## Il controllo di *SQL Injection* nelle pagine ASP e ASP.NET

Approfondimenti

Con il termine **SQL Injection** si intende l'aggiunta di istruzioni SQL nell'input che l'utente fornisce nel browser tramite un form HTML. Viene utilizzato dagli utenti non autorizzati (*hacker*) per accedere ai database senza controlli e senza identificazione oppure per far eseguire comandi SQL diversi da quelli previsti dall'applicazione.

Poiché questo comporta rischi per la sicurezza dei database, è importante prevenire l'inserimento di codice SQL indesiderato (nel gergo informatico, *malicious SQL*, codice doloso).

Per comprendere come funziona l'*SQL Injection*, si consideri la pagina Web che richiede all'utente lo username e la password per accedere a un servizio Internet. Essa contiene due caselle di testo all'interno di un form HTML, denominate *utente* e *password*, e richiama l'esecuzione di una pagina ASP che legge dalla tabella *Utenti* di un database l'elenco degli utenti autorizzati, controllando che i dati forniti corrispondano all'identificativo dell'utente. Supponiamo che la tabella *Utenti* contenga, per ogni riga, due soli campi: *username* e *password*.

I seguenti frammenti di codice descrivono le principali operazioni svolte dalla pagina ASP.

- acquisizione dei dati dal form della pagina Web:

```
user = Request.Form("utente")
pwd = Request.Form("password")
```

- commando SQL:

```
  strSQL = "Select * From Utenti "
  strSQL = strSQL & "Where "
  strSQL = strSQL & "username ='" & user & "' and "
  strSQL = strSQL & "password ='" & pwd & "'"
```

- esecuzione del comando SQL e controllo dei dati:

```
Set rs = conn.Execute(strSQL)
If not(rs.Eof) Then
     Response.Write("utente identificato")
Else
     Response.Write("utente non identificato")
End If
```

Se l'utente fornisce come username *user1* e come password *passw1*, il comando SQL nella pagina ASP diventa:

```
Select * From Utenti
Where username ='user1'
And password ='passw1'
```

Se l'utente esiste nel database, il processo di identificazione fornisce esito positivo.

Supponiamo ora che l'utente inserisca per lo username, o per la password o per entrambi, la seguente sequenza di caratteri:

```
1' Or '1' = '1
```

Poiché l'apice è il delimitatore delle stringhe, il comando SQL diventa:

```
Select * From Utenti
Where username ='1' Or '1' = '1'
And password ='1' Or '1' = '1'
```

Le condizioni scritte dopo *Or* sono sicuramente vere ('1' = '1'), rendendo complessivamente vere le condizioni per lo username e la password: il controllo di identificazione viene superato in modo positivo. In questo modo un utente potrebbe accedere ai dati senza conoscere username e password.

Per impedire l'*SQL Injection* occorre stabilire permessi di accesso più restrittivi per gli utenti del database e inserire all'interno delle pagine ASP meccanismi di validazione dei dati forniti dall'utente, prima che essi vengano utilizzati per l'accesso alle tabelle del database.

Di seguito vengono illustrati alcuni metodi di validazione dei dati.

- **Sostituzione degli apici con i doppi apici**

L'apice viene sostituito con il doppio apice in modo che non possa essere interpretato come delimitatore di stringhe:

```
user = Request.Form("utente")
pwd = Request.Form("password")
user = Replace(user,"'","''")
pwd = Replace(pwd,"'","''")
```

La funzione predefinita **Replace** sostituisce all'interno della stringa, indicata come primo parametro, tutte le occorrenze del carattere indicato come secondo parametro con i caratteri del terzo parametro.

- **Validazione dei caratteri inseriti**

Con questa seconda tecnica ciascun carattere delle stringhe fornite dall'utente viene controllata per vedere se corrisponde a un carattere alfabetico (maiuscolo o minuscolo) oppure a una cifra numerica.

```
user = Request.Form("utente")
pwd = Request.Form("password")
caratteri =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
ok1 = True
For i = 1 To Len(user)
car = Mid(user, i, 1 )
If (InStr(caratteri,car) = 0) Then
  ok1 = False
  Exit For
End If
Next
```

```
ok2 = True
For i = 1 To Len(pwd)
car = Mid(pwd, i, 1 )
If (InStr(caratteri,car) = 0) Then
  ok1 = False
  Exit For
End If
Next

If (ok1 = True And ok2 = True) Then

     accesso alla tabella del database

Else
  Response.Write("Inserimento dati non corretto")
End If
```

Il codice precedente utilizza alcune funzioni stringa predefinite del linguaggio VBScript:
- **Len** restituisce la lunghezza di una stringa;
- **Mid** estrae una sottostringa da una stringa;
- **InStr** trova la posizione della prima occorrenza di una stringa all'interno di un'altra stringa e restituisce il valore 0 in caso di esito negativo per la ricerca.

