



Waterford Institute of Technology

Web App Security

Jimmy McGibney

Jack Donoghue

BSc (Hons) in Computer Forensics and Security – Year 2

20072172

Donoghuej97@gmail.com

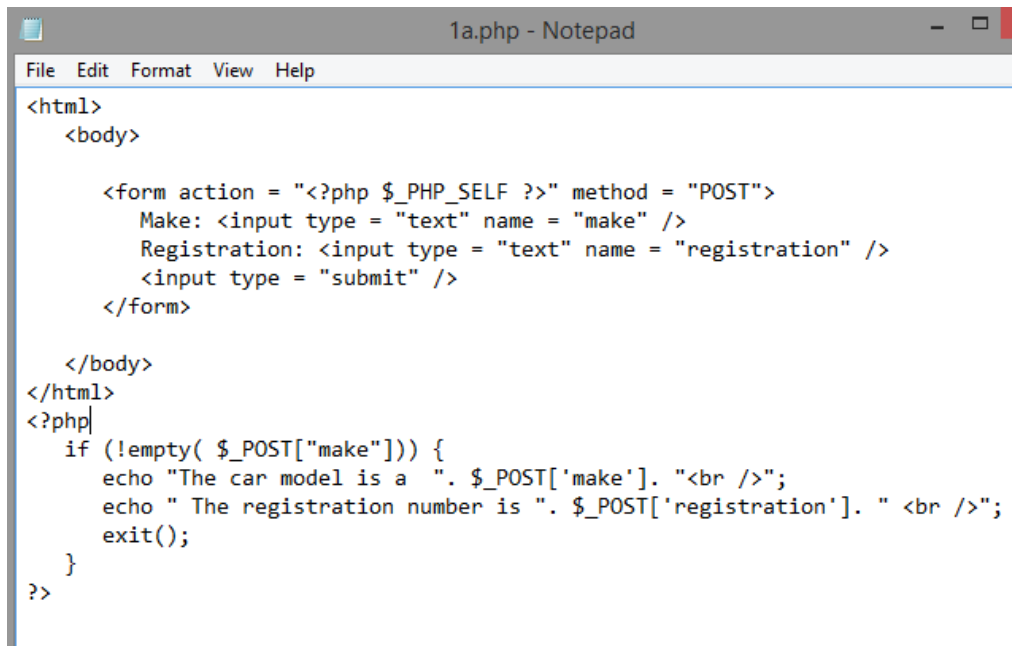
Table of Contents

1A.....	3
POST Function	3
1B.....	4
GET Function	4
GET vs POST	5
1C.....	6
XSS example	6
2A.....	7
Database Query	7
2A.php.....	8
2B.php	9
SQL Injection Attack	9
3.....	10
HTML Forms	10
Radio Button	11
ReadOnly.....	13
Weakness of HTML Forms	14
4Empty	15
Validation	15
4Preg_match.....	16
4SpecChar	17
HTML Special Characters	17
4StripTags	18
5.....	19
Cookies.....	19
6.....	22
Sessions.....	22
Session ID / Hijacking	22
Hijacking.....	23
Bibliography.....	24

1A

POST Function

The first PHP script uses the POST function to retrieve input from the html input. As seen below the file is based around a car model / registration system and uses the POST function to fetch the values entered by the user.



```

1a.php - Notepad
File Edit Format View Help
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Make: <input type = "text" name = "make" />
      Registration: <input type = "text" name = "registration" />
      <input type = "submit" />
    </form>

  </body>
</html>
<?php
  if (!empty( $_POST["make"])) {
    echo "The car model is a ". $_POST['make']. "<br />";
    echo " The registration number is ". $_POST['registration']. " <br />";
    exit();
  }
?>

```

Figure 1 1a.php POST example

I then tested its functionality in my localhost 'localhost/1a.php'

