Jackson State University
Department of Computer Science
CSC 541 Cryptography and Network Security
Fall 2015
Instructor: Dr. Natarajan Meghanathan
Project # 2

Use of CAPTCHA (Image Display and Selection Strategy) to Prevent XSRF Attacks

Due: November 30, 2015: 4 PM (hard deadline; no postponement)
Maximum Points: 100

In this project, you will be presented with an online banking web application. It will be your task to implement a user-acknowledgment system similar to CAPTCHA. If you do not have a copy of XAMPP in your system, download it as mentioned in Step 1. **Images:** Look in the last page of this project description for the type of images to download and use for your CAPTCHA, assigned for each student. You can try downloading from the http://images.google.com/

1. Download XAMPP for your operating system

Visit the XAMPP website (http://www.apachefriends.org/en/xampp.html) and download the latest version of XAMPP installer for your operating system and start the installation process.

2. Install XAMPP for your operating system

Once the download of the installer has completed, follow the installation instructions given. Make sure to install xampp directly under the C:\drive by creating a folder named 'xampp'. It is better NOT to install Apache and MySQL as services during installation.

3. Download the online banking application archive (a Zip file, posted in JSU Blackboard: as part of the Content for the CSC 541 course)

4. Install the online banking application archive

In the online banking application archive file you have downloaded in Step 3, unzip it. You can find two folders, "Content" and "Data." Open the "Data" folder and copy the folder it contains (hometown_bank) to the /mysql/data/ folder within the XAMPP installation folder on your machine. You should now have a folder /mysql/data/hometown_bank/ containing a number of files.

Open the "Content" folder and copy its contents to the HometownBank folder created within the /htdocs/ folder of the XAMPP installation folder on your machine. You should now have a folder /htdocs/HometownBank/ containing a number of files.

5. Start the XAMPP Services

Start the XAMPP Control Panel (file named: *xampp-control*, located in the main XAMPP folder)

Click the “Start” button next to Apache. It will shortly change to a “Stop” button and display a green “Running” bar. Once Apache is running, click the “Start” button next to MySql. When it has started, the “Start” button will change to a “Stop” button and the green “Running” bar will be displayed next to the MySql label.

To test that XAMPP has been installed properly, open a web browser and navigate to http://localhost. If you see the XAMPP web page, Apache has been installed correctly. Also navigate to http://localhost/phpmyadmin to ensure that MySql has been installed correctly.

6. Open the Online Banking Application in Your Browser

Open a web browser and navigate to http://localhost/HometownBank/ . You should see the Hometown Bank homepage. You should notice that the site contains several pages: “Home,” “Login,” “View Account,” and “Transfer Funds.” You will also notice that you cannot view the “View Account” and “Transfer Funds” pages without logging in.

There is one user account already created within the system. To view the information for this account, open a new browser tab or window and navigate to http://localhost/phpmyadmin . This should display a page with a number of databases listed on the left-hand side. Select the “hometown_bank” database in the list on the left-hand side. Once you have selected the “hometown_bank” database, select the “accounts” table from the list on the left-hand side of the page and click on the icon next to it (placing your mouse on the top of the icon will display ‘Browse’). This should take you to a new page where you may browse the contents of the “accounts” table.

Now, select “Insert” at the top of the page. Fill in the fields with login and imaginary account information for yourself. The username should be your school ID number (J#), and you may enter any password you like. The account number (should be 9-digits long) and the **Amount field should be a value equal to 1000 plus the last 4 digits of your J#**. You should also create three additional fictional user accounts. For these, you may enter any information you like. When you are done, there should be five accounts in the database, the one that was originally in the database, the account you created for yourself, and the three additional accounts you created.

<u>Screenshot # 1</u>: Once you have created the accounts, take a screenshot of the updated “accounts” table to include in your report.

Now that you have created your own user account, return to the browser window containing the bank app and log into your account using the information you just inserted into the database. Once you are logged in, you may click on the “View Account” link at the top. You should now be able to view the information you have just entered into the database.

<u>Screenshot # 2</u>: Take a screenshot of the “View Account” page displaying your account information to include in your report.

