Come difendersi

Per prima cosa è necessatio convalidare tutti gli input, indipendentemente dalla fonte: la convalidazione dovrebbe essere basata su una white list (una lista di valori ammessi), per cui verrebbero accettati solo i dati compresi e rifiutati tutti gli altri.
Occorre controllare, oltre che i valori siano fra quelli ammessi o che rientrino in un determinato intervallo di validità, se corrispondano alle attese anche il tipo, la dimensione e il formato dei dati in input.
Un altro accorgimento consiste nell'encoding (codifica) di tutti i dati dinamici, cioè nella neutralizzazione dei caratteri pericolosi, in modo da rendere inattivi eventuali inserimenti malevoli. La codifica dovrebbe essere sensibile al contesto, in base al tipo di dato che si vuole neutralizzare: se ci si aspetta che possa esserci codice HTML abusivo, occorre codificare gli eventuali tag HTML, se ci si potrebbe trovare di fronte a uno script, allora bisogna codificare gli elementi sintattici di Javascript, ecc.
Si consiglia di utilizzare la libreria di codifica ESAPI o le funzioni di libreria sistema incorporate.
Nell'intestazione di risposta Content-Type HTTP, definire esplicitamente la codifica dei caratteri (charset) per l'intera pagina
Impostare la flag httpOnly sul cookie della sessione, per impedire che eventuali attacchi XSS possano manometterlo.

Per ulteriori informazioni si veda: http://cwe.mitre.org/data/definitions/79.html
CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

7.3.2 Resource Injection

Come riconoscerla

L'applicazione apre un socket di rete, per l'ascolto delle connessioni in entrata, utilizzando dati non attendibili. In questo modo si consente a un utente malintenzionato di controllarlo.

L'attaccante potrebbe perciò essere in grado di aprire una backdoor che gli consenta di connettersi direttamente al server delle applicazioni, acquisendo il controllo del server o esponendolo ad altri attacchi indiretti. In particolare, modificando il numero di porta del socket, potrebbe essere in grado di aggirare controlli di rete deboli, mascherando l'attacco da parte di altri dispositivi di rete.
Una resource injection può essere sfruttata anche per bypassare i firewall o altri meccanismi di controllo degli accessi. Si può anche utilizzare l'applicazione come proxy per la scansione delle porte delle reti interne e per l'accesso diretto ai sistemi locali; oppure per indurre un utente a inviare informazioni riservate a un server fraudolento.

Come difendersi

Non consentire a un utente di definire i parametri relativi ai sockets di rete.
Questo esempio in PLSQL prende un path di tipo URL da una CGI ed esegue il download del file contenuto. La vulnerabilità è rappresentata dalla possibilità per un utente malintenzionato di modificare il path o il nome del file, ricevendo dal server del contenuto arbitrario e potenzialmente dannoso.
Esempio:

```
filename := SUBSTR(OWA_UTIL.get_cgi_env('PATH_INFÒ), 2);
WPG_DOCLOAD.download_file(filename);
```

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

7.3.3 SQL Injection

Come riconoscerla