purposes of this lab it's best to name it guestuser and set its password to SU2orange! to be consistent with the instructions below.

Step 1 – Create Some Data.

First, let's set up some data and an internal model (views and stored procedures). Logged into SQL Server with your own credentials, code and execute the following SQL commands:

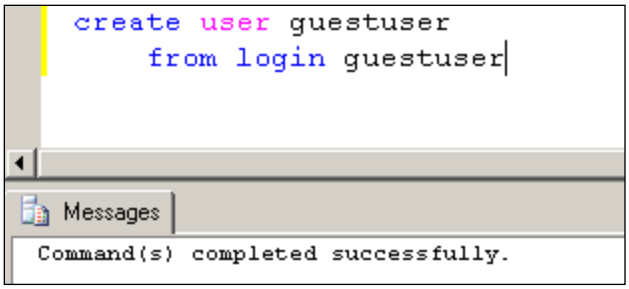
```
1  CREATE TABLE Lab9 (
2      Lab9ID  int identity primary key
3      , Lab9Text char(30) not null
4      , EnteredBy char(30) not null
5  )
6  GO
7  CREATE VIEW PartnerLab9 AS
8      SELECT * FROM Lab9 WHERE EnteredBy = 'Partner'
9
10 GO
11 CREATE PROCEDURE AddPartnerLab9 (@Lab9Text char(30)) AS
12 BEGIN
13     INSERT INTO Lab9 (Lab9Text, EnteredBy) VALUES (@Lab9Text, 'Partner')
14 END
15 GO
16 CREATE VIEW MyLab9 AS
17     SELECT * FROM Lab9 WHERE EnteredBy = 'Me'
18 GO
19 CREATE PROCEDURE AddMyLab9 (@Lab9Text char(30)) AS
20 BEGIN
21     INSERT INTO Lab9 (Lab9Text, EnteredBy) VALUES (@Lab9Text, 'Me')
22 END
23 GO
24 EXEC AddMyLab9 'First Value'
25 EXEC AddMyLab9 'Second Value'
26 EXEC AddMyLab9 'Third Value'
27 EXEC AddMyLab9 'Fourth Value'
28 SELECT * FROM MyLab9
29 SELECT * FROM Lab9
```

Take a screenshot of your results (just the results from lines 18 and 19 above) and paste them into your answers doc.

Step 2 – Connect as a Guest and do some things.

Up to this point, we have always logged in as ourselves. We now need to connect to the server as a different user. As noted above, these instructions are written for use on the iSchool server, so adjust accordingly if you're running your own.

Code and execute the following statement against your database:

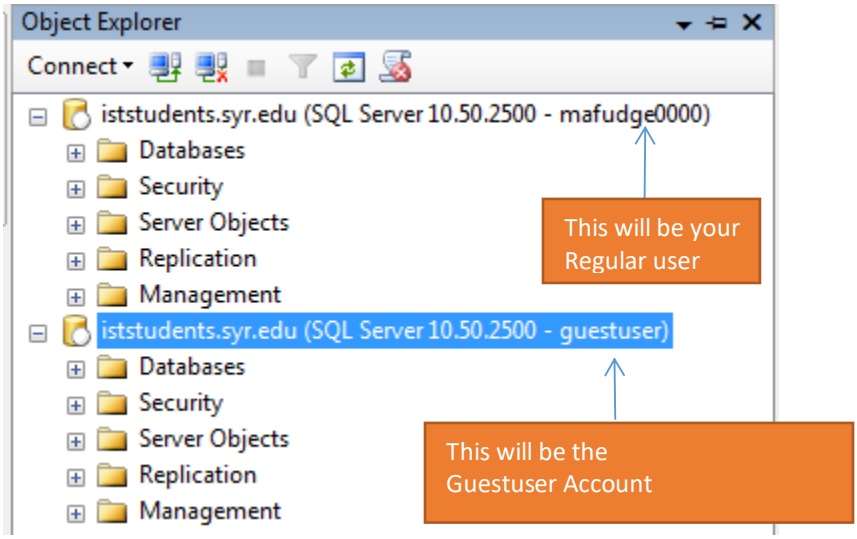


This statement creates a user called guestuser in your database and uses the database server login, guestuser, as the login. Now, when someone logs into the server using that login, they can access your database and execute statements against your database based on permissions you've given them.