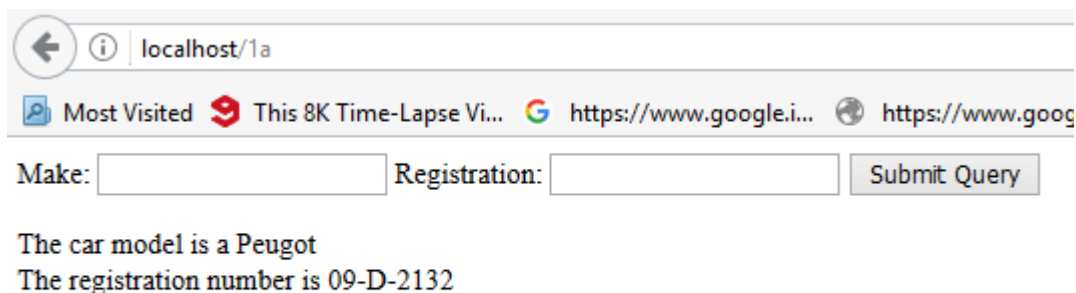
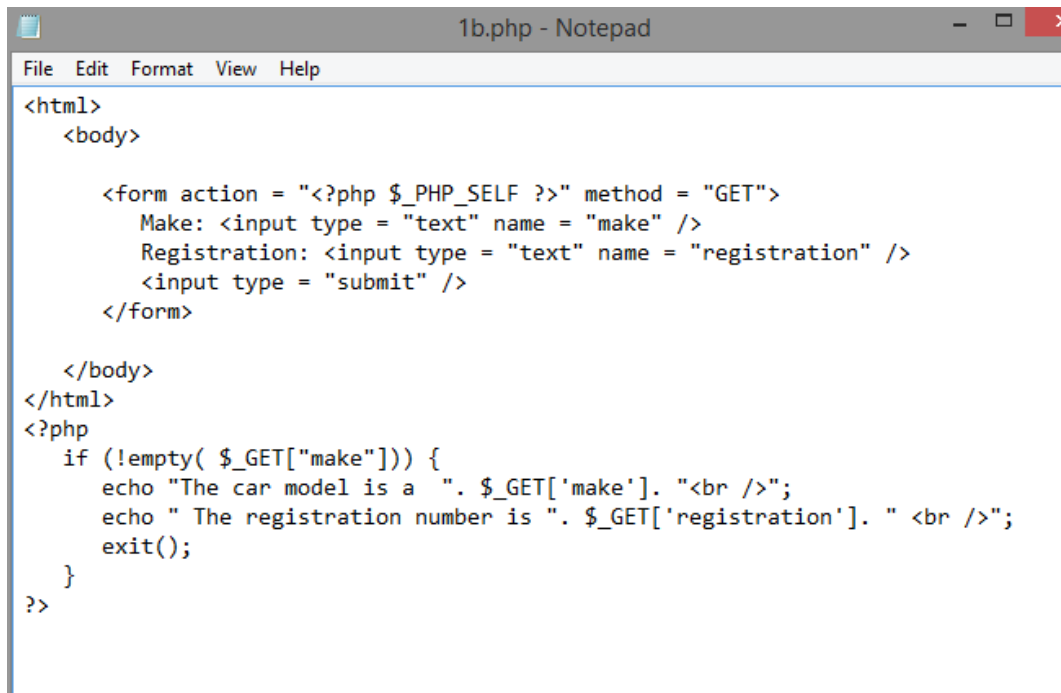


Figure 2 1a taking and returning input with POST

1B

GET Function

The second PHP script uses the less safe GET function to communicate between the html user input and the PHP script this script is similar to 1a, however it GET replaces POST.



```

1b.php - Notepad
File Edit Format View Help
<html>
  <body>

    <form action = "<?php $_PHP_SELF ?>" method = "GET">
      Make: <input type = "text" name = "make" />
      Registration: <input type = "text" name = "registration" />
      <input type = "submit" />
    </form>

  </body>
</html>
<?php
  if (!empty( $_GET["make"])) {
    echo "The car model is a ". $_GET['make']. "<br />";
    echo " The registration number is ". $_GET['registration']. " <br />";
    exit();
  }
?>

```

Figure 3 1b.php GET function example

Then I tested the functionality using my localhost.

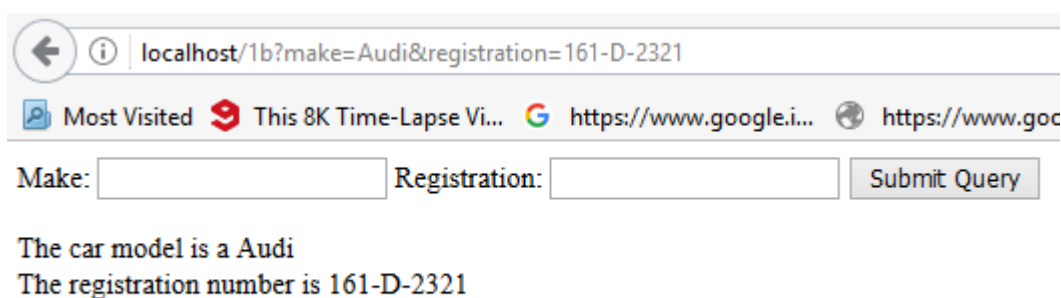


Figure 4 1b working with GET

GET vs POST

The two functions previously displayed in this document seem to perform similar tasks when it comes to the user end, however there are vast differences between them when it comes to security. One of the main differences between them is how POST allows for any input including files while GET is restricted to ASCII characters¹.

