Il codice sorgente può essere manomesso, per ottenere l'accesso a un file sensibile, sostituendo il nome del file con il percorso al file 'sensibilè al quale si vuole accedere:

```
../../../../etc/password
```

Contromisure

In una web application si dovrebbe evitare di utilizzare percorsi di file system inseriti dall'utente. Se l'utente dovesse scegliere un file, occorrerebbe limitare la selezione imponendogli una scelta limitata di file ammessi (white list), attraverso un indice numerico. Nel caso in cui fosse necessario utilizzare un percorso fornito dall'utente, occorrerebbe verificarlo e/o sottoporlo a escaping.
Un'altra contromisura, valida soprattutto sui sistemi Unix/Linux, potrebbe essere quella di creare una chroot jail, ossia non permettere di sfuggire alla root accessibile dalla web application, in maniera tale da salvaguardare le directory critiche del sistema operativo. Lo stesso risultato potrebbe essere raggiunto consentendo l'accesso a un utente che ha accesso limitato, la cui home directory coincida con la document root.

6.1.7 SQL Injection

SQL Injection è una problematica che colpisce principalmente le applicazioni Web che s'interfacciano a un layer di back-end che utilizza un database relazionale, anche se non unicamente circoscrivibile a quest'ambito. La SQL Injection è, infatti, una vulnerabilità che affligge tutte le applicazioni (anche client/server) che interrogano un DB. Si verifica quando uno script o un'altra componente applicativa non filtra opportunamente l'input passato dall'utente, rendendo possibile per un aggressore l'alterazione della struttura originaria della query SQL, attraverso l'utilizzo di caratteri speciali (ad esempio apici e virgolette) o mediante la concatenazione di costrutti multipli (ad esempio utilizzando la keyword SQL UNION). A seconda delle circostanze e del tipo di database server con cui l'applicazione si interfaccia, l'aggressore può sfruttare una problematica di SQL Injection per:

- Bypassare i meccanismi di autenticazione di un portale (ad esempio forzando il ritorno di condizioni veritiere alle procedure di controllo);
- Ricostruire il contenuto di un Database (ad esempio localizzando le tabelle contenenti i token delle sessioni attive, visualizzando le password degli utenti cifrate/non cifrate o altre informazioni di natura critica);
- Aggiungere, alterare o rimuovere i dati già presenti nel Database;
- Eseguire stored-procedures.

Si riportano di seguito tre problematiche di SQL Injection che rappresentano le tecniche di base da cui derivano tutti i casi possibili:

- Iniezione di una seconda query mediante il carattere ";"
  Esempio:
  Si consideri la query: $sql = "SELECT * from utenti WHERE id=$id";
  Se il parametro $id fosse acquisito da input utente e inizializzato alla stringa: 1; DROP table utenti
  La query risultante sarebbe: SELECT * from utenti WHERE id=1; DROP table utenti che causerebbe la rimozione da parte dell'aggressore della tabella utenti. Le query multiple non sono comunque supportate da tutti I database server.
- Modifica della query attraverso introduzione del commento '--'
  Esempio:
  Si consideri la query: $sql = "SELECT * from utenti WHERE login='$login' AND password='$password'";
  Se il parametro $login fosse acquisito da input utente ed inizializzato alla stringa: xyz' OR 1=1 –-
  La query risultante sarebbe: SELECT * from utenti WHERE login='xyz' OR 1=1 --' AND password='' ed il database tratterebbe la parte successiva a "--" come commento, ignorandola e permettendo quindi all'aggressore di accedere senza specificare alcuna password.
- Iniezione di caratteri jolly ed eliminazione di parte della query: