

Forum:[Hack Forums](#)

Thread: [Registration script.](#)

VipVince - April 22, 2014, 11:35 am

Made this a while back. Its appears to be working fine. Excuse the front end appearance, only concerned with the back-end functionality at the moment. Any feed back/constructive criticism is welcome.

register.php

PHP Code

```
<?php
<html>
<head>
<title>Registration page</title>
</head>

<body>
<form method="POST" method="">
<center>
<h1>Thank you for Registering at <?php

echo $_SERVER['SERVER_NAME'];

?></h1>
Username:<input type="text" name="username" autocomplete="off" /> <br />
Password:<input type="password" name="password" autocomplete="off" /> <br />
Re-Enter Password:<input type="password" name="password2" autocomplete="off" /> <br />
Email:<input type="text" name="email" autocomplete="off" /> <br />
<input type="submit" name="submit" value="Register" /></center>
</form>
</body>
</html>

<?php

require '/password.php';
require '/config.php';

if (isset($_POST['submit']))
{

    // database connection
    $conn = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);

    if (empty($_POST['username']) || empty($_POST['password']) || empty($_POST['password2']) ||
        empty($_POST['email']))
    {
        die("Error: All fields must be filled in!");
    } else
```

```

{

$username = $_POST['username'];
$password = $_POST['password'];
$password2 = $_POST['password2'];
$email = $_POST['email'];
$userIp = ($_SERVER['REMOTE_ADDR']);
$joinDate = date("Y-m-d H:i:s");

//secure the password
$hashedpass = password_hash($password, PASSWORD_DEFAULT);
$hashedpass2 = password_hash($password2, PASSWORD_DEFAULT);

//compare the plain text passwords
if ($password !== $password2)
{
    die("Error: Passwords do not match!");
}

//check if email is valid
if (!filter_var($email, FILTER_VALIDATE_EMAIL))
{
    die("Error: E-mail address is not valid!");
}

if (strlen($username) > 15)
{
    die("Username character limit exceeded!");
}

if (strlen($username) < 6)
{
    die("Username must contain more than 5 characters!");
}

if (strlen($password) < 6)
{
    die("Password must contain more than 5 characters!");
}

if (strlen($email) > 30)
{
    die("Email character limit exceeded!");
}

//check if the username is already registered. we use prepare() to get the data ready for assignment
$stmt = $conn->prepare(
"SELECT COUNT(`username`) FROM members WHERE `username` = :name");

//notice we use :name which is a placeholder for the username variable value, we execute the comm

```

```

and.
$stmt->execute(array(':name' => $username));

//we use pdos fetch to see if its equal to 1 which means it exists in the db
if ($stmt->fetch(PDO::FETCH_COLUMN) >= 1)
{
    die("Username: " . htmlentities($username) . " already exists in our database!");
}

//check if the email is already registered. Again we start with prepare.
$stmtz = $conn->prepare(
"SELECT COUNT(`email`) FROM members WHERE `email` = :email");

//execute() again assigning the placeholder with the variable data
$stmtz->execute(array(':email' => $email));

if ($stmtz->fetch(PDO::FETCH_COLUMN) >= 1)
{
    die("Email: " . htmlentities($email) . " already exists in our database!");
}

//get data ready for db insert, notice the insert into command in caps.
$sql =
"INSERT INTO members (username, password, email, ip, joindate) VALUES (:username, :password
, :email, :ip, :joindate)";

//prepare statement before inserting.
$q = $conn->prepare($sql);

//execute statement, with variable data into placeholders
$q->execute(array(
    ':username' => $username,
    ':password' => $hashedpass,
    ':email' => $email,
    ':ip' => $userip,
    ':joindate' => $joinDate));
if ($q)
{
    echo "<b>Thank you for registering " . htmlentities($username) .
        "! <a href='login.php'>Click here to login</a></b>";

    //send email to newly registered user
    $to = $email;
    $subject = "Your registration details.";
    $message = "Thank you for registering.Your credentials are stated below:<br>Username:
$username Password: $password";
    $message = wordwrap($message, 70);
    $from = "webadmin@host.com";
    $headers = "From: $from";
    mail($to, $subject, $message, $headers);
}
}

```

```

}

?>
login.php
PHP Code
<?php
<html>
<head>
<title>Login page</title>
</head>
<h1><center>Login @<?php

echo $_SERVER['SERVER_NAME'];

?></center></h1>
<body>
<form method="POST"><center>
Username:<input type="text" name="username" autocomplete="off" /><br />
Password:<input type="password" name="password" autocomplete="off" /><br />
<input type="submit" name="submit" value="Login" /></center>
</form>
</body>
</html>

<?php

session_start();
require '/password.php';
require '/config.php';

if (isset($_POST['submit']))
{

    // database connection
    $conn = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);

    $username = $_POST['username'];
    $password = $_POST['password'];

    if ($username == "")
    {
        die("You must enter a username!");
    }
    if ($password == "")
    {
        die("You must enter a password!");
    }
    //login query
    $result = $conn->prepare("SELECT password FROM members WHERE username = :user");
    $result->execute(array(':user' => $username));
    $row = $result->fetch(PDO::FETCH_ASSOC);

```

//password_verify means that you need to select the hashed password from the database, and then compare that value to the plaintext password they entered in the \$password variable, and introduce sessions.

```
if (password_verify($password, $row['password']))
{
    $_SESSION['username'] = $username;
    header("location:home.php");
} else
{
    echo "Access denied!";
}
}
```

?>

Thanks in advance.