7. Intro to Cross-Site Request Forgery

Now click on the “Transfer Funds” link. On this page you may transfer funds from the account you are logged into, to another account. You will first be transferring $200.00 from your account to Gregg McDonald’s account. His account number is **1000000001**. You may also find his account number by viewing the contents of the database. To transfer these funds, enter his account number in the "Transfer To:" field and enter "200.00" in the "Amount" field. When you have done this, click the "Submit" button.

<u>Screenshot # 3</u>: Once you have transferred the funds to Gregg McDonald's account, take a screenshot of the Transfer Verification page to include in your report.

Once again, use the "Transfer Funds" page to transfer money. This time, transfer $200.00 from one of the fictional accounts you created to your own account. To do this, you will need to be logged in as the fictional account and transfer to the account number of your own account. When you have completed the funds transfer, you should have the same account balance as you did when you started.

<u>Screenshot # 4</u>: Once you have transferred funds from the fictional account to your own account, take a screenshot of the Transfer Verification page to include in your report.

Notice the URL of the Funds Transfer verification page. It will appear something like:
http://localhost/HometownBank/transfer_action.php?TransferTarget=1000000026&TransferAmount=200.00
where the TransferTarget is the account to which you wished to transfer money and the TransferAmount is the amount you transferred.

Before proceeding, make sure you log back in to your own account.

Suppose you receive an email with the following text:
Click here to claim your new computer!
Press Ctrl and Click on the link above and answer the following questions:
Note: If nothing works by clicking the above link, click on the link below:
http://localhost/HometownBank/transfer_action.php?TransferTarget=1000000001&TransferAmount=200.00

<u>Questions 1 through 6:</u>

1. **Where does the link take you?**
2. **What is the URL of the link?**
3. **What does the link do?**
4. **What is your new account balance?**
5. **Why does the link work?**
6. **Examine the URL of the page that the link directed you to. Modify this URL to transfer $200.00 to one of the fictional accounts you created and navigate to that URL in the browser. Did the amount of $200.00 get transferred from your account to the fictional account?**

<u>Screenshot # 5</u>: Once you have activated the above URL, take a screenshot of your new account balance. You may access this by clicking the "View Account" link at the top of the Hometown Bank page.

You have just been the victim of a cross-site request forgery attack (XSRF). XSRF attacks successfully occur when a website wrongfully believes that a request being made to it is being willfully triggered by the user.

In this case, the bank's website wrongfully trusts that any request to the transfer_action.php page is being willfully executed by you, the authorized user. That is not necessarily true here. While you did willfully click the link, you most likely did not intend to transfer all of your money to someone else's account.

One way to prevent such attacks from the point of view of a web developer is to ensure that whenever any kind of transactions or changes are made to a user's account, the user acknowledges the transaction. When you clicked the link above, if you had been displayed a page that said "Are you sure you wish to transfer $20,000 to Account Number 1000000026?" you probably would have said "No."

8. Implementing a CAPTCHA-type Scheme

Now imagine that you are a web developer. You get a call from Hometown Bank saying that they have discovered an enormous security flaw in their online banking system, and they need you to fix it for them. They tell you about users falling victim to an email scam that transfers money from their accounts.

You know just how to solve the problem. You will implement a scheme like the CAPTCHA technology with which you are probably familiar. The CAPTCHA system is a challenge-response system which displays a series of characters and requires the user to read and enter the characters displayed in a box to verify the user's humanity (and acknowledgement of the transaction).

The system that you will be implementing will be a little simpler, but should work pretty well. The first thing that you will need to do will be to find 10 simple pictures online by searching http://images.google.com . The pictures may be of anything, but each picture should only include one thing, for example:

- A hat
- A car
- A dog

You should avoid pictures with more than one object in them (a picture containing a horse and a truck), or multiple pictures of the same item (2 pictures of cats). You should also avoid obscure objects that members of the general public might not recognize, or might misidentify. You should also try to pick pictures that are roughly square-shaped.

