Web Application Security

Rajendra Kachhwaha
rajendra1983@gmail.com

October 28, 2015

Outline

Attacking Data Source:

1. Get the database details of the victim
2. Preventing SQL Injection
    1. Perform Input Validation.
    2. Enforce least privilege.
    3. Use Parametrized Queries.
    4. Use Stored Procedures.

Process:

1. At this point, you know the www.example.cxx website is vulnerable to Sql injection attack, as per discussed in last lecture.
2. Now we will try to get the database details including database server version, database tables, columns details in the tables.
3. I think, finding this much information from the database server will be enough.

Process:

1. Get the version of the database server through modified query.

Process:

1. At this point, you know the www.example.cxx website is vulnerable to Sql injection attack, as per discussed in last lecture.
2. Now we will try to get the database details including database server version, database tables, columns details in the tables.
3. I think, finding this much information from the database server will be enough.

Process:

1. Get the version of the database server through modified query. *select @@VERSION*

cont.

Process:

2. Get the existing table in the database server through modified query.

Process:

2. Get the existing table in the database server through modified query.
   select table_name from INFORMATION_SCHEMA.TABLES
3. Get the existing columns from tables in the database server through modified query.

Process:

2. Get the existing table in the database server through modified query.
   select table_name from INFORMATION_SCHEMA.TABLES
3. Get the existing columns from tables in the database server through modified query.
   *select * from information_schema.columns*
4. Get the details from tables in the database server through modified query.

Process:

2. Get the existing table in the database server through modified query.
   select table_name from INFORMATION_SCHEMA.TABLES
3. Get the existing columns from tables in the database server through modified query.
   *select * from information_schema.columns*
4. Get the details from tables in the database server through modified query.
   *select * from Table_Name*

Techniques:

Following is the common method for accessing databases (UNSAFE):

```
String query = "SELECT account_balance FROM user_data WHERE user_name = "
  + request.getParameter("customerName");

try {
       Statement statement = connection.createStatement( … );
       ResultSet results = statement.executeQuery( query );
}
```

Techniques for Prevention:

1. Perform Input Validation.
2. Enforce least privilege.
3. Use Parametrized Queries.
4. Use Stored procedures.(adv-disadv)

Techniques for Prevention:

Perform Input Validation:

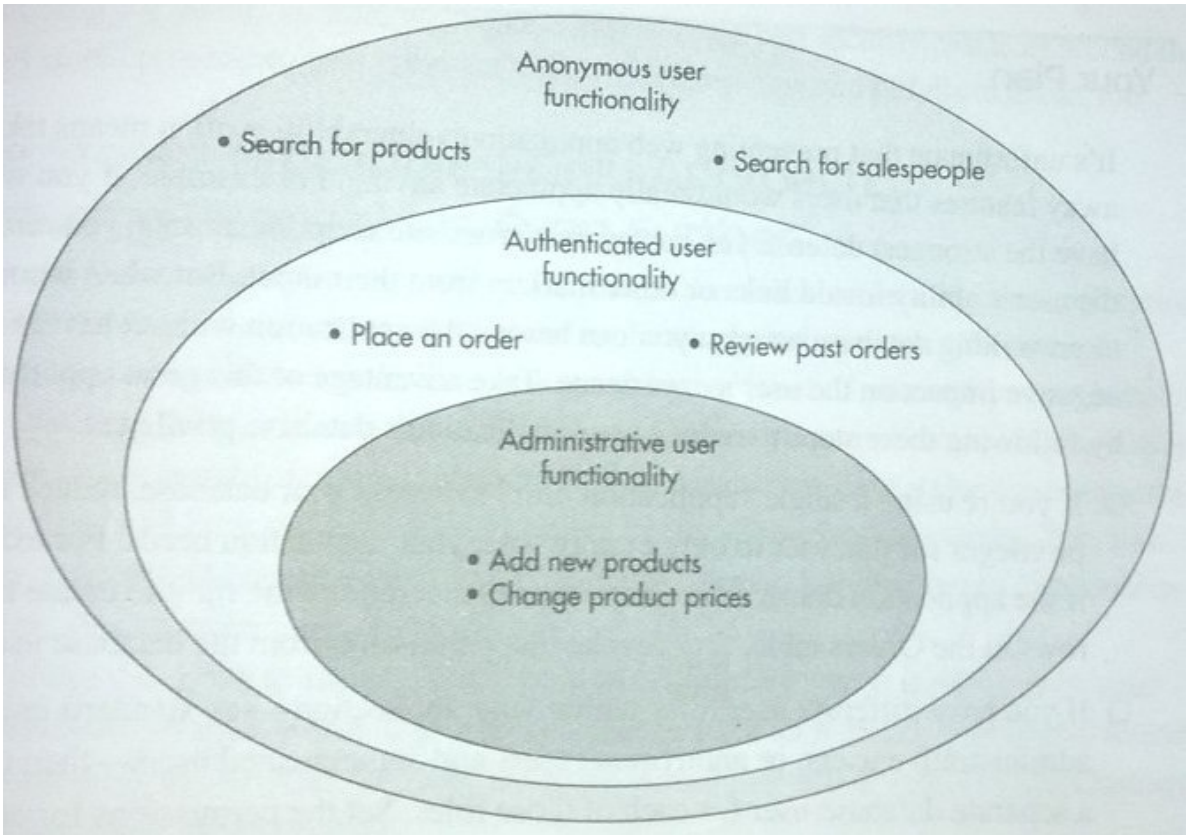
1. Input validation can be used to detect unauthorized input before it is passed to the SQL query. It includes White list validation and Black list validation.

