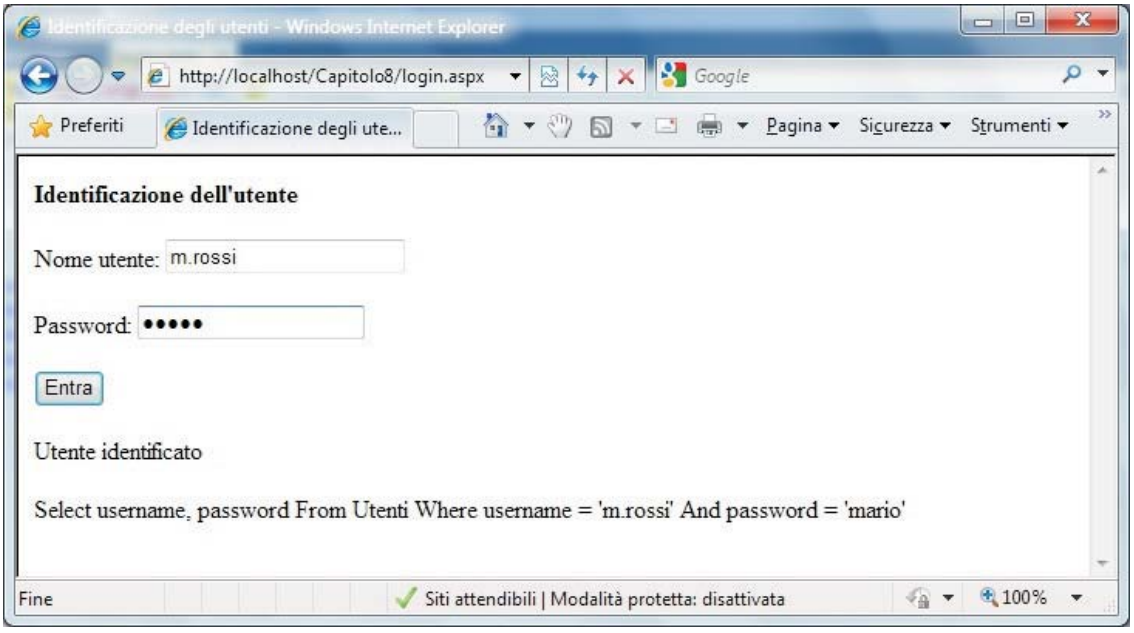
Il controllo di SQL Injection

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 (pirati informatici o *cracker*) 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 o **codice malevolo** (nel gergo informatico, **malicious SQL**).
Per comprendere come funziona l'*SQL Injection*, si consideri la pagina Web che richiede all'utente *username* e *password* per accedere a un servizio Internet.
Essa contiene due caselle di testo all'interno di un form, denominate *utente* e *password*: la pagina ASP.NET legge dalla tabella *Utenti* del database *dbProva.accdb* l'elenco degli utenti autorizzati, controllando che i dati forniti corrispondano ad uno tra gli identificativi registrati nella tabella. Supponiamo che la tabella *Utenti* contenga, per ogni riga, due soli campi: *username* e *password*.



Nella pagina è stata inserita anche una Label che consente, in fase di collaudo della pagina, di visualizzare la stringa del comando SQL.