Once you have found your 10 images, save them to the /HometownBank/images/ directory in your XAMPP install. You will now need to rename the images so that the image names are not descriptive of what is in the picture. Randomly mashing on the keyboard should give you a pretty unrecognizable sequence of characters for a file name. Make sure that there are no punctuation marks or spaces in the file name, only letters between A and Z. The filenames should also be around 10-12 characters long. Once you are done you should have renamed your 10 images to things like:

- eajjeagkajeg.jpg
- yturuyozya.png
- behaleghgh.gif

Now that your images have been renamed, you will have to add them to the database. To do this, navigate to http://localhost/phpmyadmin . Once there, click on the "hometown_bank" database on the left side. You will see that there are two tables in this database, "accounts" and "image data". Click on the "image_data" table. You will see that this table contains two fields: "URL" and "Description". You will now need to add the data for your images into this table. To do this, click "Insert" at the top of the page. This will take you to a page where you may insert the information for your images, one image at a time. In the "URL" field, insert the filename of your image. Be sure to include the extension (.jpg, .gif, .ping, etc.) **with the proper case** (yes, it is case-sensitive!!) or your application will not work. In the “Description” field, enter a one-to-two word description of the image (cat, house, donut, etc.). Once you have done this, click the “Go” button, and continue inserting the information until you have inserted the data for all of your images.

9. Implementation of the CAPTCHA Strategy

We will employ the following strategy to implement the user verification with pictures: We will have the “Transfer Funds” page display five random pictures from our database. It will then request that the user select the picture that corresponds to a randomly chosen description of one of the five pictures. The user will then submit the form. If the user selects the picture that matches the randomly chosen description, he or she will be considered to have been authenticated as a willing human user. The good thing about this method is that it is user-friendly. The user only has to click on a button below the appropriate picture.

Important Note: Do not copy and paste from this document to your PHP editor. Type out all the statements; there could be errors due to differences in quotation characters " and “.

Whenever a user wants to transfer funds, he or she must correctly select an image. If the correct image is not selected, or no image is selected at all, the funds transfer will not take place. This will require that we make some changes to the transfer.html page and the transfer_action.php page.

9.1.1 Edit transfer.html

We will edit our transfer.html page (located in the C:\xampp\htdocs\HometownBank folder) first to include some PHP code that will randomly select and display 5 images from our database. The page will also pick a description of one of the five items and ask the user to click on the picture of the item that is described in words. For example, the user will be presented with pictures of a horse, a cat, a dog, a bird, and a snake and asked to click on the picture of the dog.

The first thing that we will need to do will be to rename transfer.html to transfer.php. Once you have renamed the file, open it in a text editor. Scroll down to line 66, which says:
<input type=”submit” value=”Submit” />
We will be placing all of our PHP code before this line, so everything that you insert should come immediately after the </table> tag on line 65 and before the <input> tag that is currently on line 66.

A. Set up PHP and the database connection

To start a PHP code segment, you need to create your beginning and ending PHP tags. The tag to begin a PHP code segment is

<?php

and the tag to end a PHP code segment is

?>

You will now need to insert the code to access the database. Inside your PHP start/end tags, insert the following code:

```
$host = “localhost”;
$user = “root”;
$password = “”;
```

```
$dbname = “hometown_bank”;
$connection = mysql_connect( $host, $user, $password );
$database = mysql_select_db( $dbname, $connection );
```

The first line stores the hostname where the database is located. The second and third lines store the username and password used for access to the database. The fourth line stores the name of the particular database that we will be using. The fifth line creates a connection object on the host with the specified username and password. The sixth line selects the specific database that we want on the created connection.

B. Perform a query on the database and retrieve the results

Now write an SQL query to get the entire contents of the “image_data” table and store it as the variable *$query*. When you have completed this, your code thus far should look like:

```
…
<?php
    $host = “localhost”;
    $user = “root”;
    $password = “”;
    $dbname = “hometown_bank”;
    $connection = mysql_connect( $host, $user, $password );
    $database = mysql_select_db( $dbname, $connection );
    $query = “SELECT * FROM image_data i”;
?>
```

Extend the above PHP code (<?php ………… ?>) further as explained below:

Now you will need to execute the query on the database in order to retrieve the results and store them in a variable. This is done with the line:

```
$result = mysql_query( $query, $connection );
```

C. Create a table to display the images

We will now set up a table to hold our pictures. Regular HTML tags can be used inside PHP code by using the *echo* command to print the HTML to the browser, for example:

```
echo '<br /> <br />';
echo '<table cellpadding="0" cellspacing="0">';

 echo '<tr>';
```

This inserts two blank lines before the table, creates a table, and starts the first table-row (tr). The first row of the table will contain the randomly-selected pictures and the second row of the table will contain the option buttons with which the user can select the picture matching the displayed text.

