Subject: Re: PHP issue
Posted by danpaul88 on Sat, 06 Nov 2010 10:03:17 GMT
View Forum Message <> Reply to Message

Actually I disagree Crimson, protecting database inputs against injection attacks is something you should learn as early as possible so that it becomes second nature when coding in PHP.

cnc95fan, consider what would happen if I submitted your search form with the following;

```
$_POST['bookid'] = '0; DROP TABLE books';
```

Based on your current code, this would result in the following query being run;

```
SELECT * FROM books WHERE bookid=0; DROP TABLE books;
```

Obviously this is a huge security problem, however there is a simple solution: Run anything from POST or GET which will go into a database through functions to verify it is valid. For numerical (int, float) values use something like;

```
function prepare_db_number($number)
{
  if ( is_numeric($number) )
  {
    return $number;
  }
  return 0;
}
```

This is an extremely simple function which checks the input is numeric and returns it if it is. If it is NOT numeric it returns 0, preventing any SQL injection attacks through that variable.

For strings you can use something a bit like this;

```
function prepare_db_string( $string, $encode_html_entities = FALSE )
{
 // If magic quotes are enabled then strip the existing slashes from the string first
 if(get_magic_quotes_gpc())
  $result = stripslashes(trim($string));
 else
  $result = trim($string);

 // Encode HTML entities if required
 if ( $encode_html_entities === TRUE )
  $result = htmlentities($result);

 // Return MySQL safe string
 return mysql_real_escape_string($result);
}
```

This function does several things - firstly it trims whitespace from around the input string (ie: spaces or tabs before or after any actual content) and, if magic quotes are enabled, it removes the slashes (otherwise you would end up with some things double escaped). Secondly, it optionally converts special characters to their HTML entities, this is useful if you know the string is going to be output directly to HTML and you need to ensure there are no HTML tags inside of it, for example a forum post.

Finally it uses mysql_real_escape_string to escape any character sequences which could be used to break out of the string and inject an additional query.