- Prendere il controllo remoto di un browser;
- Ottenere un cookie;
- Modificare il collegamento ad una pagina;
- Redirigere l'utente a un URI differente dall'originale;
- Forzare l'immissione di dati importanti in form non-trusted (phishing);

Esempio:
Segue un esempio di servlet vulnerabile a Cross Site Scripting:

Contromisure

Al fine di evitare il Cross Site Scripting è di fondamentale importanza verificare l'input che proviene dall'esterno, prima di utilizzarlo all'interno della web application.
Tale verifica comporta l'utilizzazione di funzioni di escaping, le quali rilevano caratteri ritenuti pericolosi, ad esempio <, >, &, /, ' ," , sostituendoli con del testo.
Esistono a tal proposito molte librerie che consentono di neutralizzare tag html, come anche pezzi di codice Javascript.

6.1.6 Directory Traversal

Le problematiche di Directory Traversal, note anche come Dot-Dot Vulnerability, si verificano quando un aggressore ha la possibilità di immettere dell'input che verrà utilizzato dall'applicazione per accedere ad un file in lettura e/o scrittura. Solitamente le applicazioni vietano l'utilizzo di percorsi completi (ad esempio `/etc/shadow o c:\winnt\system32\cmd.exe`) ma in assenza di controlli sui dati acquisiti in ingresso, un aggressore può ugualmente raggiungere e acquisire il contenuto di un file residente all'esterno dell'area a lui accessibile, anteponendo una sequenza di punti al nome dello stesso (ad esempio `../../../../nomefile` oppure `.../.../.../nomefile`). Poiché le problematiche di Directory Traversal sono state utilizzate dagli aggressori fin dallo sviluppo dei primi Web Server, sono oggi tra le più note. Non a caso molte applicazioni vengono progettate in modo da mitigare il rischio del loro sfruttamento. Alcune fra queste tentano di correggere i dati non validi acquisiti in input, trasformandoli in un flusso considerato valido. La casistica ha comunque dimostrato che è quasi sempre sconsigliato (al di fuori di specifiche eccezioni) affidarsi all'input utente per costruire nomi file e percorsi all'interno dell'applicazione, in quanto vi è un'alta possibilità di introdurre ulteriori fattori di instabilità o insicurezza all'interno del software sviluppato.

Esempio:
Se nel codice sorgente viene utilizzato il nome del file:

```
BufferedReader reader = new BufferedReader(new FileReader("data/"+ argv[1]));
String line = reader.readLine();
while(line!=null) {
   System.out.println(line);
   line = reader.readLine();
}
```