Come difendersi

Per prima cosa, occorre convalidare tutti gli input, indipendentemente dalla loro provenienza: la convalidazione dovrebbe essere basata su una white list (una lista di valori ammessi), per cui verrebbero accettati solo i dati che corrispondono, e verrebbero 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 nel codificare completamente tutti i dati dinamici (encoding) in modo da neutralizzare eventuali inserimenti malevoli. La libreria ESAPI fornisce funzioni di encryption per una grande varietò di tipologie di input atteso 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.
Definire in modo esplicito la codifica dei caratteri (charset) per l'intera pagina nell'intestazione di risposta HTTP Content-Type.
Impostare il flag HttpOnly a true, per evitare tentativi di furto tramite la lettura tramite script dei cookie di sessione.
Esempio:
Nel codice che segue, un valore preso dalla request viene scritta direttametne sulla response:

```
String nomeUtente = request.getParameter("nome");
response.getWriter().write("Nome Utente: " + nomeUtente);
```

Il rimedio consiste nel filtrare il valore in input:

```
response.getWriter().write(ESAPI.encoder().encodeForHTML
                                ( request.getParameter( "nome" ) ) );
```

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').

7.2.2 Code injection

Come riconoscerla

Accade quando l'applicazione utilizza, concatenandole, stringhe in input non bonificate. L'attaccante potrebbe introdurvi script che potrebbero essere eseguiti direttamente nell'application server. Ciò potrebbe portare ad azioni indesiderate. È simile al Cross Site Scripting, ma qui il codice introdotto non viene integrato nella pagina HTML, ma viene eseguito a sé.

Come difendersi

Evitare di eseguire del codice dinamicamente, specialmente se costruito a partire da input proveniente dall'esterno.
Occorre verificare sempre l'input, fissando controlli rigidi che impediscano di immettere caratteri e tipi di dati potenzialmente dannosi. L'optimum è designare una white list di valori ammessi e scartare tutto ciò che non vi rientra.

Esempio:
Il seguente codice permette di eseguire un file puntato dinamicamente in base al valore dell'input proveniente dall'esterno, senza controlli.

```
public class CodeInjection {
       static void main(String[] args){
              System.load(args[0]);
       }
}
```

Nel codice seguente, l'input viene controllato contro una white list di valori ammessi: