

MySQL Prevent SQL Injection Attacks

<http://www.learnphponline.com/security/sql-injection-prevention-mysql-php>

SQL Injection Prevention: The Solution In Preventing SQL Attacks

We could've given you the answer right away, but what fun would that have been? Too often, PHP developers are becoming lazy and not following proper security tactics the way they should. By reaching this point in the lecture, you've increased your knowledge on how SQL injections are used, how not to prevent the attacks, and finally: you'll learn the right way to keep injection attacks at bay.

We'll accomplish this last feat with a simple function that the developers of PHP made especially for SQL injections. We call this function `mysql_real_escape_string()` - take a look at it below:

`mysql_real_escape_string()` In Action!

```
$name = "John";  
$name = mysql_real_escape_string($name);  
$SQL = "SELECT * FROM users WHERE username = '$name'";
```

Although for a more practical use, we would have the `$name` variable pointed to a POST result, as seen below:

`$_POST` Can Dig In On The Action Too!

```
$name = mysql_real_escape_string($_POST['user']);
```

And we can even make things easier by putting it into one line:

All In One Line Now; Here We Go!

```
$SQL = "SELECT * FROM users where username =  
"mysql_real_escape_string($_POST['user']);
```

So what's the output like if malicious users try to get access to our SQL server? Well glad you asked! Their attempts may look something like this:

Cause And Effect With `mysql_real_escape_string()`

```
$malicious_input = "' OR 1'";  
// The Above Is The Malicious Input. Don't Be Scared!  
// With The mysql_real_escape_string() usage, the following is  
obtained:  
  
    \' OR 1\  
// Notice how the slashes escape the quotes! Now users can't enter  
malicious data
```

And the best part is, they just wasted their time and effort for nothing. Now how's that for vindication!

SQL Injection: Closing Comments

We've learned quite a bit today. SQL injections are bad. All Internet users are equally as bad. Protecting against both ensures a happy and stable web application. And above all else, never use magic quotes! Despite their cleverly disguised name, we've found no evidence of magic.

Lastly, note that there are libraries and classes that can help aid in the fight against SQL injection. Prepared statements are plausible as well, but as for us, we enjoy sticking to the `mysql_real_escape_string()` function for less headaches.

Bottom Line: `mysql_real_escape_string()` - It doesn't have a magically awesome name, but it's 24 characters worth of SQL injection-protection goodness.