The main feature we are concerned with is safety, the GET function is known for not being as secure as POST, it displays whatever it retrieves in the URL bar, this also means that it is logged by your browser history, the links will contain the users input and may lead to problems when processing sensitive data including passwords.

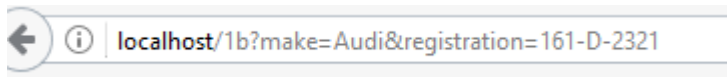


Figure 5 1b.php (GET) displaying inputs in URL

This example is taken from 1b.php and shows the input entered by the user, in this case the Make was 'Audi' and the registration was '161-D-2321' this outlines the key security concerns relating to the GET function.

Here is an example of what the URL looks like when POST is used instead of GET.

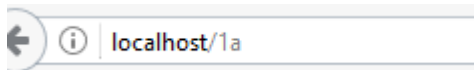


Figure 6 1a.php (POST) doesn't display

¹ <http://www.diffen.com/difference/GET-vs-POST-HTTP-Requests>
GET and POST information

1C

XSS example

The first cross site scripting example displayed here is a basic attack, I did this just to demonstrate that 1a.php was vulnerable to XSS. The code used for this is

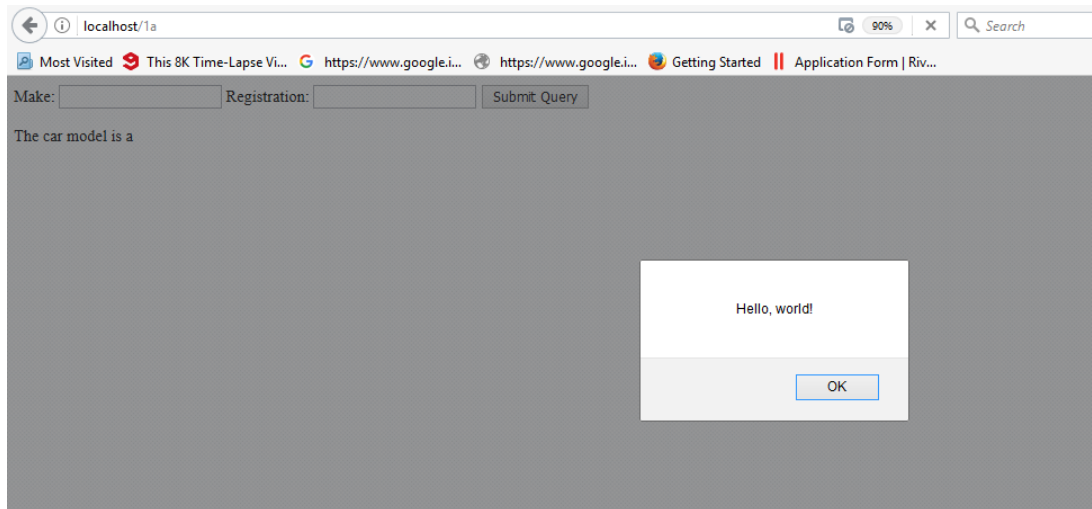


Figure 7 Showing 1a.php is vulnerable to XSS

I then attempted to insert code that would redirect the page using JavaScript, I did this by entering '`<script> window.location.href = "http://twitter.com"; alert('Redirecting to our new website');</script>`' into the text box, when I did this the page notified me it was redirecting and then redirected to the twitter website, this could be useful if the attacker had a malicious website that replicated the login page of a well-known website.

