- 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:

L'applicazione utilizza una stringa inserita dall'utente per costruire una query XPath:

```
from sys import stdin
import XPath
print 'Insert item number: '
userInput = stdin.readline()
XPath.find('//item' + userInput, doc)
```

La stringa inserita dall'utente viene trasformata in un numero intero prima dell'uso nella query XPath:

```
from sys import stdin
import XPath
print 'Insert item number: '
userInput = stdin.readline()
XPath.find('//item' + str(int(userInput)), doc)
```

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

7.7.7 Resource Injection

Come riconoscerla

Quando un'applicazione definisce un tipo di risorsa o posizione in base all'input dell'utente, come un nome file o un numero di porta, questi dati possono essere manipolati per eseguire o accedere a risorse diverse.
L'attacco di "path traversal" è un caso particolare della resource injection. In tal caso a essere iniettato è un path manipolativo che punta a risorse diverse nel file system.
Se si utilizza l'input dell'utente per definire la porta sulla quale aprire un socket, si da all'utente la possibilità di introdurre una backdoor attraverso la quale potrebbe prendere il controllo del sistema.

Come difendersi

- In molti casi non è necessario aprire un socket manualmente; meglio affidarsi a librerie e protocolli esistenti.
- Tutti i dati inviati devono essere crittografati, se sono sensibili. Nel dubbio se i dati siano sensibili o possano diventarlo, meglio comunque crittografarli.
- Qualsiasi input letto dal socket deve essere validato.
- Le applicazioni non dovrebbero utilizzare l'input dell'utente per accedere a risorse del sistema. Nel caso si scelga di farlo, è obbligatorio validare l'input, per esempio attraverso una white list. Se si consente la creazione di socket, controllare scrupolosamente questo tipo di attività.

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

7.7.8 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