```
        ProtocolType.Tcp);

        TcpClient client = new TcpClient("example.com", 80);
        UdpClient listener = new UdpClient(80);
    }
```

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/99.html,
CWE-99: Improper Control of Resource Identifiers ('Resource Injection').

7.6.7 SQL Injection

Come riconoscerla

I dati forniti dall'utente, in un modulo (form) o attraverso i parametri URL, devono essere sempre considerati non attendibili e potenzialmente corrotti. La composizione dinamica di query SQL, a partire da dati non verificati, consente agli aggressori di inserire valori appositamente predisposti per modificarne il contenuto e il significato. Gli attacchi di SQL injection possono leggere, modificare o eliminare informazioni riservate dal database e talvolta persino metterlo fuori uso o eseguire comandi arbitrari di sistema operativo.

Come difendersi

- In genere, la soluzione consiste nel fare affidamento su query parametriche, piuttosto che sulla concatenazione di stringhe. Questa tecnica fornisce un valido escaping dei caratteri pericolosi.
- Validare tutti gli input, indipendentemente dalla loro provenienza. La validazione dovrebbe essere basata su una white list: dovrebbero essere accettati solo i dati che adattano a una struttura specificata, scartando quelli che non rispettano la white list. Occorre controllare il tipo del dato, la sua dimensione, l'intervallo di validità (range), il formato ed eventuali valori attesi (white list).
- Limitare l'accesso agli oggetti e alle funzionalità di database, in base al "Principle of Least Privilege" (non fornire agli utenti permessi più elevati di quelli strettamente necessari per svolgere il loro lavoro).

Esempio:
Codice vulnerabile:

```
public IActionResult Autenticazione(string nomeUtente)
{
  // Non sicuro. Un utente malintenzionato può aggirare l'autenticazione passando
  // nomeUtente con il valore "' or 1=1 or ''='";
var query = "SELECT * FROM Utenti WHERE Nome = '" + nomeUtente + "'";
var nomeUtenteExists = _context.nomeUtentes.FromSql(query).Any();

   return Content(nomeUtenteExists ? "success" : "fail");
}
```

Codice sicuro:

```
public IActionResult Autenticazione(string nomeUtente)
{
var query = "SELECT * FROM Utenti WHERE Username = {0}"; // Safe
var userExists = _context.Users.FromSql(query, nomeUtente).Any();
return Content(userExists ? "success" : "fail");
}
```

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').

7.6.8 XPath Injection

Come riconoscerla