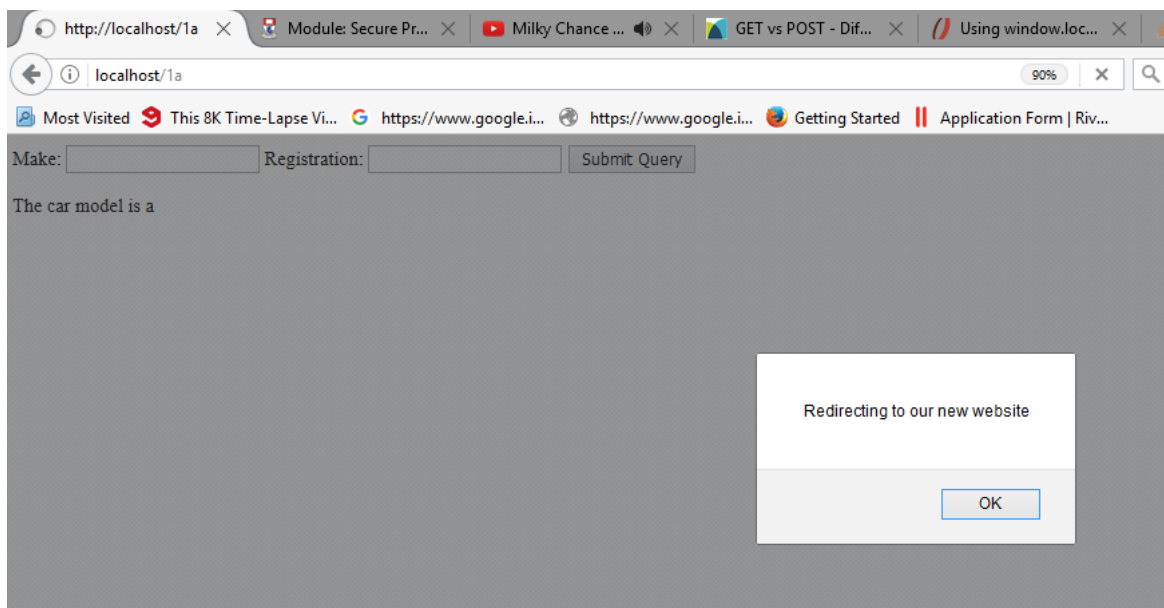


Figure 8 1a.php with SQL injection

2A

Database Query

For this section I created a database using the instructions displayed in our labs, I first logged into phpMyAdmin and created a database called 'CarHub'.

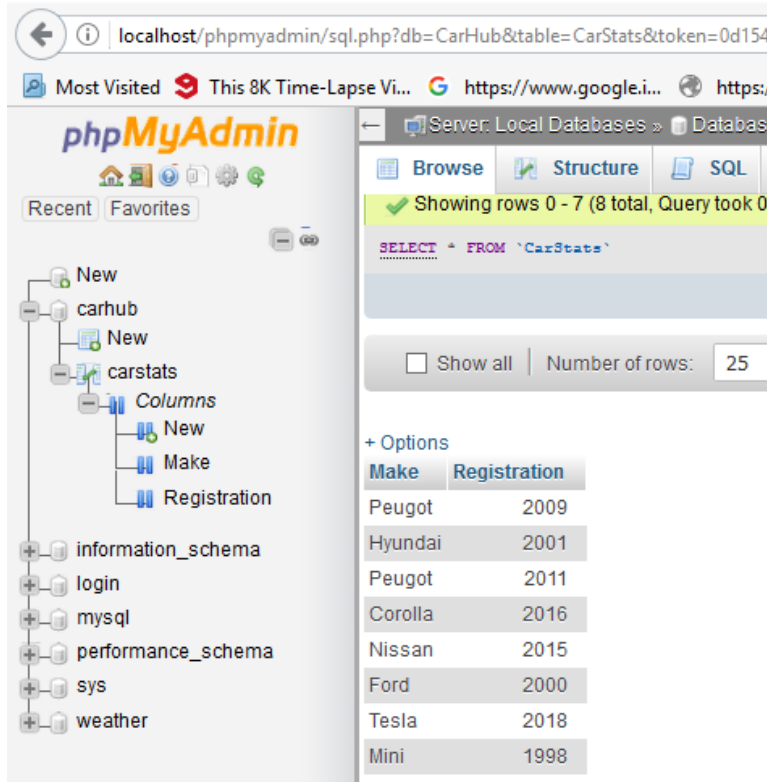


Figure 9 phpMyAdmin CarHub database

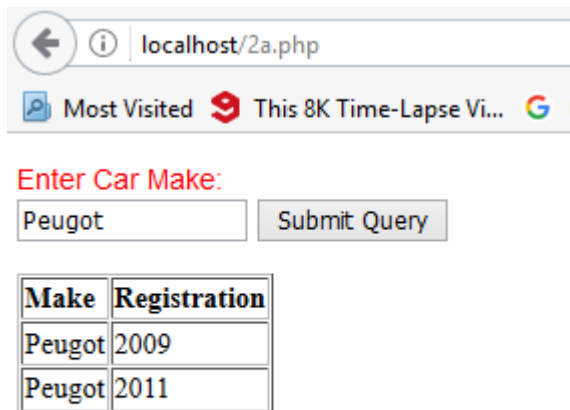
I then added table called users with two columns, I chose to populate my table by importing an SQL file that I had modified from a previous lab, I changed the variable names to match what type of input I had specified in my table, which was two columns with TEXT.

Make	Registration
Peugot	2009
Hyundai	2001
Peugot	2011
Corolla	2016
Nissan	2015
Ford	2000
Tesla	2018
Mini	1998

Figure 10 CarStats table

2A.php

Below is an excerpt of 2a.php running and retrieving results that match user input from the database.