C.1 Randomly select images for the table

We will know need to create a random number generator object, some arrays to hold our randomly-selected objects, and a counter. This can be done with the code below:

```
srand(time());
$url[] = null;
```

```
$description[] = null;
$total_selected = 0;
```

We will need to construct two loops. The first loop will randomly select 5 objects from the database and the second will insert the pictures into the table.

The first loop will be a while() loop. It will first get a random picture's URL and description from the database. We will then need to check whether or not that picture has already been inserted into the picture array. If it has not, then we will insert it and increase the counter. If it has not, we will simply let the loop continue.

The structure of the first loop will be:

```
while ($total_selected < 5)
{
    get a random picture

    if (the picture has not already been selected)
        insert the picture into the array
        increment the counter
}//end while
```

To get a random picture, we will first generate a random number between 0 and 9 (both inclusive) and assign it to a variable. This is done with the line:

```
$random = (rand()%10);
```

To get the URL associated with that item in our query results, we can use the line:

```
$current_url = mysql_result($result, $random, “URL”);
```

where $result is the name of our results variable, $random is our random row number, and “URL” is the name of a field in the table. We can then do the same thing to get the value of the “Description” field for the same row and save it in a variable named $current_description .

C.2 Check for duplicate selections

Now that we have our current randomly-selected values, we need to make sure that these have not already been selected and included in the arrays. The *in_array* function in PHP allows us to search for a particular value within an array. The line:

```
if (!in_array($current_url, $url) )
{
}//end if
```

checks whether or not the value of $current_url, our current URL value from the database, is contained already within the array $url. If it is not (!), then the commands within the braces ({ })will be executed. To insert our picture into the array, we can simply assign it using a line such as:

```
$url[$total_selected] = $current_url;
```

and do the same to assign the $current_description to the $description array. We will finally need to increment our $total_selected counter by 1.

When this first loop has completed executing, we should have 5 picture URLs and 5 corresponding descriptions.

C.3 Insert the images into the table

In the second loop, we will simply insert our pictures into the table. This can be done with a simple for() command:

```
    for ($i = 0; $i < 5; $i++)
    {
        echo '<td><img src="./images/';
        echo $url[$i];
        echo ' " height="50" width="50" /></td>';
    }//end for
```

We have now built the first row of our table, so now we need to close the first row by printing a </tr> tag and open the second row by printing a <tr> tag.

D. Insert option buttons into the table

We will now construct a third loop that will place the option buttons across the second row of the table. The structure of the for() loop will be the same as that that was used to insert the pictures, but instead of a picture, we will insert an option button at each iteration. The PHP code for inserting an option button is given below:

```
    echo '<td><center><input type="radio" name="ObjectURL" value="';
                echo $url[$i];
                echo ' " /></center></td>';
```

Once the option buttons are in place and after we get out of the for loop, we will need to close the table row by printing a </tr> tag to the browser and close the table by printing a </table> tag to the browser using the "echo" command.

E. Randomly select an image and store its value in the form

The last thing that we will need to do in our transfer.php file will be to randomly pick one of our 5 objects, and ask the user to select that object.

This is done by selecting another random number between 0 and 4 (both inclusive). We will then use the code:

```
$random = (rand()%5);
echo '<input type="hidden" name="ObjectToFind" value=" ';
    echo $description[$random];
    echo ' ">';
```

to insert a hidden field in the form that will contain the name of the object the user was asked to select. This way the only information that will be submitted with our form is the object that the user is supposed to find and the selection the user made. The form has no way of knowing whether the selection the user made was correct or not, and thus, neither has an automated system any way of knowing.

