**Creating a Secure Login System the Right Way**

http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/

Making a custom login system is a common task for beginning PHP developers. Jumping right into it, however, may not be the best approach. There are several important aspects to building a login system that not only makes it work, but makes it safe.  
Updated on December 15th 2009: Added Session Control Section  
Updated on December 15th 2010: Switched to sha256

**Getting Started**

To begin with, we’ll create our login form. This doesn’t need to be anything fancy, just a couple of input fields and a submit button:

|  |
| --- |
| <form name="login" action="login.php" method="post">      Username: <input type="text" name="username" />      Password: <input type="password" name="password" />      <input type="submit" value="Login" />  </form> |

The above example is stripped down; there is no formatting or styles so it will most likely won’t look to great if you copy and paste the code. Making your form pretty is beyond the scope of this article. In the form tag, notice that it has three attributes: name, action, and method. Name identifies the form and is not very important in the context of this article. Action identifies the script that will be processing the login, often times your form and the processing code are in a single file, but this does not have to be the case. Method typically takes one of two values: post or get. If you submit a form using GET the data is URL encoded and will be visible in the address bar. If the method is post the data will not be URL encoded. As you may have guessed there is actually a lot more to it than that but the difference between post and get is a topic for another article. We want to use post.

Notice that our input fields are given name attributes, this is important as we will need to identify and access these values by this name. PHP identifies form data using the name attribute, not the id. Now that our form is complete we can move on to processing the data.

**Storing our Data**

Actually we can’t process the data just yet. Before that we need to worry about how our data will be represented in the database. There are three essential values we must store in our database: the username, password, and a salt. We will get into what a salt is later. In addition to this you may choose to store other things about your user. It is also common to give the user a numeric user id. This is not absolutely necessary but it is common for tables to have a numeric, sequential primary key. For our purposes assume our table structure is this:

|  |
| --- |
| CREATE TABLE users (      id INT NOT NULL AUTO\_INCREMENT,      username VARCHAR(30) NOT NULL UNIQUE,      password VARCHAR(64) NOT NULL,      salt VARCHAR(3) NOT NULL,      PRIMARY KEY(id)  ); |

In the above SQL code we create a table called users having columns id, username, password, and salt. Even though we could use usernames to uniquely identify users we will use the id for this instead. One reason for this is that integer comparisons are cheaper than string comparisons, so searching through a large number of users will require fewer resources. Convention is another reason.

There are a few keywords in the above code that I’ll define for you. Not null means that each tuple (a tuple is a row in our table, each user will have a row) must have a value for this column. In this case all of our columns are ‘not null’ so every user must have each of these values. Auto increment applies to numeric primary keys. It allows us to give each user a sequential id without worrying about collisions or what the most recent users id was; the database will assign each user a correct id automatically. Unique, as you may guess, means that the value must be unique. In our case, no two users may have the same username. Finally, primary key tells the database which field will uniquely identify each row. No two users can have the same id. It also creates and index on the specified column, meaning that lookups will be faster (at the expense of memory).

Each column also has a data type. The id is and integer while the remain fields are varchars. Varchars are just arrays of characters, as are strings. Varchar(30) means a sequence of 30 characters. For the username I chose 30 as the length for no particular reason; it allows for a reasonable amount of characters without letting users write a paragraph. The lengths for password and salt are important and I’ll get into that later. Now that we have created our table we are ready to process our data.

**Populating our Table**

Actually we can’t process the data just yet. You can’t login if your users table is empty. Just like on any site, you have to register before you can login. To accomplish this we will create a simple registration form.

|  |
| --- |
| <form name="register" action="register.php" method="post">      Username: <input type="text" name="username" maxlength="30" />      Password: <input type="password" name="pass1" />      Password Again: <input type="password" name="pass2" />      <input type="submit" value="Register" />  </form> |

The above code is similar to our login form code with one notable difference. In the username input field I specify the maxlength attribute as 30. This means the field can only contain 30 characters and corresponds to the username length we specified in our SQL code. Notice I don’t enforce the length of passwords even though they are defined as 40 characters in the SQL code, you will see why it is not necessary to do so later. Now that we have our registration form we can process our data.

**Sign Me Up**

Our registration data that is, we still can’t process our login data. At this point we actually get to write some PHP code (are you as excited as I am?).

register.php (part 1):

|  |
| --- |
| <?php  //retrieve our data from POST  $username = $\_POST['username'];  $pass1 = $\_POST['pass1'];  $pass2 = $\_POST['pass2'];  if($pass1 != $pass2)      header('Location: register\_form.php');  if(strlen($username) > 30)      header('Location: register\_form.php'); |

In the above code we retrieve our data from $\_POST, which is an associative array where all of the post data is stored. We also check if pass1 and pass2 are equal which is an example of validating user input. If they are not equal we use the header function to redirect back to our registration form (assume the registration form is located in a file called register\_form.php). Ideally we would want to display an error message, but for the sake of example we will keep it simple. We also check if the username exceeds 30 characters. Even though we set the form to allow no more than 30 characters it is important that we check as well; it is possible (and very simple) to bypass the limit imposed in the html code. In addition to this you should check for any other constrains you have placed on your data (valid characters, minimum length, etc).

**Hashing**

As I mentioned earlier, the length of the password varchar in the database is significant. This is because we are not actually going to (and never should) store the password in the database. We are going to store an sha256 hash which is a string always containing 64 characters. In simple terms a hash is an algorithm that maps inputs to outputs in a deterministic way. Meaning that given an input the algorithm will always produce the same output. Sha256 is an algorithm that outputs a 64 digit hexadecimal value. As in most cases, given the output of the sha256 algorithm it is not easy to determine the input. In PHP we can get the sha256 hash of our password like this:

regsiter.php (part 2)

|  |
| --- |
| $hash = hash('sha256', $pass1); |

**Pass the Salt**

As I mentioned before, there is no simple way to determine the input of the sha256 algorithm from the output. It is possible however through brute force, provided the input string (ie the password) is short enough. One way to improve the security of your users’ passwords is to use a salt. A salt is just a random string of characters that is appended to the hash, which is then hashed again.

regsiter.php (part 3)

|  |
| --- |
| //creates a 3 character sequence  function createSalt()  {      $string = md5(uniqid(rand(), true));      return substr($string, 0, 3);  }  $salt = createSalt();  $hash = hash('sha256', $salt . $hash); |

Some people will tell you that a salt is not necessary, but it certainly doesn’t hurt to use one.