Enter Car Make:

Peugot Submit Query

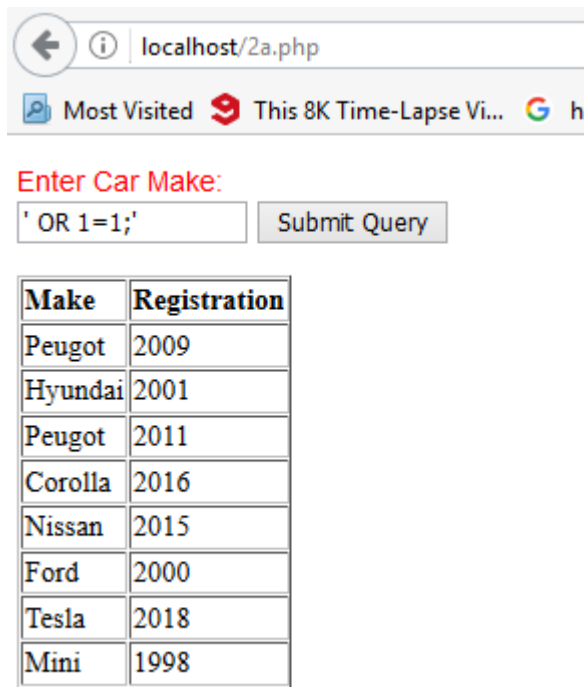
Make	Registration
Peugot	2009
Peugot	2011

Figure 11 2a.php working

2B.php

SQL Injection Attack

I carried out an SQL injection attack on the 2a.php and CarHub database in an attempt to prove the insecurities and the usefulness of SQL injection attacks when it comes to revealing information.



Enter Car Make:

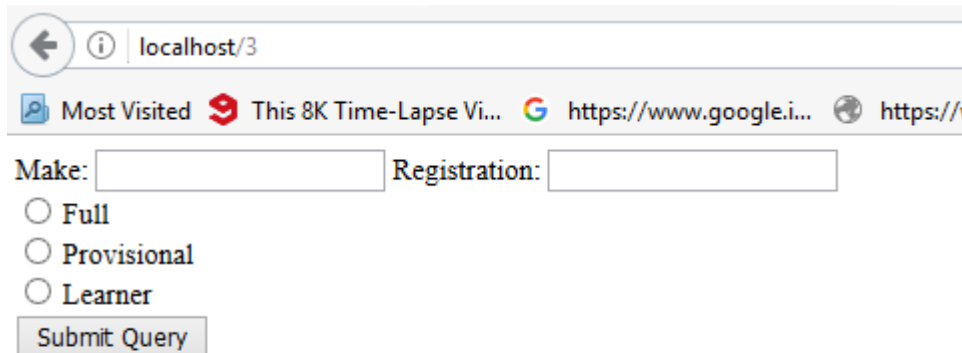
Make	Registration
Peugot	2009
Hyundai	2001
Peugot	2011
Corolla	2016
Nissan	2015
Ford	2000
Tesla	2018
Mini	1998

Figure 12 SQL injection attack on 2a.php

3

HTML Forms²

To introduce html forms into my files I decided to proceed with the existing file from 1a.php, I kept the POST function and decided to add some HTML forms to it to show the effects of the different forms on the existing program. Below is a picture of 1a.php/3.php after being modified to include my HTML forms.

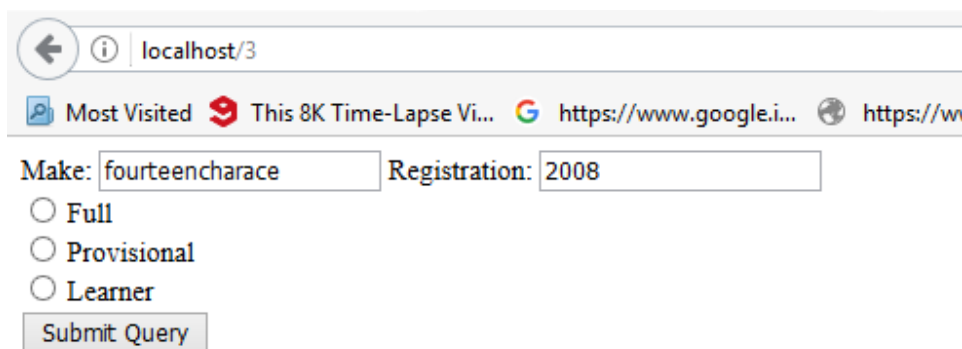


Make: Registration:

☐ Full
☐ Provisional
☐ Learner

Figure 13 3.php including radio and a max length

The HTML forms added to 1a.php include a max length on Make and Registration, I allocated 15 characters for the Make and 4 characters for the Registration year.