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

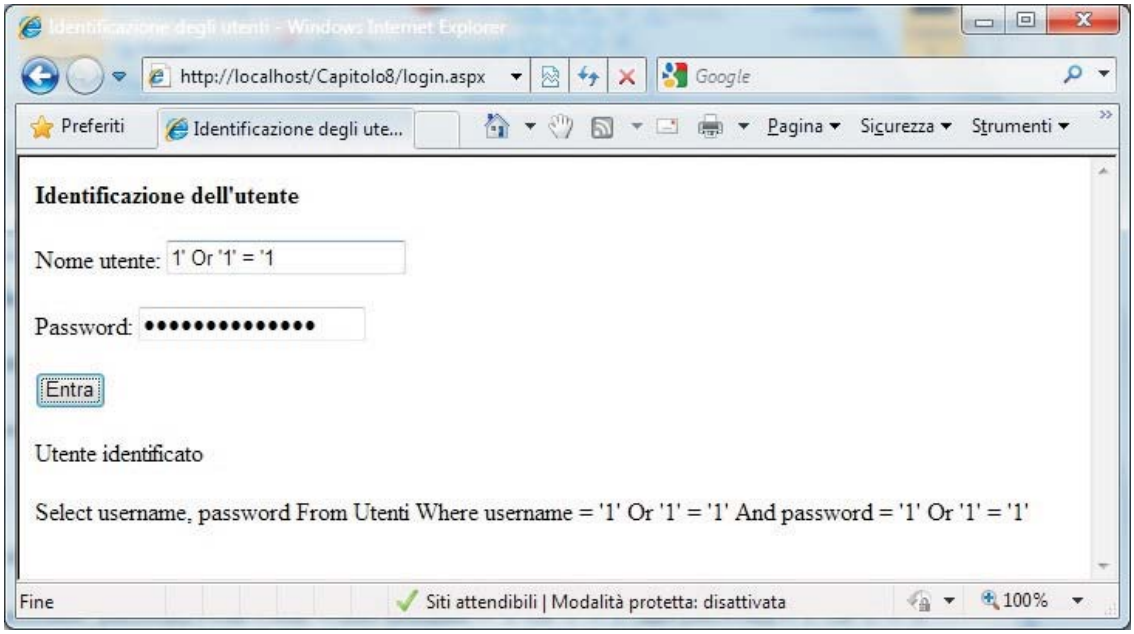
```
1' Or '1' = '1
```

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

```
Select username, password 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 Web 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**

Dopo aver acquisito i valori dalle caselle di testo, l'apice viene sostituito con il doppio apice in modo che non possa essere interpretato come delimitatore di stringhe:

```
user1 = utente.text
passw1 = password.Text
user1 = user1.Replace("", """)
passw1 = passw1.Replace("", """)
```

Il metodo **Replace** applicato a una stringa sostituisce all'interno della stringa tutte le occorrenze del carattere indicato come primo parametro con i caratteri del secondo parametro.

- **Validazione dei caratteri inseriti**

Con questa seconda tecnica ciascun carattere delle stringhe fornite dall'utente viene controllato per verificare che corrisponda a un carattere alfabetico (maiuscolo o minuscolo) oppure a una cifra numerica.

```
Dim user1, passw1 As String
Dim caratteri, car As String
Dim ok1, ok2 As Boolean
Dim i As Integer
user1 = utente.Text
passw1 = password.Text
```

```
caratteri =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
ok1 = True
For i = 0 To user1.Length - 1
   car = user1.Substring(i, 1)
   If Not caratteri.Contains(car) Then
      ok1 = False
      Exit For
   End If
Next
ok2 = True
For i = 0 To passw1.Length -1
   car = passw1.Substring(i, 1)
   If Not caratteri.Contains(car) Then
      ok2 = False
      Exit For
   End If
Next
If (ok1 = False Or ok2 = False) Then
   lbl2.Text = "Caratteri non consentiti"
Else
   ' accesso alla tabella Utenti del database
   dbconn = New OleDbConnection(". . . . . .
```

Dopo *Else* c'è la parte di codice per l'accesso al database come nella pagina presentata in precedenza.

Il codice precedente utilizza alcune **funzioni stringa** predefinite del linguaggio Visual Basic:
- **Lenght** restituisce la lunghezza di una stringa;
- **Substring** estrae una sottostringa da una stringa;
- **Contains** controlla se una stringa contiene una sottostringa indicata come argomento e restituisce un valore *Boolean*.

• Validazione con controllo dei comandi SQL

La terza modalità controlla che le stringhe fornite dall'utente non contengano comandi SQL. Le parole da controllare sono contenute in un array (**ArrayList**), le stringhe vengono inserite nell'array con il metodo **Add**.
La funzione stringa **Contains** controlla se una stringa contiene una sottostringa con il comando SQL e restituisce un valore *Boolean*.

```
Dim user1, passw1 As String
Dim ok1, ok2 As Boolean
Dim i As Integer
Dim caratteri As New ArrayList
caratteri.Add("select")
caratteri.Add("insert")
caratteri.Add("update")
caratteri.Add("delete")
caratteri.Add("drop")
```

```
caratteri.Add("alter")
user1 = utente.Text
passw1 = password.Text
ok1 = True
For i = 0 To 5
   If user1.Contains(caratteri(i)) Then
      ok1 = False
      Exit For
   End If
Next
ok2 = True
For i = 0 To 5
   If passw1.Contains(caratteri(i)) Then
      ok2 = False
      Exit For
   End If
Next
If (ok1 = False Or ok2 = False) Then
   lbl2.Text = "Caratteri non consentiti"
Else
   ' accesso alla tabella Utenti del database
   dbconn = New OleDbConnection(". . . . . .
```

Anche in questo caso, dopo *Else* c'è la parte di codice per l'accesso al database e il controllo dell'utente.