Enforce least privilege:

1. We should minimize the privileges assigned to every database account in your environment.
2. Start from the ground up to determine what access rights your application accounts require, rather than trying to figure out what access rights you need to take away.
3. Make sure that accounts that only need read access are only granted read access to the tables they need access to.
4. Minimizing the privileges granted to your application will reduce such unauthorized access attempts.

Techniques for Prevention:

Enforce least privilege:



Techniques for Prevention:

Use Parametrized Queries:

1. Parametrized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later.
2. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.
3. Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.
4. In the safe example below, if an attacker were to enter the userID of tom' or '1'='1, the parametrized query would not be vulnerable and would instead look for a username which literally matched the entire string tom' or '1'='1.

cont.

Techniques for Prevention:

Use Parametrized Queries:

```
String query =
        "SELECT account_balance FROM user_data WHERE user_name = ?";
try {
       OleDbCommand command = new OleDbCommand(query, connection);
       command.Parameters.Add(new OleDbParameter("customerName", CustomerName Name.Text));
       OleDbDataReader reader = command.ExecuteReader();
       // …
} catch (OleDbException se) {
       // error handling
}
```

Techniques for Prevention:

Use stored procedures:

1. A stored procedure (also termed proc, StoreProc, sp, or SP) is a subroutine available to applications that access a relational database management system.
2. Typical uses for stored procedures include data validation (integrated into the database) or access control mechanisms.
3. Stored procedures can consolidate and centralize logic that was originally implemented in applications.
4. Extensive or complex processing that requires execution of several SQL statements is moved into stored procedures, and all applications call the procedures.
5. One can use nested stored procedures by executing one stored procedure from within another.

Techniques for Prevention:

Use stored procedures: The following code example uses a SqlCommand, .NETs implementation of the stored procedure interface, to execute the same database query. The "*sp_getAccountBalance*" stored procedure would have to be predefined in the database and implement the same functionality as the query defined in last slide.

```
Try
        Dim command As SqlCommand = new SqlCommand("sp_getAccountBalance", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add(new SqlParameter("@CustomerName", CustomerName.Text))
        Dim reader As SqlDataReader = command.ExecuteReader()
        ' …
Catch se As SqlException
        ' error handling
End Try
```

Techniques for Prevention:

Use stored procedures:Advantages:

1. Reduce network usage between clients and servers - stored procedures perform intermediate processing on the database server reducing unnecessary data transfer across the network.
2. Stored procedures are tunable to improve the performance. When same stored procedure executed again, it can use the previously cached execution plans.
3. Separate or abstract server side functions from the client side.
4. Access to other database objects in a secure way.
5. Can be tested independent of the application.
6. Having Stored Procedures in one location means that there's no confusion of having business rules spread over potentially disparate code files in the application.

Techniques for Prevention:

Use stored procedures:Disadvantages:

1. Writing and maintaining stored procedures requires more specialized skills.
2. Stored procedure language may differ from one database system to another.
3. Tightly coupled to the database system.
4. Sometimes it is hard to understand the logic written in dynamic SQL.
5. Any data errors in handling Stored Procedures are not generated until run time.