Make: Registration:

☐ Full
☐ Provisional
☐ Learner

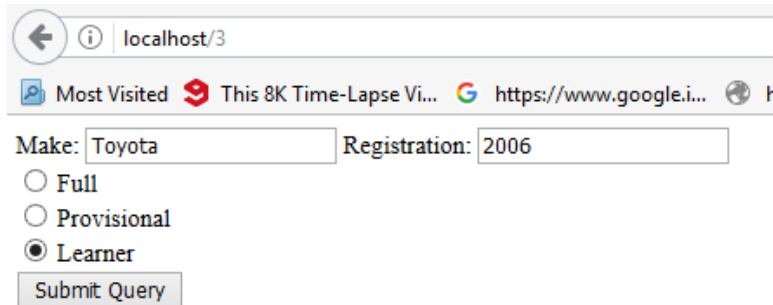
Figure 14 3.php displays max length form

² https://www.w3schools.com/html/html_forms.asp
HTML Forms

Radio Button

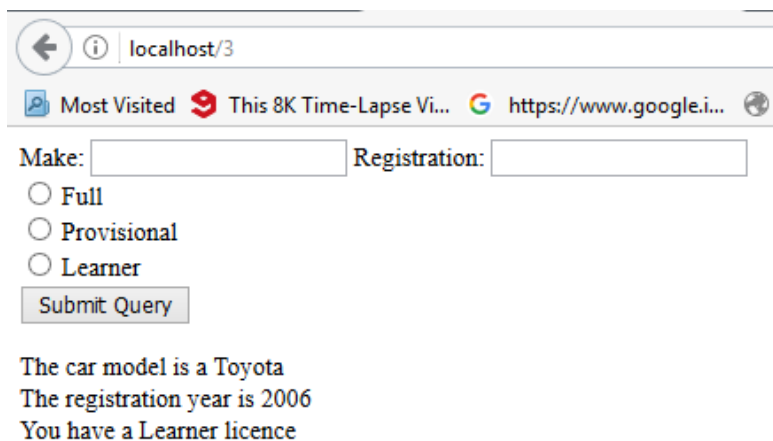
An additional feature I added was radio button input, this is an alternative way for the user to give information rather than typing in text, this can be used if you only offer a predetermined set of options for your service, in my example there are only three types of licences that I want the users to be able to choose from.

This is an example of the functionality of the radio buttons.



A screenshot of a web browser window. The address bar shows 'localhost/3'. The browser's Most Visited list includes 'This 8K Time-Lapse Vi...' and 'https://www.google.i...'. The form contains two text input fields: 'Make:' with the value 'Toyota' and 'Registration:' with the value '2006'. Below these fields are three radio button options: 'Full', 'Provisional', and 'Learner'. The 'Learner' option is selected, indicated by a filled circle. A 'Submit Query' button is located at the bottom of the form.