Now comes the database portion of our code, for this I will assume we are using a mysql database. I will also assume the database host, name, user, and password. I won’t anticipate any database connection errors in the code, but you should.

register.php (part 4):

|  |
| --- |
| $dbhost = 'localhost';  $dbname = 'tinsology';  $dbuser = 'tinsley';  $dbpass = 'myrealpassword'; //not really  $conn = mysql\_connect($dbhost, $dbuser, $dbpass);  mysql\_select\_db($dbname, $conn);  //sanitize username  $username = mysql\_real\_escape\_string($username);  $query = "INSERT INTO users ( username, password, salt )          VALUES ( '$username' , '$hash' , '$salt' );";  mysql\_query($query);  mysql\_close();  header('Location: login\_form.php'); |

In the above code we establish a connection to our MySQL database and add our new user to the users table. Then we redirect to our login form. Notice that we call the funciton mysql\_real\_escape string. This function helps to prevent SQL injections by escaping the input. Using an abstraction layer like PDO will also help to prevent this. Now that we’ve processed our registration data we can write the code to process our login data.

**Logging in**

Seriously this time. Our login processor will pull the login data from post and compare it to the database values.

|  |
| --- |
| $username = $\_POST['username'];  $password = $\_POST['password'];  //connect to the database here  $username = mysql\_real\_escape\_string($username);  $query = "SELECT password, salt          FROM users          WHERE username = '$username';";  $result = mysql\_query($query);  if(mysql\_num\_rows($result) < 1) //no such user exists  {      header('Location: login\_form.php');  }  $userData = mysql\_fetch\_array($result, MYSQL\_ASSOC);  $hash = hash('sha256', $userData['salt'] . hash('sha256', $password) );  if($hash != $userData['password']) //incorrect password  {      header('Location: login\_form.php');  }  //login successful |

As I mentioned before, this is a stripped down example. Ideally you would not have the code you use to connect to your database in each file, but rather in a single file or function that you could include. In addition to this you should maintain a session throughout the process in order to store and report error message, as well as other useful data. The most important thing to remember when creating a login system is that you should never trust your users. Validate all user input, protect against SQL injections, and never store raw passwords in the database.

**P.S. Session Control**  
*Added December 15th 2009*

All of the above code illustrates how to register a user, and allow them to login. There is, however, one fundamental piece that is missing: session control. In order for a login system to be useful, it must provide some means to distinguish between a logged in user and a non-logged in user across the entire site. Sessions are the means by which we do this. What we want to do is, after a user has successfully logged in is indicate, using a session variable, that the user has done so.

**Accessing Session Data**

Activating and managing sessions in PHP is very straightforward, you only need one function to both create and recall a particular session: session\_start(). Here is a generic example of how to use session\_start to store session data:

page1.php

|  |
| --- |
| session\_start();  $\_SESSION['foo'] = 'bar'; |

page2.php

|  |
| --- |
| session\_start();  echo $\_SESSION['foo']; //will output bar |

In the above example you can see that in page1 we start a session and assign the session variable “foo” the value “bar”. If the user visits page2 sometime there after the session will be recalled and the value of foo (bar) will be echo’d. Notice that session\_start creates a new session if one does not exist or recalls that session if it already exists.

There are a few subtleties relating to session\_start that are important to remember. You must call session\_start before any headers are sent to the browser. This means that your script cannot have any output or calls to header() (or any other function that sends headers) before calling session\_start. The simplest way to avoid this problem is to call session\_start before anything else. One common case where this problem can occur is when some code that normally wouldn’t have any output generates an error or warning prior to calling session\_start. The default error handler will automatically output any error messages to the browser.

Generally (though not necessarily), a session lives as long as the browser remains open.

**Using Sessions in Our Login System**

There are three basic functions we want to incorporate into our login system: validating a user (i.e. indicating that user has logged on), checking if a user is logged on, and logging a user out.

*Validating a User*

|  |
| --- |
| function validateUser()  {      session\_regenerate\_id (); //this is a security measure      $\_SESSION['valid'] = 1;      $\_SESSION['userid'] = $userid;  } |

This function simply sets the session variable ‘valid’ to 1. You may also want to use this function to store certain variables. It is a good idea to store frequently accessed data about a particular user (such as a user id, username, NOT a password or sensitive data). The $\_SESSION['userid'] = $userid; line is an example of how to store user info. Additional information about session security will be provided in the following section.

*Checking if a User is Logged On*

|  |
| --- |
| function isLoggedIn()  {      if(isset($\_SESSION['valid']) && $\_SESSION['valid'])          return true;      return false;  } |

This function simply checks if the session variable ‘valid’ is set to 1.

*Logging Out*

|  |
| --- |
| function logout()  {      $\_SESSION = array(); //destroy all of the session variables      session\_destroy();  } |

When it is time to log a user out, we destroy the session. All of these functions assume session\_start has already been called.

Now it is time to incorporate this into our login script:

|  |
| --- |
| session\_start(); //must call session\_start before using any $\_SESSION variables  $username = $\_POST['username'];  $password = $\_POST['password'];  //connect to the database here  $username = mysql\_real\_escape\_string($username);  $query = "SELECT password, salt          FROM users          WHERE username = '$username';";  $result = mysql\_query($query);  if(mysql\_num\_rows($result) < 1) //no such user exists  {      header('Location: login\_form.php');      die();  }  $userData = mysql\_fetch\_array($result, MYSQL\_ASSOC);  $hash = hash('sha256', $userData['salt'] . hash('sha256', $password) );  if($hash != $userData['password']) //incorrect password  {      header('Location: login\_form.php');      die();  }  else  {      validateUser(); //sets the session data for this user  }  //redirect to another page or display "login success" message |

Now we can use the isLoggedIn function to determine if a user is logged in and act accordingly:

membersonly.php

|  |
| --- |
| session\_start();  //if the user has not logged in  if(!isLoggedIn())  {      header('Location: login.php');      die();  }  //page content follows |

**A Note About Session Security**

By default, sessions are cookie based. This means that a particular session is associated with a user by use of a cookie. This being the case, there are certain vulnerabilities that arise. There are a variety of methods by which a session can be hijacked (XSS for example; a javascript injection can cause a user to give up their session id). Unfortunately you cannot expect to eliminate the possibility that a user has hijacked someone else’s session. Further research into the subject may yield a few tricks, but ultimately the best practice is to be cautious about certain tasks. If a user wants to change their password (after logging in), require the old password and whenever possible avoid displaying sensitive data.

Earlier I mentioned that you shouldn’t use session variables to store sensitive data. It is important to remember that session data lives on the server, this means that a user cannot directly view or modify session data. On a shared server, however, other users of that server may be able to access this information.