To try this, let's login to the server as the guestuser. From the menu, choose File ➔ Connect Object explorer. The familiar SQL Server login dialog will appear, but **this time you will not login with your account.**

Login as guestuser the password for guestuser is SU2orange!

You will now see TWO connections in object explorer. Your original connection, and the guestuser connection, like this:



Next, we need to open a separate tab to work in as guestuser. We can do this because each tab connects using credentials borrowed from the connection defined in the Object Exploer. Since we have 2 connections now, be sure to select the guestuser connection (see above) and click the New Query button in the toolbar.

Note that the user is identified in the tab at the top of the query window. It will probably be helpful to add a comment to the top of the new query indicating it is the guestuser tab (so we don't get them confused). From here forward, we refer to this new query window as the guestuser tab and the other one as the regular tab. If you close the query for your regular account, simply click the connection in the Object Explorer and click the New Query button in the toolbar.

In the **guestuser tab**, code and execute the following statement.

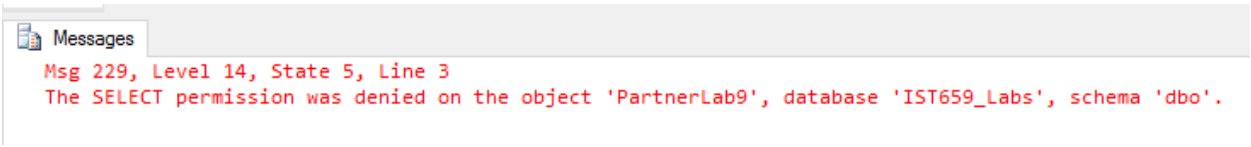
```
1  -- This is my guestuser tab!!
2
3  -- Replace IST659_Labs with your database name!
4  USE IST659_Labs
5
```

This statement changes the currently active database for this query window. (Use this code or change the database from the dropdown any time you have to create a new query window.)

In the **guestuser tab**, code and execute the following statement:

```
SELECT * FROM PartnerLab9
```

Note the following error message when we execute that otherwise benign SELECT statement:



This is because we haven't yet give the guestuser access to this view. To allow guestuser to select from that view, return to your **regular tab** and execute the following statement:

```
GRANT SELECT ON PartnerLab9 TO guestuser
```

Now return to your guestuser tab and re-run the SELECT statement from before:

```
SELECT * FROM PartnerLab9
```

Did it work? If so, paste a screenshot of your results to your answer worksheet. (There should be no rows in your data yet, but you should get the column headings and no rows or errors).

Now, we want this user to be able to add rows to the Lab9 table, but we want to control what gets added and how. Specifically, when this user adds a record, we want the EnteredBy to be Partner. We can either entrust the programmer (guestuser in this scenario) to always put it in the INSERT statement, or we can just give them access to a Stored Procedure that does the whole thing for them. Let's do the second.

In the beginning of the lab, we coded a stored procedure called AddPartnerLab9 just for this scenario. Let's return to the **regular tab** and grant the guest user access to execute this procedure:

```
GRANT EXECUTE ON AddPartnerLab9 TO guestuser
```

Now, return to your guestuser tab and execute the following SQL statements:

```
EXEC AddPartnerLab9 'First Partner Record'
EXEC AddPartnerLab9 'Second Partner Record'
EXEC AddPartnerLab9 'Third Partner Record'
EXEC AddPartnerLab9 'Fourth Partner Record'

SELECT * FROM PartnerLab9
```

Copy and paste the results of the last select statement to your answer doc.

In your guestuser tab, run the following statement:

```
DELETE Lab9 WHERE EnteredBy = 'Partner'
```

Did this work? Why or why not? Add your answer to the answers doc.

Lastly, in your regular tab, execute the following SQL statements:

```
REVOKE SELECT ON PartnerLab9 TO guestuser

REVOKE EXECUTE ON AddPartnerLab9 TO guestuser
```

In your answer doc, answer the following questions:

1. What does the CREATE USER statement do? Hint: It's not creating a user….
2. Which types of objects are securable with GRANT and DENY statements? (Research required.)
3. Describe a scenario by which you would not want a user reading the table directly but rather using a view instead.
4. Describe a scenario by which you would not want a user updating data in the table directly but rather using a stored procedure instead?
5. Write SQL to deny the guest user permissions to delete from a table called Product
6. Write SQL to allow the guest user the ability to run one of your SQL Views. Place your sql here and the message returned when executed.