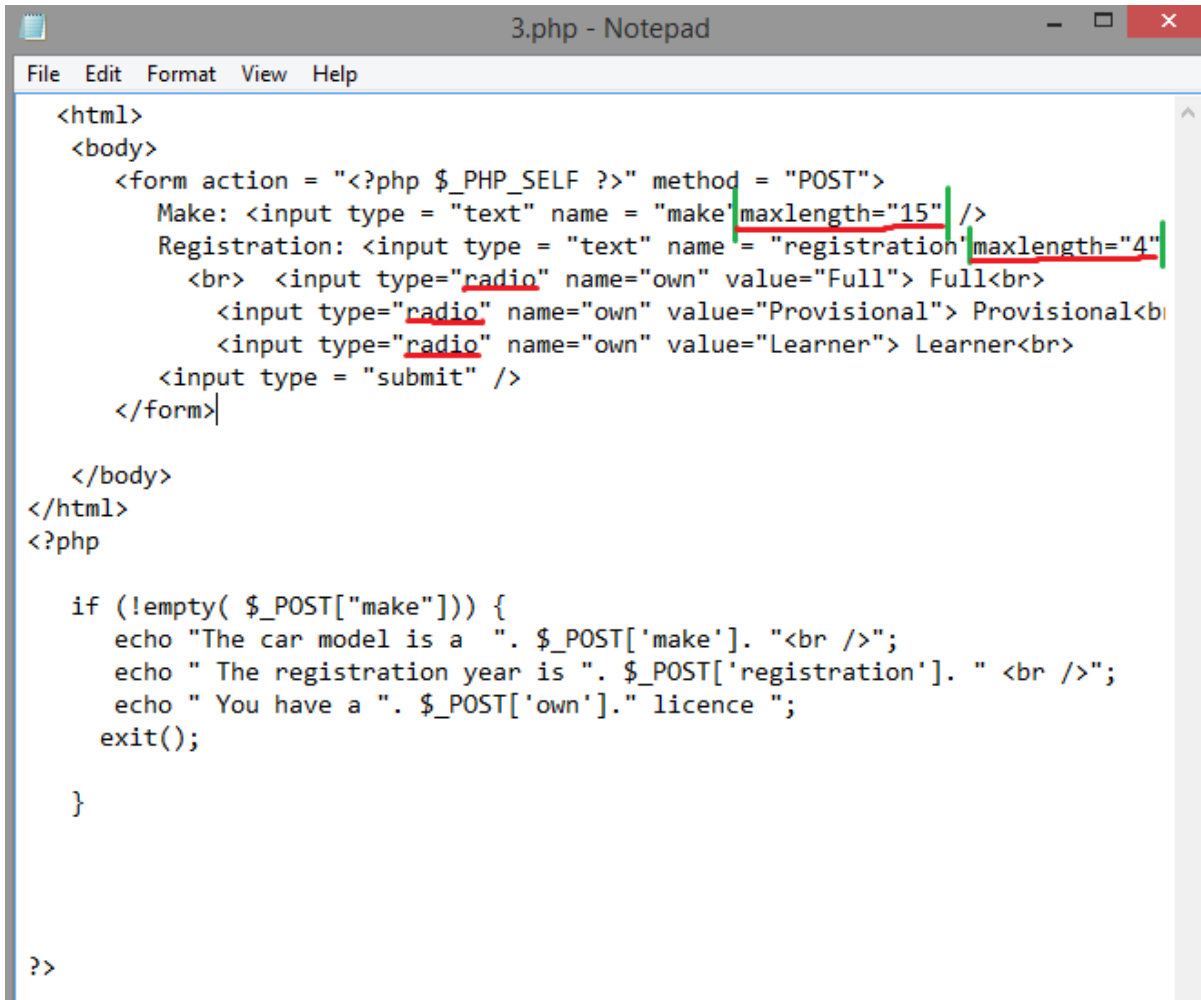
Figure 15 before entering



A screenshot of the same web browser window after the form has been submitted. The form fields are now empty. The radio button options remain, but none are selected. The 'Submit Query' button is still present. Below the form, the following text is displayed: 'The car model is a Toyota', 'The registration year is 2006', and 'You have a Learner licence'.

Figure 16 after entering

This is an excerpt of the code that went into making these HTML forms.



```

<html>
<body>
  <form action = "<?php $_PHP_SELF ?>" method = "POST">
    Make: <input type = "text" name = "make" maxlength="15" />
    Registration: <input type = "text" name = "registration" maxlength="4" />
    <br> <input type="radio" name="own" value="Full"> Full<br>
    <input type="radio" name="own" value="Provisional"> Provisional<br>
    <input type="radio" name="own" value="Learner"> Learner<br>
    <input type = "submit" />
  </form>

</body>
</html>
<?php

if (!empty( $_POST["make"])) {
  echo "The car model is a ". $_POST['make']. "<br />";
  echo " The registration year is ". $_POST['registration']. " <br />";
  echo " You have a ". $_POST['own']. " licence ";
  exit();
}

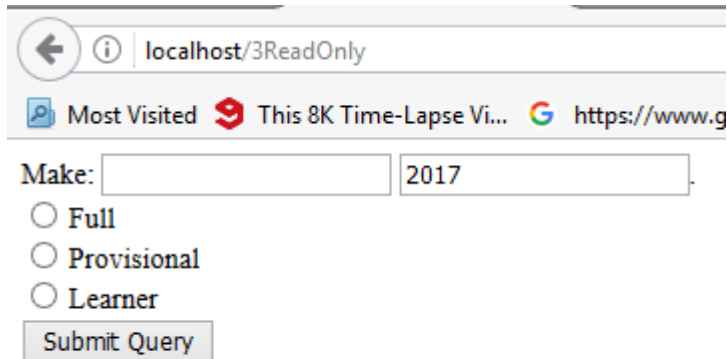
?>

```

Figure 17 maxlength and radio

ReadOnly

For the read only form I decided to make the registration year of all new vehicles to be fixed at 2017 and have only the make and the radio buttons selectable in this example.



