

Forum:[Hack Forums](#)

Thread: [How can I improve the security on this?](#)

Webman.io - June 25, 2012, 2:45 am

<https://hackforums.net/showthread.php?tid=2564825>

How can I improve the security on this above?

J?mes - June 25, 2012, 4:18 am

Mysql -> Make sure you always use `mysql_real_escape_string` for escaping queries or you may get SQLi'd.

Preventing XSS -> `strip_tags` and `htmlentities` is always a good start (If you're putting the input back into a html input (input tags) then you will need more protection.

That's the basic stuff :).

Phizo - June 25, 2012, 6:45 am

(Jun 25, 2012 04:18 AM)JamesHammond Wrote: [\[CITE\]](#) Mysql -> Make sure you always use `mysql_real_escape_string` for escaping queries or you may get SQLi'd.

Preventing XSS -> `strip_tags` and `htmlentities` is always a good start (If you're putting the input back into a html input (input tags) then you will need more protection.

That's the basic stuff :).

Pretty much this.

If you need some code to look at then here is some of mine:

PHP Code

```
<?php
```

```
error_reporting(0); // To prevent disclosure of any useful information.
```

```
require('includes/config.php');
```

```
session_start();
```

```
$action = $_GET['action'];
```

```
if(!isset($_SESSION['user']) || $action == 'logout') // Headers must be sent before the body.  
header('Location: index.php');
```

```
// Functions //
```

```
function strip_attr($input) // strip_tags() can be exploited by using HTML attributes maliciously.  
{  
    $bad_attr =  
'onmouseover|onerror|onclick|ondblclick|onfocus|onload|onunload|onchange|onreset|onselect|onsubmit|on  
keyup|javascript:';  
  
    return preg_replace("/{ $bad_attr }|style\s*(?==)/is", '[REMOVED]', $input);  
}
```

```

function login()
{
    $username = mysql_real_escape_string($_POST['username']);
    $password = sha1($_POST['password']);

    $verify_user = "SELECT `username` FROM `users` WHERE `username` = '{$username}'
AND `password` = '{$password}'";
    $result = mysql_query($verify_user) or report_error($verify_user);

    if(mysql_num_rows($result) == 1)
    $_SESSION['user'] = mysql_result($result, 0);
}

function logout()
{
    unset($_SESSION['user']);
}

function changePassword()
{
    $user = mysql_real_escape_string($_SESSION['user']);
    $old_pass = sha1($_POST['old_password']);
    $new_pass = sha1($_POST['new_password']);

    $verify_old = "SELECT `password` FROM `users` WHERE `password` = '{$old_pass}'";
    $update_pass = "UPDATE `users` SET `password` = '{$new_pass}' WHERE `username` = '{$user}'";

    if(isset($_POST['chpass']))
    {
        if(mysql_num_rows(mysql_query($verify_old)) == 1)
        {
            if($_POST['new_password'] == $_POST['confirm_password'])
            {
                mysql_query($update_pass) or report_error($update_pass);
                echo 'Your new password has been updated and emailed to you.';

                mail('*@hotmail.com, info@*.com.au', 'Password Change',
                "You recently changed your password to the following: {$_POST['new_password']}
. This is just a reminder incase you forget it.");
            }
            else
            {
                echo 'The new password fields did not match.';
            }
        }
        else
        {
            echo 'The old password you entered was incorrect.';
        }
    }
}

```

```

function select_body($id)
{
    $select_body = "SELECT `body` FROM `misc_content` WHERE `id` = {$id}";
    $row = mysql_fetch_assoc(mysql_query($select_body));

    echo $row['body'];
}

function update_body($id)
{
    $insert_body = mysql_real_escape_string(strip_tags(strip_attr($_POST['new_body']),
'<p><center><b><i><li><a><img>'));
    $update_body = "UPDATE `misc_content` SET `body` = '{$insert_body}' WHERE `id` = {$id}";

    if(isset($_POST['submit']))
    {
        mysql_query($update_body) or report_error($update_body);
        echo 'Page has been modified and updated successfully.';
    }
}

```

?>

Another thing is to use custom error handling functions, such as my report_error() function. This is to prevent disclosure of any useful information that can be obtained by creating a MySQL error. If you are expecting a lot of errors then log them into a table, I have them sent to my email because I don't plan on getting any often.

PHP Code

```

<?php
function report_error($query)
{
    $email = '*@hotmail.com';
    $subject = 'MySQL error.';
    $message = "IP: {$_SERVER['REMOTE_ADDR']} \n URL: http://{$_SERVER['SERVER_NAME']}{
$_SERVER['REQUEST_URI']} \n\n MySQL query: {$query} \n\n MySQL error: " . mysql_error();

    mail($email, $subject, $message);
    die('Oops, an error has occurred. The administrator has been notified.');
```

Haxalot - June 25, 2012, 12:19 pm

If you're using the HTTP GET method to perform actions; such as using ?action=delete which would be used to delete something, then use Nonces. If you don't know what they are; they're methods to stop people sending you links, such as: www.site.com/acp.php?id=3&action=delete, which would presumably delete your user if clicked. They do this by preventing actions initiated by links or REQUESTS from being used more than once. Every time a link/form is printed on the screen, your Nonce will embed a key/value pair to be sent to the receiving script. Every time that script is called, it checks for the key/value pair and then authenticates it. If it passes, the action is performed, if it fails, the action is not performed.

You can read up more about these nifty security features [here](#).

?Cooldude? - June 25, 2012, 2:20 pm

(Jun 25, 2012 12:19 PM)Haxalot Wrote: [\[CITE\]](#) If you're using the HTTP GET method to perform actions; such as using ?action=delete which would be used to delete something, then use Nonces. If you don't know what they are; they're methods to stop people sending you links, such as: www.site.com/acp.php?id=3&action=delete, which would presumably delete your user if clicked. They do this by preventing actions initiated by links or REQUESTS from being used more than once. Every time a link/form is printed on the screen, your Nonce will embed a key/value pair to be sent to the receiving script. Every time that script is called, it checks for the key/value pair and then authenticates it. If it passes, the action is performed, if it fails, the action is not performed.

You can read up more about these nifty security features [here](#).
So that is just like MyBB's post key?

Darkvengance - June 25, 2012, 4:46 pm

(Jun 25, 2012 02:45 AM)Mmm Wrote: [\[CITE\]](#) <https://hackforums.net/showthread.php?tid=2564825>

How can I improve the security on this above?

Check out this ebook that I wrote a few months back on web application security. It covers basically all the topics and is directed towards PHP.

[Web Application Security](#)

Imbue - June 25, 2012, 5:23 pm

If you're looking at using a login system which connects to a database may be an idea to start of with no using just MySQL but look into MySQLi or PDO because they offer improvements over other things.

PDO also offers prepare statements which can in a way make more sense when it comes to securing a query. Yet you'll still need to strip any malicious HTML out of anything you plan to display to users.

YERANUK - June 26, 2012, 3:26 pm

What Haxolot stated, is called CRSF, it's a way of sending commands to a website using your own forms (only way to make a post request) or a simple link and tricking others into using them to use their privileges to your own advantage.

Imagine if someone had a Paypal type website, and someone manages to make you click a form that simulates a a transaction on their own website (but its pointing to the paypal type website), they could take all your money without having to login to your account.