F. Provide user instructions

We should now print out a blank line to the browser using the "echo" command, as well as a message to the user to "Select the picture of the [selected item] and press the Submit button." This is done as follows:

echo '<br />Select the picture of the ';
echo $description[$random];
echo ' and click Submit to complete your transfer.<br /><br />';

When this is done, our transfer.php page will be complete. We may now move on to updating our transfer_action.php page.

G. Update Links to transfer.php

After you have added the PHP code to your transfer.php page, you will need to update the other pages so that they link to the appropriate page (at the beginning of this section, you changed the name from transfer.html to transfer.php). In turn, open each of the following pages in a text editor and change the two references to "transfer.html" to "transfer.php". The line numbers at which these references occur are given below, but you can also search within the document and find them.

• default.html	(lines 30 and 52)
• login.html	(lines 30 and 61)
• transfer.php	(line 44 and the 12th line from the bottom)
• transfer_action.php	(lines 44, the 11th from the bottom, and several other locations)
• view.php	(lines 44 and 172)
• login.php	(lines 32 and 136)

9.1.2 Edit transfer_action.php

Updating the transfer_action.php page requires the addition of much less code than the transfer.php page did. To update the transfer_action.php page, we will simply need to get the values of the item description, and the item URL that the user selected from the form. The correct item URL for the given description (obtained from the database) will be compared with the item URL selected by the user. If the two URLs match, the user will be allowed to process a funds transfer. If the two do not match, the user will not be allowed to transfer funds because no proper selection (or no selection) was made.

A. Retrieve user input

To retrieve the two required values from the form, we will use the $_GET[] method. The syntax of these two lines will be:
$ObjectToFind = $_GET['ObjectToFind'];
$ObjectURL = $_GET['ObjectURL'];
where $ObjectToFind and $ObjectURL are PHP variables and 'ObjectToFind' and 'ObjectURL' are the "name" values of the input elements from the form.

You need to insert the above two lines after line # 66 in the transfer_action.php file, immediately following the "$amount = $_GET['TransferAmount']" line.

One additional consideration here is that if no values are passed to this page for 'ObjectToFind' and 'ObjectURL', PHP will display a notice stating that "**Notice**: Undefined index: ObjectToFind in **transfer_action.php** on line **71"** and also a notice regarding the fact that 'ObjectURL' has an undefined index.

B. Determine if user input has been passed

Now we need to determine whether or not values for the ‘ObjectToFind’ and ‘ObjectURL’ variables have actually been passed to the form. If the transfer_action.php page were accessed from any page other than the appropriate transfer.php page, or directly from a URL as in the example of the XSRF link in section 7, then these values would not be present.

We will insert the code to check these parameters inside the first if statement (currently if (false)). Edit the *if (false)* statement to read:

if ($ObjectToFind == “” || $ObjectURL == “” || $ObjectToFind == null || $ObjectURL==null)

This will determine whether any of our variables is empty or null. If either of them is not specified, we need to display an appropriate message to the user.

Inside the *if( )* block you just edited, you will see the text

//this will be executed if no valid parameters are passed

Here you will need to use the echo command to insert a message to the user warning that there was an error in processing the transfer request and that he or she should return to the transfer page.

C. Construct a query to check the passed parameters

We will now need to construct a query which selects the rows from the image_data table where the URL in the table is equal to the $ObjectURL variable and execute the query on the database.

The statements for the query and its execution results should be placed after the lines that reads *$result = null;*:

$query = "SELECT * FROM image_data i WHERE (i.URL = '$ObjectURL')";
$result = mysql_query($query, $connection);

D. Verify the passed parameters

Now we need to alter the next two if() statements to validate the input. The first if() statement will check whether or not any rows were returned. If the user somehow made a selection on the previous page that does not correspond to any elements in the database, the query will not find any matching database elements and the result will not contain any rows. If a row was returned, we should retrieve the value of its “Description” field so that it may be compared with the $ObjectToFind variable retrieved from the form.