**A Note About Hashing Algorithms**  
*Added December 15th 2010*

Lately I’ve been seeing a lot of articles talking about password cracking. I wrote about one[here](http://tinsology.net/2010/08/do-we-need-longer-passwords/). Recently I responded to a question regarding a similar article ([link](http://webmaster-talk.com/php-forum/218125-sha-1-cracked-with-cloud-computing.html)). It is important to know that when someone says that they can crack as hashed password (which is to say they can determine the input of the hash function given only the output) they might mean one of a few things. They might mean that they can perform a preimage attack, which means that given any arbitrary hash they can find a corresponding input string in a certain amount of time. They might also mean that they can perform a brute force attack, which means that given a hash they can find a corresponding input string by generating a hash for every possible string up to a certain length.

The key difference here is that one attack depends on a vulnerability in the hash algorithm while the other depends on a vulnerability in the password itself. My advice is this:

1. Know if you’re hashing algorithm is collision resistant. If you’re using md5 you haven’t done your homework. If you’re using sha1 you’re probably okay, but it may be time to consider something better.
2. Do not disclose how you’re generating your hashes. If an attacker cannot use a preimage attack and does not know how to generate the hashes to perform a brute force attack then there isn’t a whole lot else they can do.

**Something Important to Remember**

Just a reminder, all of the code examples I provided are exactly that: examples. They are meant to serve as a starting point, and hopefully shed some light on a few key concepts, but they are not a real world implementation. I don’t recommend, for example, using header() to bounce your users around to different pages; there are better methods that reduce the number of page loads and give the user a more fluid experience. I use this method here because it simplifies things and makes the examples easier to follow.

1. Adam on [**February 15, 2012 at 8:25 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9530) said:

Thank you Tinsley for publishing this. It’s been difficult to find an easy to read intro to a basic login system. This is great. Best wishes!!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9530#respond)

1. Jo on [**February 10, 2012 at 11:44 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9509) said:

First of all I wanna thank you for this post.  
I find it very usefus and clear, but I’ve a little security related question: hashing the password in the db make it really undiscoverable in any way. This is a real good thing. But what happens to the form data (username and password) between client and server?! I mean: unless you don’t use an https connection this data travels in plain text and can be easily sniffed.  
So my question is: there’s a way to encrypt data BEFORE send’em? (maybe with javascript?!)

After my question I want to say to all those who criticize this post that I think these resorces need to be contextualized in your knowledge. If you don’t have the right PHP or programmin’ basis you really don’t understand a word of what you can read here!!! I say this because I’m not a programmer, only an enthusiast with a little PHP and programming basis. Anyway I’ve realized what I’ve read in this post in about 5 minutes… I apologize for my honesty but I think that first of all u’ve to say “Thank’s for sharing!” and then you can give your constructive contribution.  
Cheers!!!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9509#respond)

* + [**Tinsley**](http://tinsology.net/)on [**February 10, 2012 at 9:00 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9512) said:

*But what happens to the form data (username and password) between client and server?! I mean: unless you don’t use an https connection this data travels in plain text and can be easily sniffed. So my question is: there’s a way to encrypt data BEFORE send’em? (maybe with javascript?!)*

The data is sent unencrypted so yes someone listening in could retrieve your raw password. The solution is to use ssl.

If you needed to you could use javascript to encrypt the form data and decrypt it server side. You would need to use some kind of public key encryption (ex RSA). Basically you would use the public key to encrypt all of the form data and then server side you would use the private key to decrypt it. Someone intercepting the request wouldn’t be able to decrypt the form data without the private key. You would also need to use a nonce to prevent replay attacks (which ssl also protects against).

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9512#respond)

* + - Joon [**February 11, 2012 at 2:59 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9514) said:

Thank you for ur reply!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9514#respond)

1. Steve on [**February 10, 2012 at 6:02 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9503) said:

This would be a great tutorial if it were more complete, but for someone new into PHP, it’s absolutely useless. It’s like bits of code splashed around on a page, no direction nor bearing. I think people would appreciate a direct link to a ZIP instead of this. Stingey coders…

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9503#respond)

* + [**Tinsley**](http://tinsology.net/)on [**February 10, 2012 at 8:56 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9505) said:

The goal isn’t to teach you PHP. The zip file isn’t happening. Stingy coders is one way of looking at it; another is lazy readers.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9505#respond)

* + - Jagedmanon [**February 10, 2012 at 11:25 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9508) said:

I thoroughly disagree with, Steve. If you read the entire post, it explains everything. You will never be a good coder by downloading scripts and copying and pasting. Do the work!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9508#respond)

* + - Derekon [**February 10, 2012 at 9:48 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9513) said:

100% agree I started reading this to better secure my custom made login system.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9513#respond)

1. Chris on [**February 5, 2012 at 2:13 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9473) said:

A great starting point. Thanks a bunch!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9473#respond)

1. Ollie on [**February 4, 2012 at 9:17 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9472) said:

This evening I’ve been researching secure login and registration practices and this one is exactly what I’ve been looking for — great well written article. Thanks

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9472#respond)

1. qamar on [**January 28, 2012 at 7:31 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9431) said:

Nice tutorial, You have a great way of making things simple.  
cheers

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9431#respond)

1. David H on [**January 27, 2012 at 11:25 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9429) said:

Terrific article. I was able to get this going very quickly.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9429#respond)

1. Iosif Miclaus on [**January 27, 2012 at 10:57 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9424) said:

Hello there :)

I really like this article and will surely help me with some security issues i currently have for my project.

I still have a “small” question, that you might be able to answer:  
What if I need multiple logins at the same time? Can i still use the $\_SESSION variable or should I use $\_COOKIE insead?

I hope you can help me :)

Sincerely,  
I.M.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9424#respond)

* + [**Tinsley**](http://tinsology.net/)on [**January 27, 2012 at 11:27 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9425) said:

Don’t use cookies. Cookies are stored client side and are sent to the server with each request. Session data is stored on the server. Out of the box PHP uses a cookie to identify each user with their particular session, but this does not allow session data to be manipulated; storing all of the data in cookies would.

What do you mean by multiple logins? Do you mean having multiple users logged into your site at once? If so the answer is yes, you can use sessions.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9425#respond)

1. [**Dennis**](http://divtastik.com/) on [**October 18, 2011 at 7:40 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9008) said:

This article is the best so far, Thinsley, Please keep up and post or upload more tutorials, you giveaway very useful information.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9008#respond)

1. derp on [**August 18, 2011 at 6:25 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8654) said:

really nice tutorial, was very helpful! Could you be a liitle more specific on this: ” I don’t recommend, for example, using header() to bounce your users around to different pages”

What methods are recommended?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8654#respond)

* + [**Tinsley**](http://tinsology.net/)on [**August 18, 2011 at 1:34 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8655) said:

What I meant by that is that you shouldn’t be redirecting users from page to page if you don’t have to. The php file that generates your login form should be the same one that processes the data. This way if there is an error (ie incorrect password) you don’t have to redirect the user back to the login.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8655#respond)

1. Johnson on [**August 6, 2011 at 9:22 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8573) said:

Thank you for the tutorial!

