Assignment 3

# PART 1.A

For part 1.a the two files to look at are part1.html and name.php. The html file lets a user submit their name and then when submitted the name.php is used to display the names. The name.php file uses the HTTP POST request to get the data.

# PART 1.B

For part 1.b the two files to look at are part1b.html and nameGET.php. These files have the same purpose as part 1.a except they use HTTP GET instead of POST.

The main difference I can see between the GET and POST requests are that GET request displays the information in the URL which isn’t good for security reasons whereas the POST dose not display any information in the URL.

GET URL:



POST URL:



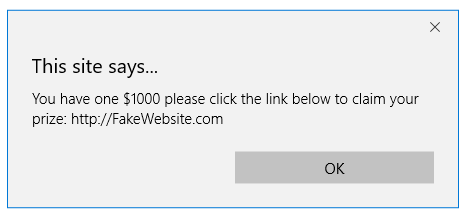
As you can see there is a clear difference in the URLS above and this could be of concern as what if a password was stored in the URL easily viewable for hackers.

# Part 1.c

The security on my script in part 1.a was already prone to a XXS attack. So to carry out a XXS attack on my code I entered this script: <script> alert('You have one $1000 please click the link below to claim your prize: http://FakeWebsite.com')</script> . Like so:



This is a very basic attack as this is all it does displays a window with some text:



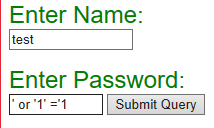
This is all the attack done but some naïve user could easily fall for this.

# part 2.a

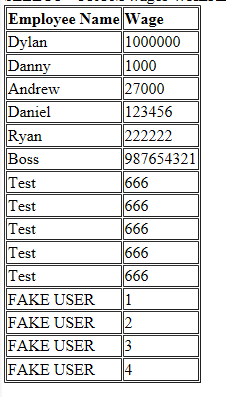
For part 2.a I made a fictional database of employee wages where the employee had to enter their name and password to see there wage.

# part 2.b

My script was already vulnerable to an SQL attack so there was no modification needed here all that was left was to inject some malicious SQL code.



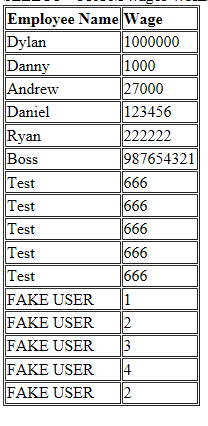
Entering this code into the password will get the database to produce all the users and all their wages.



This is the result of the code above.

For the second SQL injection enter anything into Name text field, and for the password enter a'; INSERT INTO wages VALUES('FAKE USER',2 ,'fakeuser');#4

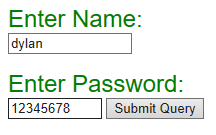
This will add another user to the database without any privileges or anything:

As you can see another user has been added.

# Part 3

MAX LENGTH

<input type="text" maxlength="8" name="Employee\_Password" size="10">



As you can see this has just restricted the user password to a max input length of 8 it will not allow anymore characters.

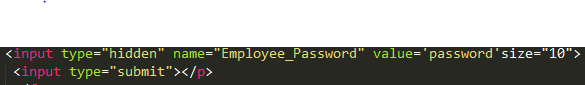
The max length has some pros and cons setting a max length for password restricts the users input making a password possibly weaker. It is also frustration for users if they want a password over the limit. It doesn’t offer much protection as SQL injection can be very short. The best option is to allow the user to enter whatever they what and then have this input processed and checked for SQL or any garbage and this would be more effective then limiting the user input an example would be regular expressions.

Read-Only





The read only forum is good to stop attackers from entering malicious code as it doesn’t enable them to enter anything, the bad part of it is that everything is predefined so there is no variation and everything is the same in my case I set password and had it as read only and this only printed out 1 value over and over.



Hidden type doesn’t allow the user to see the option which is similar to the example above.

# Part 4

The file to look at here is the validation.php file here the file takes a few user inputs and checks the input to see if it matches with what is expected. The purpose of this php script is to check user input and only let valid input pass this was done using a number of different form fields and regular expressions.

The first function we will look at is the empty(), this just checks to see If the field is empty on not a simple Boolean.

# Part 5

For the cookies I Made and extension to the validation.php file to make a cookie when a user name is inputted.

With cookies the security option is the Http Boolean and all this dose is deciding whether the cookie is accessible to java script or not.



Here I do a simple check to see in the dob variable is not empty them go and do the next function.

Next function to look at is preg\_match(). I used this to check if the variable format matched the regular expression I made.



Here is have it compare the reg variable to the premade regular expression that works with all Irish registrations.

Next function to look at is filter\_input() this takes in a variable and optionally filters it with what is specified.



I also used the htmlentities() function to stop XXS attacked from the URL.



I use htmlspecialchars() for the name in this file as well as not all names are on the same keyboard.