D.1. Check for a matching database row

The first if statement that we will be revising is the one that currently reads

```
if(false) //if the number of rows is zero
{
    //this will be executed if the result contains zero rows
}
```

The first thing to do will be to update the contents of the if() statement. Replace the word “false” with:

mysql_numrows($result) == 0

Now, we need to fill in the contents of the if() block. Replace the line that currently reads
//this will be executed if the result contains zero rows
with
$validinput = false;

Immediately following that if() block, you will find a line which reads *$result_object = null;* . We need to replace this with the MySQL command which will retrieve the value of the "Description" field from the first row of our result. To do this, change this line to read
$result_object = mysql_result($result, 0, "Description");

D.2. Check for matching parameters

The second if() statement should compare $ObjectToFind with the value of $result_object obtained from the database. The line currently reads *if (false) //if the values of the parameters match* . It should be changed to:
if ($ObjectToFind == $result_object)

The code inside of this if() block will execute if the value of $ObjectToFind retrieved from the transfer.php page matches that of the description of the object matching the URL of $ObjectURL in the database. The code inside of the if() block should be changed from:
//this will be executed if the values of the parameters match
to
$validinput = true;
This will let the program know that valid input was passed from the transfer.php page and that the values match.

You should then change the code inside the else block from:
//this will be executed if the values of the parameters do not match
to:
$validinput = false;

E. Check for complete input validity

Now we have checked all the conditions to determine whether the parameters passed from the transfer.php page are valid or not. The only thing left to do is to change the code in our transfer_action.php page so that if the parameters were not valid and correct, the transfer will not take place.

You will see a line which currently reads
if (true) //if the user entered proper information on the transfer.php page
We need to change this to read
if ($validinput == true)
This way, the actual transfer code contained within this if() block will only execute if everything is correct.

Now, scroll down all the way to the bottom of the transfer_action.php code and you will see an else block which currently reads
else

{
//this will be executed if there are no matches
}

Insert code inside of this else block which will use an echo statement to output a message to the user informing him or her that there was an error processing the form and request that the user return to the transfer.php page, similar to that shown below:

echo 'There was a problem with the transfer, please click <a href="./default.html">here</a> to return to the Transfer page and try again.</br><br />';

The required edits to transfer_action.php should now be complete. Save your document.

Screenshot # 6: When you have completed this step, take a screenshot of your new transfer.php page.

Screenshot # 7: Now take 2 series' of screenshots. In the first series, take screenshots of:

- **valid data entered into the transfer.php page**
- **the resulting transfer_action.php page**

Screenshot # 8: In the second series, take screenshots of:

- **invalid data (say a negative number for the Amount to be transferred) entered into the transfer.php page**
- **the resulting transfer_action.php page**

Screenshot # 9: Click on the XSRF link in section 7 again and take a screenshot of the resulting page

11. Further Problems

There are still a number of bugs in the transfer system's code (as you might have noticed above to obtain Screenshot 8, while passing invalid data for the Amount to be transferred). You will now need to fix these bugs. Some of these bugs which need to be addressed are:

- If a user's account balance is negative, he can still transfer money to another account. If a user does have a negative account balance, he should not be able to transfer money.
- If a user's account balance is less than the amount she wishes to transfer, she is still allowed to make the transfer (which will result in her having a negative balance). This should no occur.
- Users are able to transfer money to their own accounts. This is unnecessary and should not occur.
- Users are able to transfer a negative amount of money to someone else's account (thereby increasing the amount of money in their own at someone else's expense).

These issues are fairly simple and can be easily fixed. We will now walk through the process of fixing the first one together. After that, you will need to fix the remaining three bugs.

The first bug is that a user can transfer money to another account even when he has a negative account balance. To fix this, we simply need to check the user's account balance before we initiate any transfer.

In the transfer_action.php code, scroll down to the line that says

if (false) //if the source account has a negative balance

We will need to use this if statement to compare the source account's balance with zero to determine if the amount in the account is less than or equal to zero. A look at the code reveals that there is a variable named "$source_balance". This is the variable that holds the amount of money in the source account. We will need to replace the contents of the if() statement to be:

if ($source_balance <= 0)

We will then need to insert a line into the if() block which will issue a message to the user if this condition has occurred. Immediately following the line that says

//check the source account balance

Insert the following line:

echo '<br />The source account has a negative balance. No transfer will occur.<br />';

Now save the transfer_action.php code and return to the application. To test the fix, you will need to alter the data in the database (http://localhost/phpmyadmin) to give your account a negative balance.