I had my login system already but I had doubts if it was secure or not. This helped me make the system a little more secure.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8573#respond)

1. nat on [**July 28, 2011 at 3:44 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8527) said:

nice but am looking for a loin script that need: email + password only no username.

If you find one on the net please email me.  
thanks.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8527#respond)

* + [**Tinsley**](http://tinsology.net/)on [**July 28, 2011 at 4:06 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8529) said:

What’s to stop you from using an email as a username?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8529#respond)

1. [**leif**](http://www.nothing.org/) on [**July 28, 2011 at 2:30 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8526) said:

One thing I learned when I built a registration system. People get confused by the difference between registering as new or just logging in. For the application I built it was just better to give the user one form, if they were already registered then they were considered logged-in, if they were new, then they were treated as a new user. What security issues would you see in simply providing a single form with the double password for both new and registered?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8526#respond)

* + [**Tinsley**](http://tinsology.net/)on [**July 28, 2011 at 4:05 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8528) said:

I’m not too sure that using a single form is the solution to people being confused; maybe have a more prominent registration link? Wouldn’t using a single form mean asking users to input their password twice every time they log in? Also, how would you distinguish between someone trying to register with a taken user name and that user logging in?

As far as security goes, I don’t think there is anything inherently insecure about it, but I do think it’s somewhat cumbersome to deal with (both for the user and the person implementing it).

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8528#respond)

1. Jesse S. on [**July 14, 2011 at 6:24 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8366) said:

This was great! I’m a novice and was still able to set this up on my site. I just have one question:

What happens if my users forget their password? If the actual password is not stored in the database, then how could I ever retrieve it?

Thanks!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8366#respond)

* + [**Tinsley**](http://tinsology.net/)on [**July 14, 2011 at 6:30 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8367) said:

You can’t retrieve a hashed password. You can reset the password. Just generate a random string, hash it, store that hash in place of the old hash, and email the new password to the user.

Any site that allows you to retrieve your password is storing raw passwords, which is a bad idea. If anyone were to gain access to the user database they would have all of the users’ passwords.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8367#respond)

* + - Jesse S.on [**July 15, 2011 at 8:16 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8381) said:

Gotcha! Thanks again!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8381#respond)

1. Harry on [**June 22, 2011 at 1:43 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8050) said:

Hi there! Thanks for the tutorial! However, I’m trying to fix the logout() bug which i’m encountering right now. After trying to log out, i get this error: Undefined index: valid in … I believe this is happening because valid is now null? after logging out, is there a work around or something simple that i might be missing to fix this issue? thanks!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8050#respond)

* + Harryon [**June 22, 2011 at 1:44 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8051) said:

i forgot to add, this bug shows both the !isLoggedin content and the content that should be showing after logging in.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8051#respond)

* + Harryon [**June 22, 2011 at 1:51 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8052) said:

actually, if anyone is having trouble with this, try adding isset to valid in the function isloggedin! hope this helps! sorry for adding so many comments ;D

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8052#respond)

* + [**Tinsley**](http://tinsology.net/)on [**June 22, 2011 at 9:55 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8058) said:

That’s not an error, it’s a notice. A script can still function as intended while still generating notices. That being said the following will not generate a notice:

|  |
| --- |
| function isLoggedIn()  {      if(isset($\_SESSION['valid']) && $\_SESSION['valid'])          return true;      return false;  } |

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8058#respond)

1. Archelutux on [**June 8, 2011 at 7:08 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7959) said:

Hello. Can you explain how you make your links have no page title in them? I’d like to my site do that as well so when I pass variables with ‘get’, people can’t see them. I used something in htaccess once on a wiki but it didn’t seem to work.

I’ve looked for info on this, but I think I’m searching for the wrong key words. So, if someone types mywebsite.com/Foo , how do I get it to treat the url like mywebsite.com/Foo/index.php or mywebsite.com/Foo.php or mywebsite.com/index.php?pagename=Foo – any of those while just showing the /Foo version in the url bar and allowing bookmarks to be written as just /Foo ?

I am posting this here because the login system stuff is leading me to that challenge. Thanks very much!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7959#respond)

* + [**Tinsley**](http://tinsology.net/)on [**June 8, 2011 at 7:23 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7960) said:

htaccess is the way to go. I haven’t tested any of these:

For website.com/foo -> website.com/foo.php

|  |
| --- |
| RewriteEngine on  RewriteRule (.\*) $1.php [NC] |

For website.com/foo -> website.com/index.php?pagename=foo

|  |
| --- |
| RewriteEngine on  RewriteRule (.\*) index.php?pagename=$1 [NC] |

For website.com/foo -> website.com/foo/index.php

|  |
| --- |
| RewriteEngine on  RewriteRule (.\*) /$1/index.php [NC] |

I recommend you read this:  
<http://corz.org/serv/tricks/htaccess2.php>

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7960#respond)

1. Romany on [**June 2, 2011 at 2:05 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7927) said:

Hello. I am finding this article helpful, but what I don’t understand is the header(location: login (or register)\_form.php). Are we meant to replace these links with other links? And if we don’t, what are we meant to put in those files? This has been skipped…

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7927#respond)

* + [**Tinsley**](http://tinsology.net/)on [**June 2, 2011 at 9:35 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7930) said:

It depends on how you implement your user management system. For example if a user inputs incorrect login information you should direct them back to the login form. Where ever that may be. Note that by ‘direct them’ I don’t necessarily mean use header to bounce them to another page. It is possible (and practical) to have your login form and your form processing components in the same script. For example:

|  |
| --- |
| if($\_SERVER['REQUEST\_METHOD'] == 'POST')  {       /\*       \* process login information.       \* if correct redirect to whatever landing page you are using for users       \* if incorrect set and error message and display the login form       \*/  }  //Display the login form here |

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7930#respond)

* + - Romanyon [**June 2, 2011 at 6:37 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7936) said:

Ah, I get it now :) Thankyou! And quick reply too!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7936#respond)

* + - Paulon [**June 20, 2011 at 1:25 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8039) said:

Hi, I noticed you used if($\_SERVER['REQUEST\_METHOD'] == ‘POST’){}  
I have in many cases used if($\_POST){} or if(isset($\_POST["variablename"])) to check if a (method=”POST”) form was submitted.  
Is your method better/safer and why in that case?  
Does it prevent people from creating their own form (on their own web page) and submit a form like that? That’s something I have worried a bit about.  
Been trying to find a good answer online but without success.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8039#respond)

* + - * [**Tinsley**](http://tinsology.net/)on [**June 20, 2011 at 1:43 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8040) said:

All three methods will work, but there are some subtle differences.

if($\_POST) works because an empty array casts to false (note that $\_POST is always set even if the request method is not post). if(isset($\_POST["variablename"])) will work so long as that variable is set. Note that it isn’t actually checking if the request is a post request, only if that particular field is set.

