Assignment: Login and Rock Paper Scissors

In this assignment we will a two page web application to play rock, paper, scissors (http://en.wikipedia.org/wiki/Rock-paper-scissors). We will only allow logged in users to play the game.

Resources

There are several resources you might find useful:

- Recorded lectures, sample code and chapters from the textbook www.wa4e.com will be helpful in understanding the aspects of PHP used in this application.
    - Arrays
    - Functions
    - Forms and POST Data
- Documentation of how salted hashes work from Wikipedia.
- Documentation on how one web page redirects to another in HTTP and PHP.
- Documentation from PHP on how the header() function works.
- NGROK - A tool to create temporary secure tunnels from a local server to the Internet if you need to submit this to an autograder.

You can download sample code for an **incomplete/broken** version of this application from:

http://www.wa4e.com/code/rps.zip
You can play with the broken sample code at:
http://www.wa4e.com/code/rps/

Sample solution

You can explore a sample solution for this problem at:

http://www.wa4e.com/solutions/rps/

Specifications

When you first come to the application (index.php) you are told to go to a login screen.

Requirements for the Login Screen

The **login.php** should be a login screen should present a field for the person's name (name="who") and their password (name="pass"). Your form should have a button labeled "Log In" that submits the form data using method="POST" (i.e. these should not be GET parameters).

The login screen needs to have some error checking on its input data. If either the name or the password field is blank, you should put up a message of the form:

```
User name and password are required
```

If the password is non-blank and incorrect, you should put up a message of the form:

```
Incorrect password
```

If there are errors, you should come back to the login screen (login.php) and show the error with blank input fields (i.e. don't carry over the values for name="who" and name="pass" fields from the previous post).

You are to use a "salted hash" for the password. The "plaintext" of the password is not to be present in your application source code execpt in comments. For this assignment, we will be using the following values for the salt and stored hash:

```
$salt = 'XyZzy12*_';
$stored_hash = '1a52e17fa899cf40fb04cfc42e6352f1';
```

Note that the sample code is using a *different* salted hash so you should change the sample code to use this hash.

The stored_hash is the MD5 of the salt concatenated with the plaintext of php123 - which is the password. This has is computed using the following PHP:

```
$md5 = hash('md5', 'XyZzy12*_php123');
```

In order to check an incoming password you must concatenate te salt plus password together and then run that through the **hash()** function and compare it to the stored_hash.

If the incoming password, properly hashed matches the stored stored_hash value, the user's browser is redirected to the **game.php** page with the user's name as a GET parameter using:

```
header("Location: game.php?name=".urlencode($_POST['who']));
```

Specifications of the Game Playing Screen

In order to protect the game from being played without the user properly logging in, the **game.php** must first check the session to see if the user's name is set and if the user's name is not set in the session the **game.php** must stop immediately using the PHP die() function:

```
die("Name parameter missing");
```

To test, navigate to **game.php** manually without logging in - it should fail with "Name parameter missing".

If the user is logged in, they should be presented with a drop-down menu showing the options Rock, Paper, Scissors, and Test as well as buttons labeled "Play" and "Logout".

If the Logout button is pressed the user should be redirected back to the **index.php** page using:

```
header('Location: index.php');
```

If the user selects, Rock, Paper, or Scissors and presses "Play", the game chooses random computer throw, and scores the game and prints out the result of the game:

```
Your Play=Paper Computer Play=Paper Result=Tie
```

The computation as to whether the user wins, loses, or ties is to be done in a function named **check()** that returns a string telling the user what happenned:

```
// This function takes as its input the computer and human play
// and returns "Tie", "You Lose", "You Win" depending on play
// where "You" is the human being addressed by the computer
function check($computer, $human) {
    ...
        return "Tie";
    ...
        return "You Win";
    ...
        return "You Lose";
    ...
}
```

The "Test" option requires that you write two nested **for** loops that tests all combinations of possible human and computer play combinations:

```
for($c=0;$c<3;$c++) {
    for($h=0;$h<3;$h++) {
        $r = check($c, $h);
        print "Human=$names[$h] Computer=$names[$c] Result=$r\n";
    }
}
```

The **$names** variable contains the strings "Rock", "Paper", and "Scissors" in this example. The output of this should look look as follows:

This will allow you to make sure that your **check()** function properly handles all combinations of the possible plays properly without having to play for a long time as the computer makes random plays.

What To Hand In

For this assignment you will hand in:

1. A screen shot (with URL) of your login.php rejecting an incorrect password
2. A screen shot (with URL) of your game.php correctly showing the 'Name parameter missing' error
3. A screen shot (with URL) of your game.php doing a normal play with the computer making a choice other than Rock
4. A screen shot (with URL) of your game.php doing a correct test run
5. Source code of login.php
6. Source code of game.php

Grading

Please review carefully. The most important aspect of peer-grading is useful comments about what might be wrong and need fixing. You cannot re-submit your assignment unless the instructor allows you to resubmit.

The total number of points for this assignment is 10. You will get up to 5 points from your instructor. You will get up to 3 points from your peers. You will get 1 for each peer assignment you assess. You need to grade a minimum of 2 peer assignments. You can grade up to 5 peer assignments if you like.

Optional Challenges

This section is entirely *optional* and is here in case you want to explore a bit more deeply and test your code skillz.

Here are some possible improvements:

- Instead of using a series of if-elseif-else statements in the **check()** function, try to compute the win/lose aspect of the game with a simple arithmetic computation using remainder operator (%)
- Add some images to the output. Don't replace the required output with images - simply add some images to make it prettier.

Provided by: www.wa4e.com