• **Validazione con controllo dei comandi SQL**

La terza modalità controlla che le stringhe fornite dall'utente non contengano comandi SQL oppure il carattere apice, oppure i caratteri – – che in molte versioni del linguaggio SQL indicano l'inizio di frasi di commento; l'inserimento di un commento, infatti, fa perdere il significato di codice alla rimanente parte del comando SQL previsto dal programmatore nell'applicazione.
Le parole da controllare sono contenute in un *array* di 8 elementi. Anche questo codice utilizza la funzione **InStr** per la ricerca di una sottostringa all'interno di una stringa, con l'aggiunta del parametro *vbtextcompare* per indicare il confronto testuale. Il primo parametro 1 indica che la ricerca deve iniziare dal primo carattere.

```
user = Request.Form("utente")
pwd = Request.Form("password")

parole = array("select","insert","update","delete","drop","alter", "--", "'")

ok1 = True
For i = 0 To 7
If (InStr(1,user,parole(i),vbtextcompare) <> 0) Then
  ok1 = False
  Exit For
End If
Next

ok2 = True
For i = 0 To 7
If (InStr(1,pwd,parole(i),vbtextcompare) <> 0) Then
  ok2 = False
  Exit For
End If
Next
```

```
If (ok1 = True And ok2 = True) Then

    accesso alla tabella del database

Else
  Response.Write("Inserimento dati non corretto")
End If
```

- **Uso dei parametri nei comandi SQL delle pagine ASP.NET**

Le pagine ASP.NET consentono di utilizzare i **parametri** nei comandi SQL costruiti dinamicamente nell'applicazione. L'uso dei parametri è un buon metodo per prevenire l'*SQL Injection.*

La seguente pagina ASP.NET illustra le modalità per l'identificazione dell'utente tramite username e password, vista negli esempi precedenti.

Essa utilizza l'oggetto **Parameter** di ADO.NET (si chiama **OleDbParameter** per i database di *Access* e **SqlParameter** per i database di *SQLServer*) che consente di memorizzare le informazioni del parametro, da aggiungere ad un oggetto *Command*.

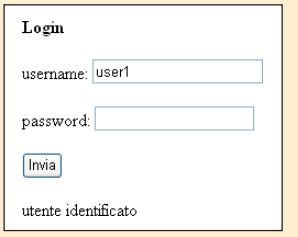
I parametri sono indicati all'interno della stringa del comando SQL con un nome preceduto dal carattere @.

L'aggiunta di un parametro (*Add*) all'oggetto *Command*, con assegnazione del valore, è effettuato con un'istruzione come la seguente:

```
dbcomm.Parameters.Add(new OleDbParameter("@param1", utente.Text))
```

Essa assegna al parametro, denominato *@param1*, il valore acquisito tramite la casella di testo *utente* del form HTML.

Si noti anche l'uso della proprietà **HasRows** dell'oggetto *DataReader*: essa assume il valore *True* se il *DataReader*, ottenuto dall'esecuzione del comando SQL possiede righe (*has rows*), cioè non è vuoto.




PAGINA ASP.NET (*login.aspx*)

```
<%@ Page Language="VB" %>
<%@ Import Namespace="System.Data.OleDb" %>
<script runat="server">

    Sub Page_Load()
    lbl1.Text="Login "
    End Sub

    Sub Submit(sender As Object, e As EventArgs)
    Dim dbconn as OleDbConnection
    Dim dbcomm as OleDbCommand
    Dim dbread as OleDbDataReader
    Dim strSQL As String

   dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data source="
          & server.mappath("db1.mdb"))
   dbconn.Open()
   strSQL = "Select * From Utenti "
```

```
    strSQL &= "Where "
    strSQL &= "username =@param1 and password=@param2"
    dbcomm=New OleDbCommand(strSQL,dbconn)
    dbcomm.Parameters.Add(new OleDbParameter("@param1", utente.Text))
    dbcomm.Parameters.Add(new OleDbParameter("@param2", password.Text))
    dbread=dbcomm.ExecuteReader()

    If dbread.HasRows Then
       lbl2.Text = "utente identificato"
    Else
       lbl2.Text = "utente non identificato"
    End If

    dbread.Close()
    dbconn.Close()
    End Sub

</script>
<html>
<head>
</head>
<body>
<form runat="server">
<p>
   <asp:Label id="lbl1" runat="server" font-bold="True"></asp:Label>
</p>
<p>
   username:
   <asp:TextBox id="utente" runat="server"></asp:TextBox>
</p>
<p>
   password:
   <asp:TextBox id="password" runat="server" TextMode="Password"></asp:TextBox>
</p>
<p>
  <asp:Button id="Button1" onclick="Submit" runat="server" Text="Invia"></asp:Button>
</p>
<p>
    
</p>
<p>
   <asp:Label id="lbl2" runat="server"></asp:Label>
</p>
</form>
</body>
</html>
```

Si osservi che la proprietà **TextMode** per la casella di testo della password è impostata al valore **Password**, per impedire la visualizzazione dei caratteri durante la digitazione dell'utente.