I wouldn’t say any of the are ‘the best’. I tend to use if($\_SERVER['REQUEST\_METHOD'] == ‘POST’), because that’s the one I tend to use.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8040#respond)

1. Eli Vegh on [**May 15, 2011 at 5:48 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7797) said:

Could you just show the finished product, please?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7797#respond)

* + [**Tinsley**](http://tinsology.net/)on [**May 15, 2011 at 6:40 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7798) said:

No.

You can find my reasoning in several of the previous comments.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7798#respond)

* + - mordofon [**October 24, 2011 at 3:22 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9038) said:

Nice :) Glad you say no to this, heh. Prevents people from just copy+pasting and having no idea what they’re doing ^-^.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9038#respond)

1. Kathie on [**April 6, 2011 at 7:54 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-6139) said:

I’m trying to follow along this tutorial. The only problem is I don’t know where I’m putting some information. I think thats why people are asking for a zip of this file. :(

I’m an idiot.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=6139#respond)

* + [**Tinsley**](http://tinsology.net/)on [**April 6, 2011 at 8:17 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-6140) said:

I still stand by my decision not to include a zip of the code I posted. When you develop your own user management system, your code is going to have to integrate with the rest of your application. Obviously it is impossible for me to write code that takes that into account, so simply dropping my code into your site is either not going to work correctly, or is going to be difficult to maintain. In other words, you can’t develop an application component in a bubble. You need to take into account how it will interact with the rest of your code.

If you can point out more specifically what you don’t understand I can try to explain it more precisely.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=6140#respond)

1. William on [**March 2, 2011 at 11:19 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5753) said:

There’s one thing I don’t understand: in which file (and where) do I put the functions for the sessions to work? Someone asked for a zip before and I really understood the answer about having to update the zip, but could someone post a zipfile none the less, as the tutorial itself isn’t enough for me to fully understand what goes where…

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5753#respond)

* + [**Tinsley**](http://tinsology.net/)on [**March 3, 2011 at 6:04 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5764) said:

You can put the function definitions anywhere so long as they are included before they are called.

The main reason I won’t upload a zip file is not because it would be tedious to update it every time I modify this tutorial (though that is a contributing factor). The main reason is that I don’t want people to just copy and paste this code into an application. My goal is to demonstrate the concepts.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5764#respond)

1. mimmi on [**February 1, 2011 at 2:36 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5516) said:

ignore my previous comments please =D there was a problem with the registration php Part 2 that you had written! Instead of this:

|  |
| --- |
| $hash = hash('sha256', $pass); |

it should be:

|  |
| --- |
| $hash = hash('sha256', $pass1); |

because pass isnt a variable at all! pass1 or pass2 are the input!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5516#respond)

* + timon [**May 3, 2011 at 12:33 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-7388) said:

Thanks mimmi! I was having the same problem! This fixed it!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=7388#respond)

1. mimmi on [**February 1, 2011 at 1:55 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5515) said:

Okay, i did an echo of the $userData['salt'] thing and it seems its a completely new salt =\

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5515#respond)

1. mimmi on [**January 31, 2011 at 11:10 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5514) said:

I’m trying to incorporate this login system but login always fails. it takes me to the login failed page. I thought i did something wrong but then i made a page to show the Hash generated from the login page and one that shows the one stored in the database. somehow they dont match. i think it may have to do with the salt so what do you think?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5514#respond)

1. inteblio on [**January 26, 2011 at 2:14 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5447) said:

Great article, concise, well written.  
I love that you respond to comments.

It seems to me worth pointing out – (your script does not suffer from this) – with the most amazing log-on page, if you then on the rest of the site, just check the cookie “logged\_in=yes” … then all an attacker needs to do is try a bunch of cookies until they find “logged\_in=yes”.

To counter this, you say in your database “user is logged on for 15 minutes after x time”. Then you reduce the window of opportunity.

You could also (which you have) recorded the IP of the user who logs in, but then this would have to be stored in a cookie too. As you say, this can be read, in transit, over http (but not https). (id be interested to know more about that), but spoofing an IP doesn’t get you very far because all the responses are sent THERE and not back to the hacker with the spoofed IP [but you could potentially get the script do 'do things' as long as you don't need an answer].

My point is, that without a log-out, i think that a hacker could simply modify their cookies until they found something which worked (i’ve seen ‘secure’ tutorials with this weakness)

another of my points is, that without being able to store cookies on the clients computer, the only way you can work a log-in system is to put the username and password into every link on the page (tailored to each signed in user), and to run the whole thing is HTTPS ?? is this correct?? or rely on the IP of the user – but this could be the same for hundreds of users from large buildings, which is not acceptably secure?? without cookies, or HTTPS, is it possible to securely log-somebody in?

I might also put a time-delay on any password page, to reduce the trys/per time of a brute-force attack. Possibly it’s worth saying “this IP can only contact this page 10 times, then i refuse it”. ??

I don’t know what i’m talking about by the way.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5447#respond)

1. Daniel on [**January 23, 2011 at 9:44 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5420) said:

Awesome thanks. I’m planning to script a little php based game in the future whenever I get back into learning PHP all over again, so this will definitely help me to not only create a secure login for my game, but can possibly help me secure existing login systems to my websites.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5420#respond)

1. Will on [**January 21, 2011 at 1:17 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5391) said:

Very nice overview. I realise that you didn’t intend this to be copy-and-pasted into production, but is there any reason why you chose a 3-character length for the salt? You’d only need 16^3=4096 rainbow tables for the salt to be overcome, which isn’t inconceivable at all. It’s so fast that you might as well set it to 64 characters or so, maybe by combining a couple of MD5s. And you could make up some of the lost time by changing the password schema from VARCHAR(64) to CHAR(64) :P

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5391#respond)

* + [**Tinsley**](http://tinsology.net/)on [**January 21, 2011 at 9:12 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5398) said:

That would be the case if we were just appending the salt to the password like this:

|  |
| --- |
| $hash = hash('sha256', $password . $salt); |

Since we are rehashing like this:

|  |
| --- |
| $hash = hash('sha256', $userData['salt'] . hash('sha256', $password) ); |

it isn’t an issue.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5398#respond)

* + - Willon [**January 21, 2011 at 1:40 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5404) said:

An attacker would just need to know your very basic algorithm for rehashing then he can set up his tables as such.