<u>Screenshots # 10:</u> Once you have done this, you will take three screenshots: a screenshot of the database contents showing your negative balance, one of the transfer.php page in which you request to transfer money, and a screenshot of the resulting transfer_action.php page which should deny you the ability to transfer funds due to your negative balance.

<u>Screenshots # 11, 12, 13:</u>

After you have proceeded so far, fix the three remaining bugs. For each bug that you fix, you will need to take a set of three screenshots:

- **A screenshot of the database contents showing the initial conditions,**
- **A screenshot of the transfer.php page where you try to enter improper data related to the bug you have fixed, and**
- **A screenshot of the resulting transfer_action.php page which should deny you the ability to transfer funds based on the conditions related to the bug you have fixed**

<u>Screenshot # 14:</u> Take a final screenshot of the contents of your "Accounts" table showing the details of each account.

<u>WHAT TO TURN IN:</u>

(1) Hardcopy of the following:

- Screenshots 1 through 14 as described above
- The transfer.php code after all of your modifications
- The transfer_action.php code after all of your modifications
- Trace the structure of your transfer_action.php code and briefly explain why you were able to easily fix the above four bugs by just replacing the *false* Boolean value with an appropriate condition in the *if* statement.

(2) Video Recording: After you have completely setup the banking website and tested it (i.e., proceeded until Screenshot 14), record a video illustrating the following (the CAPTCHA scheme should work for verification for cases iii, iv, v and vi). Upload the video through GoogleDrive and e-mail the link to natarajan.meghanathan@jsums.edu

(i) An unsuccessful login attempt
(ii) A successful login and check the account balance.
(iii) Transfer a positive amount from one account to another account, with the transfer verified using the CAPTCHA scheme
(iv) Attempt to transfer a negative amount to another account. It should not work an error message must be displayed.
(v) Attempt to transfer an amount from an account that exceeds the balance in that account. An error message must be displayed.
(vi) Attempt to transfer a positive amount to their own account. An error message should be displayed.

Images to choose from (use http://images.google.com): Choose a distinct image of each type assigned to you

#	Student Name	Image Types
1	Bernard Aldrich	Books, Laptop, Cloud, Broccoli, Rat, Horse, Snake, Couch, Television, Bus
2	Anirudh Reddy	Tree, Car, Sandwich, Cat, Sheep, Phone, Rainbow, Laptop, Apple, Microwave
3	Yashwanth Divanji	Tree, Motorbike, Rainbow, Donut, Cycle, Window, Butterfly, Rat, Dryer, Table
4	Keerthi Donepudi	Microwave, Table, Rabbit, Motorbike, Rose, Eggplant, Pizza, Tablet, Phone, House
5	Rashad Evans	Dryer, Rat, Horse, Tree, Pig, Books, Table, Cookie, Banana, Rainbow
6	Mesafint Fanuel	Snake, Cookie, Dog, Chair, Table, Books, Sheep, Bus, Horse, Tablet
7	Phillip Graise	Dryer, Snake, Car, Chair, Banana, Sheep, Pig, Road, Tablet, Tree
8	Damon Jones	Television, Broccoli, Airplane, Couch, Tree, Cycle, Rat, Hurricane, Microwave, Cookie
9	Joel Maddirala	Window, Cloud, Motorbike, Stove, Snake, Tree, Rose, Dryer, Bus, Butterfly
10	Jerry McLin	Hurricane, Rainbow, Stove, Phone, House, Banana, Peacock, Horse, Towel, Tomato
11	Kranthi Nalivela	Rose, Cabbage, Window, House, Chair, Phone, Peacock, Rat, Motorbike, Cat
12	Victoria O'Harroll	Tree, Cabbage, Elephant, Table, House, Eggplant, Tablet, Dog, Towel, Horse
13	Ayotunde Olutade	Tablet, Window, Rose, Laptop, Cloud, Snake, Books, Television, Broccoli, Cat
14	Antranella Pendleton	Cookie, Tree, Dog, Cycle, Cloud, Sandwich, Cabbage, Television, Phone, Peacock
15	Deepak Vadlamudi	Tomato, Peacock, Television, Horse, Rabbit, Tablet, Phone, Cabbage, Car, Motorbike