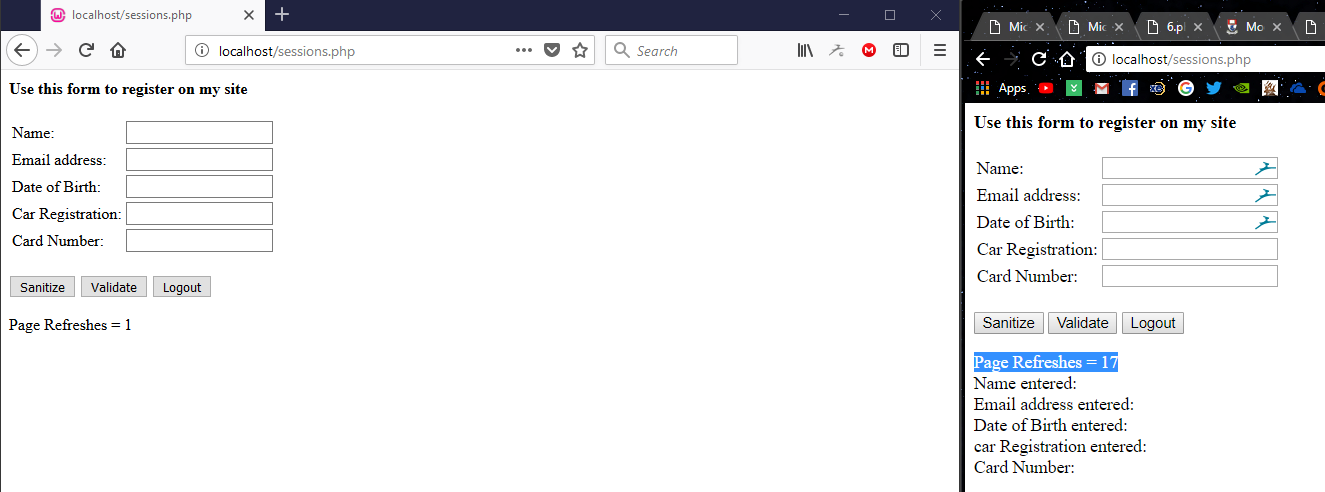
I also made another project quote.php just securing the project from part 2 by using the quote() function.



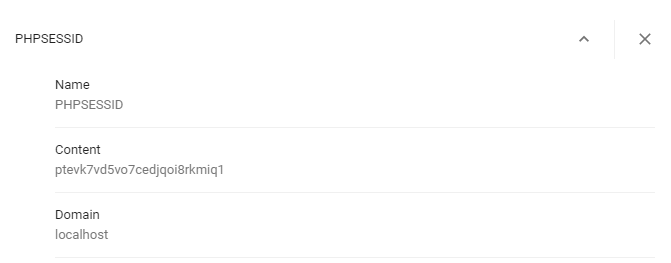
This stopped the ability to enter the code I had in part 2 and made the programme more secure.

# Part 6

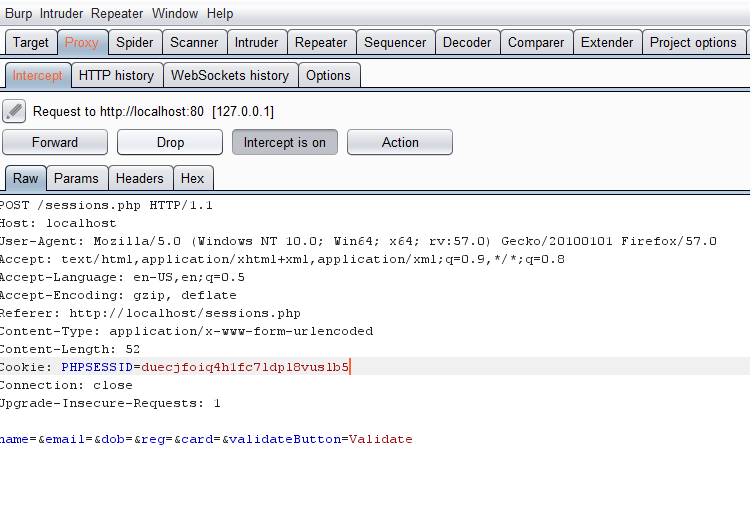
The finale file is the sessions.php file this is used to show sessions. But we are going to do a basic session hijacking. Please see below for how to do this.



Two Different browsers Firefox with 1 page refresh and chrome with 17 page refreshes.



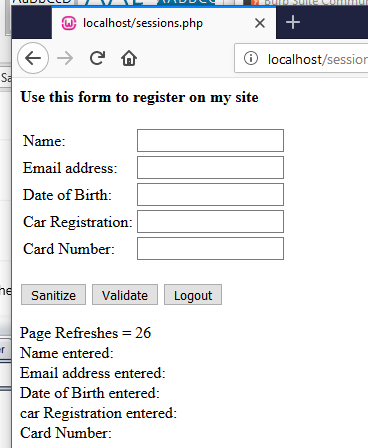
Getting the session id for the google chrome website with 17 refreshes. Then using Burp intercept a Firefox request.



Sending a request and interception the request from Firefox with burp see the session id.



Replacing the session id with chromes session id and we have hijacked the chromes session.



We hijacked the chrome session and have all the refreshed pages. (There is a few more total than expected this is because of a time difference and a couple more refreshes before it worked)