My point, if indeed I have one, is that the computational time for SHA-2 hashes is so low that you might as well just iterate a few more times. Even computing 500 or so hashes per pass wouldn’t be a problem – it takes a matter of milliseconds even on low-end servers. But taking 500 times longer to produce the tables really is a problem.

It’s also good practice to include the salt in every rehash, rather than just making a straight hash of $password for the first hash. This reduces the number of collisions quite significantly.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5404#respond)

* + - * [**Tinsley**](http://tinsology.net/)on [**January 23, 2011 at 6:56 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5424) said:

Okay I see what you’re saying now. When you mentioned rainbow tables my mind immediately jumped to general purpose rainbow tables, which wouldn’t really be of any use here.

You raise a very valid point. There are better approaches that yield better brute force resistance. I have been meaning to rewrite the hashing portion of the article to use[crypt](http://us2.php.net/manual/en/function.crypt.php). This allows you to specify the number of rounds in the algo.

|  |
| --- |
| crypt('laskdjfl;j', '$5$rounds=5000$asldkjflj$'); |

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5424#respond)

* + - * + johnhon [**June 13, 2011 at 8:53 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8001) said:

first, thanks to tinsley and everyone else who commented.

tins, any update on including crypt() into the article? i’d like to use it. to see how you do it would be a huge help.

john h

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8001#respond)

1. Martin on [**January 19, 2011 at 5:35 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5363) said:

Hello and thanks for a great article.

I have created a login system where I have used several ideas from your good blog, so thanks :) Now I also need to making a ‘remember me’ feature. Thus making a function which store a cookie, which then will login the user.

I’ve done this by storing a cookie, which keeps the user ID and “hash(‘sha256′, $saltTemp . hash(‘sha256′, password))”, where the $saltTemp value is not the same salt value as used when the user manually login. This $saltTemp is replaced each time the cookie is used to login, so that a cookie only can be used once (to login).

Now I am afraid that it becomes too easy to find the original password if some can steal enough cookies.  
So my question is, are there any problems with this model? it is a secure model?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5363#respond)

* + [**Tinsley**](http://tinsology.net/)on [**January 19, 2011 at 3:31 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-5371) said:

Given the output of hash('sha256', $saltTemp . hash('sha256', password)) there is no way for an attacker to derive the password other than brute force, which would mean they would need to know the value of saltTemp. One thing you may want to do is also include the user’s IP address in the hash:

|  |
| --- |
| hash('sha256', $\_SERVER['REMOTE\_ADDR'] . $saltTemp . hash('sha256', password)) |

Even if someone intercepted a cookie they would have to have the same IP address in order for the hash generated by the server to match.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=5371#respond)

1. Dharm on [**December 4, 2010 at 4:28 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3974) said:

hii… i have a question… i have secure my site registration at three levels… 1. email authentication – mean our registration form send email to user email address for authentication. user have to follow the link we send into this email. 2. after that user follow this link through that email and reaches to our registration page. where he gets a complete form to fill up for the registration. Now on this page i am using CAPTCHA code to know the user is a machine or human. 3. I also put a readonly user IP address for the completion of registration.

Now the problem i am facing is by one hacker… he is registering on my site since two months and his ip is hidden whenever he register on my site. I am not getting his ip address but yes i am getting the CAPTCHA code he write at the time of registration. “I am also storing the CAPTCHA code to my database”. As in the beginning i found that his IP address is hidden I then put a command for compulsary IP address at the time of registration. The IP address input box is readonly and it takes IP address of a visitor automatically. But still he is manage and overridding my registration form and able to register on my site without showing his IP address.

Do you have any idea how he is doing this? And what should i do to block him? Any Suggestions?

I am still working on it… reading articles … also read your articles… i am not a programmer and all above i mentioned are my few efforts…. but still not succeedeed to block this hacker…. ya the hacker is from china as the country he chooses is China.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3974#respond)

* + [**Tinsley**](http://tinsology.net/)on [**December 4, 2010 at 5:50 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3975) said:

Any form data, regardless whether it is read only, hidden, or restricted to discrete values (ie a select, checkbox, or radio field) can be modified by users. You need to validate that data server side.

If I’m understanding correctly, there is no need for you to even include an IP field in the form. You can get the user’s IP server side from the $\_SERVER array:

|  |
| --- |
| $ip = $\_SERVER['REMOTE\_ADDR']; |

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3975#respond)

1. Ed Z on [**December 4, 2010 at 10:08 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3970) said:

Great tutorial, thanks for putting this together! I am totally new to all this and you clearly answered a number of questions I had (eg preventing brute-forcing the hash by using the salt etc…) One slightly related question: at the end you say “I don’t recommend, for example, using header() to bounce your users around to different pages; there are better methods that reduce the number of page loads and give the user a more fluid experience”

I was curious – what are the other methods for bouncing the user around (since I have just been using header(), and would love to hear a better way)

Thanks!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3970#respond)

* + [**Tinsley**](http://tinsology.net/)on [**December 4, 2010 at 1:05 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3972) said:

In the example, when a user registers the script that processes the information is called register.php, while the script that contains the form is called register\_form.php. If the user fails to input a certain field then they are bounced back to register\_form.php. This is a waste of bandwidth and time. register.php and register\_form.php should both be the same script. For example

|  |
| --- |
| if($\_SERVER['REQUEST\_METHOD'] == 'POST')  {       //handle registration processing  }  //display registration form |

There are certain cases where it makes sense to use header to redirect a user. For example if someone attempts to access a restricted part of a site you would want to redirect them to a login page.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3972#respond)

1. Meh on [**November 10, 2010 at 7:59 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3787) said:

Tell me whats more secure in adding a salt with it? You are still sending the password unencrypted over the line. Whats more secure in sending the password to the server and hash it with known salt then just hashing it with sha1 or md5?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3787#respond)

* + [**Tinsley**](http://tinsology.net/)on [**November 10, 2010 at 8:51 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3789) said:

Salting prevents the use of rainbow tables. A rainbow table is just a table containing strings and their hash values. For instance if you had a rainbow table and the sha1 hash of a string and you wanted to find the value of that string you could just look up the hash in the table and find a corresponding string value.

In addition to this it makes it nearly impossible to brute force the hash unless you have the salt.

