L'uso di query parametriche protegge dalla possibilità di subire attacchi di SQL Injection. La query diventa la seguente:

```
SqlCommand cmd = new SqlCommand( "Insert into Utenti Values(@username,@password)",
conn)
cmd.Parameters.AddWithValue ("@username",TextBox1.text)
cmd.Parameters.AddWithValue ("@usercode",TextBox2.text)
cmd.ExecuteNonQuery();
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/89.html,
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection').

7.10.8 XPath Injection

Come riconoscerla

Gli attacchi XPath Injection sono possibili quando un sito Web utilizza le informazioni fornite dall'utente per costruire una query XPath per i dati XML. Inviando informazioni intenzionalmente malformate nel sito Web, un utente malintenzionato può scoprire come sono strutturati i dati XML o accedere a dati a cui normalmente non avrebbe accesso. Potrebbe persino essere in grado di elevare i suoi privilegi sul sito Web se i dati XML vengono utilizzati per l'autenticazione (come un file utente basato su XML).

Come difendersi

- Evitare che la costruzione della query XPath dipenda dalle informazioni inserite dall'utente. Possibilmente mapparla con i parametri utente mantenendo la separazione tra dati e codice. Nel caso fosse necessario includere l'input dell'utente nella query, questo dovrà essere precedentemente validato.
- Validare tutti gli input, indipendentemente dalla loro provenienza. La validazione dovrebbe essere basata su una white list (si dovrebbero accettare solo i dati che adattano a una struttura specificata, scartando quelli che non la rispettano). Occorre controllare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list).

Esempio:
Un esempio di una ricerca all'interno din un docuemento XML a partire da input esterno non verificato:

```
customers.SelectNodes("//customer[@name='" + txtUser.Text + "' and
    @password='" + txtPassword.Text + "']")
```

Il codice dovrebbe essere precompilato come il seguente. Si tratta, come si può vedere, di una query parametrica, la stessa soluzione valida per mitigare il rischio delle SQL injection:

```
XPathNodeIterator custData = XPathCache.Select(
    "//customer[@name=$name and @password=$password]",
    customersDocument,
    new XPathVariable("name", txtName.Text),
    new XPathVariable("password", txtPassword.Text));
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/643.html,
CWE-643: Improper Neutralization of Data within XPath Expressions ('XPath Injection').

7.11 AJAX

AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Rich Internet Application). Le vulnerabilità di questo linguaggio sono molto simili a quelle presenti nel linguaggio JavaScript.
Generalmente infatti si tratta di una tecnologia che fa uso di Javascript per veicolare dati, senza dover ricaricare la pagina corrente.
I dati vengono trasmessi nel formato XML.
Di seguito l'elenco delle principali vulnerabilità e delle relative contromisure da adottare.