VV.

Tony S - April 22, 2014, 3:32 pm

You should pick a certain coding style then follow it all the way through your scripts and projects. The fact that you keep switching between single quotes and double quotes even though you're not interpolating variables is pretty.. uh.. annoying? (not to sound like a dick)

VipVince - April 22, 2014, 3:34 pm

(Apr 22, 2014 03:32 PM)Tony Spilotro Wrote: [\[CITE\]](#) You should pick a certain coding style then follow it all the way through your scripts and projects. The fact that you keep switching between single quotes and double quotes even though you're not interpolating variables is pretty.. uh.. annoying? (not to sound like a dick)

You bring up a valid point, and I agree with you. Just overlooked it in the past. Will work on sticking to the same styles where they are applicable in the future.

Volim te - April 22, 2014, 3:36 pm

Why not have the DB connection code in it's own file? Saves on space (:

VipVince - April 22, 2014, 3:38 pm

(Apr 22, 2014 03:36 PM)Volim te Wrote: [\[CITE\]](#) Why not have the DB connection code in it's own file? Saves on space (:

I required it then initiated the connection via the script that I was working with. Thought that would be sufficient. Thanks for your input. :)

Tony S - April 22, 2014, 3:47 pm

Also,

PHP Code

```
<?php
$to = $email;
$subject = "Your registration details.";
$message = "Thank you for registering.Your credentials are stated below:<br>Username:
$username Password: $password";
$message = wordwrap($message, 70);
$from = "webadmin@host.com";
$headers = "From: $from";
mail($to, $subject, $message, $headers);
```

Sending someone's cleartext password in an email is a bad idea.

So... - April 22, 2014, 5:59 pm

If we are being picky, the username is vulnerable to xss injection, try inputting a username like

```
<script>alert("xss");</script>
```

U can filter either the input or output by using strip_tags and htmlentities. Of course this isn't full-proof, but it will stop the majority of attacks.

EDIT:: Didn't even notice that you are, huh...

ignore this then lol

Volim te - April 22, 2014, 10:38 pm

(Apr 22, 2014 05:59 PM)So... Wrote: [\[CITE\]](#) If we are being picky, the username is vulnerable to xss injection, try inputting a username like

```
<script>alert("xss");</script>
```

U can filter either the input or output by using strip_tags and htmlentities. Of course this isn't full-proof, but it will stop the majority of attacks.

EDIT:: Didn't even notice that you are, huh...

ignore this then lol1 use htmlspecialchars() not htmlentities()
2: remove unwanted chars rather than use the vulnerable strip_tags()
>"<script>alert(1)<script><" should bypass strip_tags()...

Sample: Only allow letters, numbers and underscores.

```
if (!preg_match('/^[a-z0-9_]*$/UD',$_POST[username])){  
# We're not ok, do not allow this username !  
}
```

VipVince - April 23, 2014, 4:07 am

Thanks for the feed back guys. So overall its quite secure?

u fux wid it - April 23, 2014, 4:11 am

Does this save the file as a batman file or a superman file?

Volim te - April 23, 2014, 4:18 am

(Apr 23, 2014 04:07 AM)VipVince Wrote: [\[CITE\]](#) Thanks for the feed back guys. So overall its quite secure?It's good but you should add LIMIT 1 to the query checking if the user exist and for logging in.

Squito - April 23, 2014, 4:56 am

A small bug in register.php:

PHP Code

```
<?php
```

```
<form method="POST" method="">
```

Also why prohibit long email addresses? Using filter_var on it is perfectly enough. And I suggest limiting characters used in username, like Volim te suggested, otherwise you'll have to use

htmlspecialchars() every time you echo out a username. Also you should put the registration form after the registration script and let it (the form) appear only when there is a problem (and echo previous values into it (with htmlspecialchars())).

In the login.php use session_start() before any HTML output, otherwise if session handling is cookie based it won't work. Instead of checking the username than the password simply check for the entered username and (hashed) password pair and if you find one (you may use count in the query, except you want to extract infos about the user, like user level, title, etc), then set the username as a session variable. Also you shouldn't switch autocomplete off on a login page.

Cathal - April 23, 2014, 5:28 am

PHP Code

```
<?php
if(!ctype_alnum($_POST['username']))
    die("Usernames must be alpha-numeric!");
```

Also, I would save the time registered as a timestamp, so it's easier to manipulate later on.