By the way, md5 is not intended to be used as a password hashing algorithm. It is very easy to generate collisions for md5 hashes.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3789#respond)

1. Christine on [**October 24, 2010 at 6:35 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3737) said:

I have been reading about and implementing pieces of: cookies, sessions, storing user login info in databases, salt, encyption algorithms, SSL certificates, data integrity policies, good/valid/invalid data, data filtering, and all the attacks (XSS, SQL injection) and how to mitigate them, but was wondering how to hook it all together. This example is perfect! Thanks for taking the time to put this together.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3737#respond)

1. ironandsteel on [**August 4, 2010 at 6:08 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3474) said:

Thanks for the excellent article- there are some subtleties I didn’t know about like session\_regenerate\_id.  
NOW- I need to implement remember me (auto login) the “right” way! That would be a nice tutorial. Thanks!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3474#respond)

1. Darrell Gowland on [**August 4, 2010 at 10:19 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3471) said:

Great Article… The only question that I have is would you put a username and password to the database into a file.

Would people be able to create there own users if they used a site copier.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3471#respond)

* + [**Tinsley**](http://tinsology.net/)on [**August 4, 2010 at 10:45 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3472) said:

I’m not quite sure what you are trying to say.

Are you asking if you can use a flat file to store user info? The answer is yes but I wouldn’t recommend it.

I’m also not sure what you mean when you say would you be able to create users with a site copier. I’m not sure what a “site copier” has to do with user management.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3472#respond)

* + - [**Rex**](http://www.masterrex.com/)on [**August 23, 2010 at 11:26 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3547) said:

I’m fairly certain the gentlemen is assuming that someone using some script to ‘wget’ or ‘cURL’ each page in a directory would attain the ability to connect to your database via perusing the PHP code {for database info}; as you well know, he is mistaken. Unless there is some way to directly ‘download’ a PHP/CGI/JS document without first parsing it outside of direct FTP/SSH access that I’m not aware of, that is…

Also, thanks for the excellent tutorial!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3547#respond)

1. RJP1 on [**July 11, 2010 at 3:12 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3416) said:

Hi, great article! It’ll help me start to create an admin login system for a CMS I’m building from a variety of tutorials.

Anyway, here’s my question…

Using an OOP approach as you have above, what would the best way of creating a OOP templated admin area be? Could you advise? I envisage an admin login page which opens out into an admin cpanel with links to different aspects of my CMS.

Any help would be awesome! Cheers, RJP1

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3416#respond)

* + [**Tinsley**](http://tinsology.net/)on [**July 18, 2010 at 12:05 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3439) said:

Using OOP doesn’t change much with regard to the overall approach of a login system. In fact, in my opinion, there is no need to incorporate much of the code I posted into objects, with a few exceptions. Since PHP gives you a choice when it comes to using objects (unlike Java) I recommend you exercise your ability to make that choice and use it only when you feel it is appropriate or necessary.

For instance, it is likely that you will have some kind of database class that will handle your database operations throughout your application. You may also have a user object that contains all of the information pertaining to a particular user. In this case, when a user logs in, you use the database object to run your queries and you create a user object that can then be stored in a session once the user is logged in. The bulk of the code will remain unchanged.

I personally don’t see any advantage in meticulously incorporating all of the code into objects.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3439#respond)

1. Robert Picard on [**May 29, 2010 at 11:57 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3296) said:

This is a great article. Thanks a million.

One thing I noticed though, on line 10 of the first part of “Logging In”, you included a semicolon in the string –  
WHERE username = ‘$username’;”;

Thanks again man, it’s a HUGE help.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3296#respond)

* + [**Tinsley**](http://tinsology.net/)on [**May 29, 2010 at 1:03 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3297) said:

SQL statements end with a semicolon just like lines in PHP code do. Technically it isn’t required unless you are executing multiple statements, but I’m in the habit of always ending statements with a semicolon.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3297#respond)

1. kqr on [**May 29, 2010 at 6:00 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3293) said:

Wow. Thanks. While I’ve figured most of this already, I needed something to launch me into actually implementing it in PHP. Great article.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3293#respond)

1. theiablue on [**May 7, 2010 at 10:56 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3146) said:

Great starting point! Thanks

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3146#respond)

1. Charles on [**April 27, 2010 at 1:33 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-3063) said:

Thanks for this, very useful. I am already using it on a project! :)

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=3063#respond)

1. Scott on [**March 25, 2010 at 8:28 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2771) said:

Forgive my lack of knowledge (still very new to this), but when the user posts the login form is the login information passed to the server in clear text to be processed?

In a standard implementation would these pages then need to be run inside a secure connection, or is there some inherent security that I’m missing?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2771#respond)

* + [**Tinsley**](http://tinsology.net/)on [**March 25, 2010 at 9:36 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2772) said:

Unless you are using ssl/tls (HTTPS) data sent over the network will be unencrypted. In most cases, whether or not you are using https is transparent to PHP; code that works for http connections will work for https as well. In most cases you would only use https if you were dealing with sensitive data: credit card transactions, accessing your bank account, etc. Large sites such as facebook tend to use https as well.

To answer your question, you don’t need a secure connection to implement a login system, but it does add an extra layer of security. For a small site it is probably overkill and considering that a ssl certificate ranges from $100-$400 a year it may not even be feasible.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2772#respond)

1. Peter on [**March 24, 2010 at 11:14 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2767) said:

Quote: “…And the main reason: This is not a working implementation. I don’t mean for you to copy this code and put it into production. It is meant to demonstrate the concepts. It is meant to be a reference for creating a login system of your own.”

People who like me are looking for a login code, are hoping for a usable article, those who can code will do it themselves..

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2767#respond)

* + [**Tinsley**](http://tinsology.net/)on [**March 24, 2010 at 8:07 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2769) said:

If you aren’t willing or able to take code and adapt it to your purposes, rather than copy and paste it out of the box, then why (and how for that matter) are you creating a login system? The article is meant to be a reference and assumes some coding experience. Knowing how to code isn’t the same as knowing how to create a login system (or anything else for that matter).

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2769#respond)

1. verboze on [**March 10, 2010 at 11:39 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2736) said:

Great article! Being new to implementing login systems, this article thought me the basics I needed to get started. Thanks for sharing!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2736#respond)

1. Kevin on [**March 9, 2010 at 2:33 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2730) said:

Your subject for this code is “Creating a Secure Login System the Right Way” yet you post insecure code. WTF? In your own words you say “you should never trust your users. Validate all user input, protect against SQL injections”, yet you don’t provide any sanitizing protection for the password input from the user. Newbies using your “secure” code will now be open to an sql injection attack.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2730#respond)

* + [**Tinsley**](http://tinsology.net/)on [**March 9, 2010 at 2:57 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2731) said:

Did you read the code carefully? The raw password never touches the database… it is hashed. There is no need to use escape\_string or any other sanitizing on hashed data. The ONLY user input that touches the database is the username and that IS sanitized. In addition to this I point out PDO as an alternative to using the mysql\_ family of functions, which is inherently more secure.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2731#respond)

* + - Kevinon [**March 9, 2010 at 3:11 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2732) said:

My bad! Feel free to delete my dumbass post. Thats what **I** get for writing and not sleeping. I take it all back!

