```
                        break;
                case "smtp":
                        portNum = 25;
                        break;
                default:
                        portNum = 80;
                }
                try {
                        ServerSocket serverSocket = new ServerSocket(portNum);
                } catch (Exception e) {
                        System.err.println("Caught Exception: " + e.getMessage());
                }
        }
}
```

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

7.2.7 SQL injection

Come riconoscerla

Si verifica quando l'input non verificato viene utilizzato per comporre dinamicamente uno statement SQL che poi verrà eseguito sulla base dati. Adeguatamente manipolati, i parametri di input possono modificare le query in maniera sostanziale, causando danni di impatto notevole, come l'inserimento di dati malevoli, la cancellazione e la modifica di record e la rivelazione indebita di informazioni riservate. Se i dati utilizzati per la SQL injection sono memorizzati nel database o nel file system in generale, si parla di SQL injection di second'ordine (second order SQL injection).

Come difendersi

- Come prima misura, occorre validare l'input, sottoponendolo a rigidi controlli, come già illustrato nei punti precedenti;
- Le query SQL non devono mai essere realizzate concatenando stringhe con l'input esterno. Si devono invece utilizzare componenti di database sicuri come le stored procedure, le query parametrizzate e le associazioni degli oggetti (per comandi e parametri;
- Una soluzione che può essere d'aiuto consiste nell'utilizzazione di una libreria ORM, come EntityFramework, Hibernate o iBatis;
- Occorre limitare l'accesso agli oggetti e alle funzionalità del database, in base al "Principle of Least Privilege" (non fornire agli utenti permessi superiori a quelli strettamente necessari.

Esempio:
Codice vulnerabile

```
String q='SELECT r FROM User r where r.userId=''+user+''';
Query query=em.createQuery(q);
List users=query.getResultList();
```

Codice sicuro

```
Query query=em.createNamedQuery('User.findByUserId');
query.setParameter('userId', user);
List users=query.getResultList();
```

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').

7.2.8 XPath injection

Come riconoscerla