Un altro accorgimento consiste nel codificare completamente tutti i dati dinamici (encoding) in modo da neutralizzare 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.
Nell'intestazione di risposta HTTP Content-Type, definire in modo esplicito la codifica dei caratteri (charset) per l'intera pagina.
Impostare il flag HttpOnly a true, per evitare tentativi di furto tramite la lettura tramite script dei cookie di sessione.

Esempio:
Se ci si affida a programmi C/C++ per una web application, il pericolo è insito nella specifica CGI (Common Gateway Interface), che offre l'opportunità di accedere al file system.
La necessità di bonificare l'input può essere soddisfatta sottoponendo le stringhe in entrata a una routine di encoding come la seguente:

```
void encode(std::string& data) {
  std::string buffer;
  buffer.reserve(data.size());
  for(size_t pos = 0; pos != data.size(); ++pos) {
    switch(data[pos]) {
      case '&': buffer.append("&amp;");    break;
      case '\"': buffer.append("&quot;");   break;
      case '\'': buffer.append("&apos;");   break;
      case '<': buffer.append("&lt;");    break;
      case '>': buffer.append("&gt;");    break;
      default:  buffer.append(&data[pos], 1); break;
    }
  }
  data.swap(buffer);
}
```

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').

7.1.2 Command Injection

Come riconoscerla

In questa tipologia di attacco, l'aggressore potrebbe eseguire comandi di sistema arbitrari sul server dell'applicazione.
Se è in grado di iniettare e fare eseguire un comando di sistema operativo, l'aggressore può eseguire qualsiasi comando, fino all'acquisizione completa del controllo del server.
La command injection è possibile se si utilizzano stringhe di input dell'utente per creare comandi di shell che poi vengono eseguiti.

Come difendersi

Di seguito un elenco delle azioni da intraprendere:

- Evitare qualsiasi esecuzione diretta di script di comandi utilizzando l'input utente. Utilizzare piuttosto API messe a disposizione dal linguaggio o da librerie di funzioni.
- Se è impossibile rimuovere l'esecuzione del comando, eseguire solo stringhe statiche che non includono l'input dell'utente.
- Validare tutti gli input, indipendentemente dalla loro provenienza. Operare una convalida dell'input attraverso una white list di valori ammessi e altri controlli, come evidenziato nel punto precedente.
- Eseguire l'applicazione utilizzando un account utente limitato che non disponga di privilegi non necessari.