I blindly missed the WHERE username = ‘$username’ and assumed it was WHERE username = ‘$username’ AND password = ‘$password’ which WOULD have been a problem.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2732#respond)

1. Nookie on [**March 1, 2010 at 11:42 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2710) said:

Hi,

Could you maybe post or email me entire script in a zip file?  
That would be really really great!

Thanx in advance!  
/Nookie

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2710#respond)

* + [**Tinsley**](http://tinsology.net/)on [**March 2, 2010 at 8:21 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2711) said:

Normally I would, but not in this case for three reasons:

If I post a zip file of this script I will have to update it whenever I updated this post  
All I would be doing is copying the code I posted here into a few files  
…And the main reason: This is not a working implementation. I don’t mean for you to copy this code and put it into production. It is meant to demonstrate the concepts. It is meant to be a reference for creating a login system of your own.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2711#respond)

1. Claude on [**January 28, 2010 at 12:50 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2616) said:

Thanks for the very good example. How ever I don’t know if any one experienced a crash with the registration form. I’m using Apache 2.2 with PHP 5 (WAMP configuration). It seems good practice to use mysql\_close($conn); instead of mysql\_close(); – Refer to register.php (part 4)

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2616#respond)

* + [**Tinsley**](http://tinsology.net/)on [**January 28, 2010 at 7:09 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2621) said:

You are the first person to report a ‘crash’ as far as I’m aware. What exactly do you mean by crash? Is there an error message? Also, regarding mysql\_close(), specifying the connection handle is only necessary if multiple connections are open, otherwise the behavior is identical to calling it without any parameters.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2621#respond)

* + - Claudeon [**January 30, 2010 at 3:19 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2626) said:

Hi Tinsley

Sorry for not being specific. I tested it again with mysql\_close(); Ite gave me the following error:

Apache HTTP Server has encountered a problem and needs to close. We are sorry for the inconvenience. (One of those send / don’t send error reports).

The error signature contains the following information:  
szAppName : httpd.exe  
szAppVer : 2.2.14.0  
szModName : php\_mysql.dll  
szModVer : 5.3.0.0  
offset : 00002072

I did debug the code and the error did occurred at the mysql\_close();. I did a google search and found other developers had a problem with this in the past and stated to use the connection handle variable to prevent such an error.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2626#respond)

* + - * [**Tinsley**](http://tinsology.net/)on [**January 30, 2010 at 11:47 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2627) said:

I’ve looked further into the issue and it appears that the problem stems from a bug in PHP 5.3.0 (<http://bugs.php.net/bug.php?id=48754> ); specifying the connection handle is a workaround for that bug. Upgrading to 5.3.1 should resolve the issue and you’ll be able to use mysql\_close() without parameters.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2627#respond)

1. Pingback: [A Supplement | Tinsology](http://tinsology.net/2009/12/a-supplement/)
2. Larry on [**December 3, 2009 at 2:22 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-2266) said:

Great article!! Very concise and easy to read… and of course use. This is exactly what I have been looking for.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=2266#respond)

1. David Wheeler on [**June 30, 2009 at 3:14 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1744) said:

I’m getting a 40-digit value from the sha1 hash. I can make this work by modifying my db to accept it, but is there something I’m doing wrong that it’s not coming out 28 digits?

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1744#respond)

* + David Wheeleron [**June 30, 2009 at 3:16 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1746) said:

Oh, and great article by the way. :-) It was exactly what I was looking for today.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1746#respond)

* + [**Tinsley**](http://tinsology.net/)on [**June 30, 2009 at 3:51 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1747) said:

That was my mistake. Fixed.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1747#respond)

1. [**mary**](http://codingpad.maryspad.com/) on [**June 30, 2009 at 2:19 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1743) said:

Thanks for this, much appreciated! I’m creating a little app with a login tool so this is useful.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1743#respond)

1. nano on [**June 30, 2009 at 12:39 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1740) said:

And to follow up what Rob said, the maxlength attribute only asks the browser prevent the user form inputting a long string. The form data can’t be trusted since it can be cirucmvented.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1740#respond)

* + [**Tinsley**](http://tinsology.net/)on [**June 30, 2009 at 12:43 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1741) said:

Which is exactly what I mentioned later in the article.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1741#respond)

1. [**Rob**](http://www.parksidedev.com/) on [**June 30, 2009 at 12:27 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1737) said:

Great article… One correction though. Specifying a size attribute doesn’t keep the user from typing more text… The attribute you want to use is maxlength.

Rob

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1737#respond)

* + [**Tinsley**](http://tinsology.net/)on [**June 30, 2009 at 12:38 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-1739) said:

My mistake, that’s what I get for writing instead of sleeping.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=1739#respond)

* + - Johnon [**July 30, 2011 at 5:44 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8537) said:

I find this completely useless and a waste of time. You never mentioned where to save stuff and where to close the php tags and so on. I think this was just stupid.

Kind Regards,

John

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8537#respond)

* + - * [**Tinsley**](http://tinsology.net/)on [**July 30, 2011 at 5:57 pm**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8538) said:

I’m sorry you feel that way but when I wrote this my goal wasn’t to teach people programming 101 and how to create a login system. If you can’t figure out where to close the PHP tags you need to learn the basics first.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8538#respond)

* + - * + Karlon [**August 1, 2011 at 10:41 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-8546) said:

I completely disagree with John, this is a great article and id like to thank you for the time and effort you put into writing this. I knew the basics of sessions and logging in with databases etc, but had no idea how to do a salted password. this has helped so greatly. Thank you very much!

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=8546#respond)

Steveon [**February 10, 2012 at 6:05 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9504) said:

I slightly disagree with John, but massively disagree with the both of you. This is terrible ‘help’.

You could have at least added ‘xxxx.php’ above each section to silently tell people which files they should be editing.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9504#respond)

[**Tinsley**](http://tinsology.net/)on [**February 10, 2012 at 9:06 am**](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/comment-page-1/#comment-9507)said:

Depending on your implementation all of the user management code could be in a single file or spread out across several. Having me throw together several distinct file arbitrarily isn’t as helpful to someone integrating user management into their application as understanding the concepts is. I may not be doing the work for you, but that’s probably a good thing considering I can’t possibly know anything about the design of the rest of your application.

[Reply ↓](http://tinsology.net/2009/06/creating-a-secure-login-system-the-right-way/?replytocom=9507#respond)