Assignment #8

Instructor: Richard Frank (rfrank@sfu.ca)
TA: Ankit Gupta (aga53@sfu.ca)

Total Marks: 50 (5% of the Individual Assignments)
Due Date: Nov 11, 14:30 (via the Online Submission Server)

Question 1) You are in charge of testing the Killer Application which Asok (the intern) developed for his Pointy-Haired Boss. You find out it is vulnerable to SQL Injection. Please describe 4 methods you could use to secure the Killer Application against SQL Injection. State what the approach is, how it works, and how it will secure against this attack, or how it will minimize the data-loss should this attack be successful. **[10 marks each]**

- Encrypting data: The value of a salted hash is such that a dictionary attack is not going to work as each dictionary would have to be rebuilt appending the various salt values and recomputing the hash values for each item. While it is still possible to determine the password by brute force, the use of the salt (even though it is known) greatly slows down the process. The second advantage of the salt is that it masks any situations where two independent users happen to use the same password, as the salted hash value for each user would be different if given different salt values.
- Least Privilege account: Use an account that has no privileges on the SQL Server other than that which is absolutely necessary. For example, on a SELECT statement use an account that cannot INSERT/DROP/DELETE.
- Clean and validate the user input
    - Black-list: detail what cannot be entered, or must be escaped.
    - White-list: detail the form of the input that is accepted, everything else is rejected. For example, for phone numbers, accept only numbers.
- Parameterised Queries: This is where the SQL Command uses a parameter instead of injecting the values directly into the command. For example:

```
string commandText = "SELECT * FROM Customers "+
    "WHERE Country=@CountryName";
SqlCommand cmd = new SqlCommand(commandText, conn);
cmd.Parameters.Add("@CountryName",countryName);
```

- Stored Procedures: add an extra layer of abstraction in to the design of a software system. This means that, so long as the interface on the stored procedure stays the same, the underlying table structure can change with no noticeable consequence to the application that is using the database. This layer of abstraction also helps put up an extra barrier to potential attackers. If access to the data in SQL Server is only ever permitted via stored procedures, then permission does not need to be explicitly set on any of the tables. Therefore, none of the tables should ever need to be exposed directly to outside applications. For an outside application to read or modify the database, it must go through stored procedures. Even though some stored procedures, if used incorrectly, could potentially damage the database, anything that can reduce the attack surface is beneficial.
- Disable error messages

(solution based off http://www.codeproject.com/KB/database/SqlInjectionAttacks.aspx)

Question 2) [5 marks each]

a) Poor Asok (username: Asok) messed up the Killer Application, thus his access to the AdventureWorks.Products table has been revoked. Please write the SQL command to do this.

```
REVOKE SELECT, DELETE, INSERT
ON AdventureWorks.Products
FROM Asok
```

b) To replace Asok, the megalomaniac Dogbert (username: Dogbert) has been given the privilege to manage the Products table as he sees fit (i.e. so that he can give others the privilege to allow others to access the table). Please write the SQL command to do this.

```
GRANT ALL
ON Products
TO Dogbert
WITH GRANT OPTION
```