localhost/3ReadOnly

Most Visited This 8K Time-Lapse Vi... https://www.g

Make: 2017

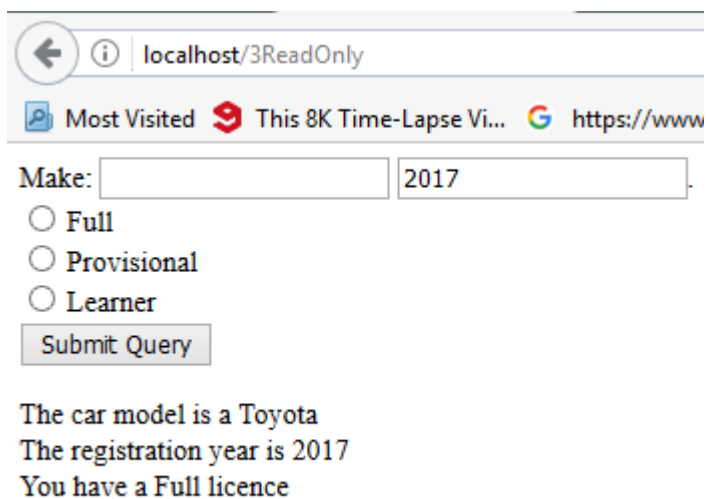
☐ Full

☐ Provisional

☐ Learner

Submit Query

Figure 18 2017 is unchangeable



localhost/3ReadOnly

Most Visited This 8K Time-Lapse Vi... https://www

Make: 2017

☐ Full

☐ Provisional

☐ Learner

Submit Query

The car model is a Toyota

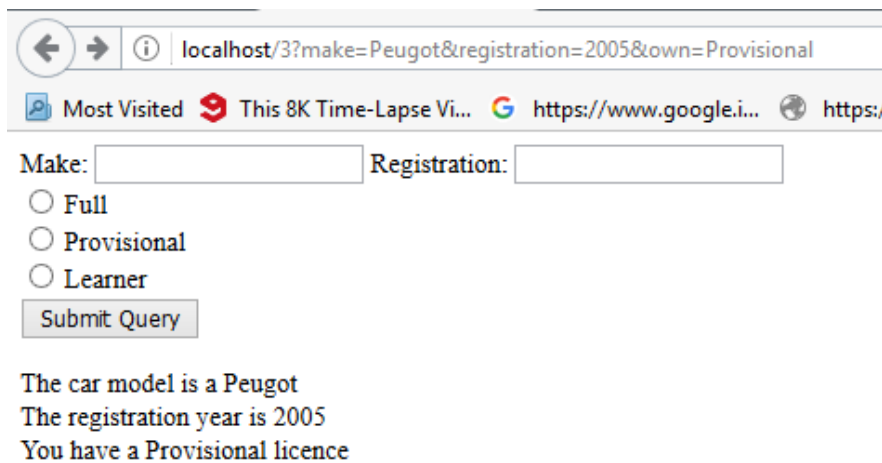
The registration year is 2017

You have a Full licence

Figure 19 It has a similar output to the others.

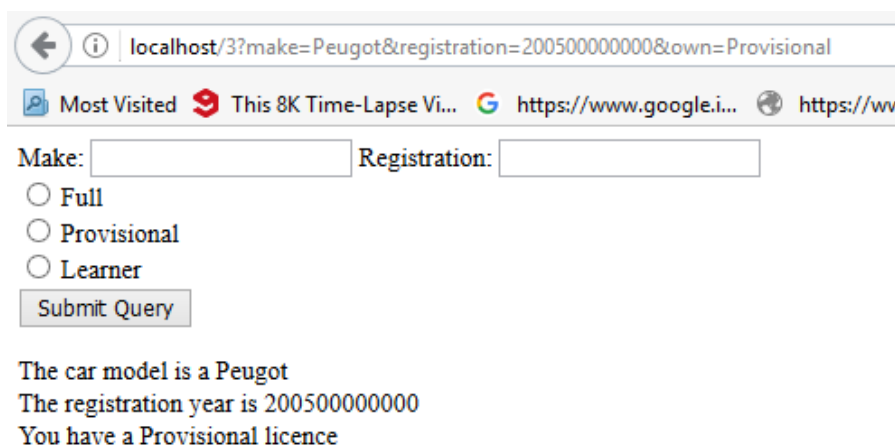
Weakness of HTML Forms

HTML Forms may restrict initial user input, but as with most things in HTML it is mainly for a visual representation, like with the character restriction, it may restrict user input initially but there are ways to bypass it, for example if you were using the GET function you could edit your input in the URL, for example: `?pin=1234` can be changed to `?pin=12345678`, this would allow the user to enter a larger than specified input. To show this example in use, I have reconfigured my 3.php file to use GET and will attempt to bypass the 4 character limit I have set on my Registration field.



The screenshot shows a web browser window with the address bar displaying `localhost/3?make=Peugot®istration=2005&own=Provisional`. Below the browser, the form fields are populated: "Make:" is "Peugot", "Registration:" is "2005", and the "own" radio button is selected. The "Submit Query" button is visible. Below the form, the output text reads: "The car model is a Peugeot", "The registration year is 2005", and "You have a Provisional licence".

Figure 20 Here we see our initial input is valid



The screenshot shows a web browser window with the address bar displaying `localhost/3?make=Peugot®istration=200500000000&own=Provisional`. Below the browser, the form fields are populated: "Make:" is "Peugot", "Registration:" is "200500000000", and the "own" radio button is selected. The "Submit Query" button is visible. Below the form, the output text reads: "The car model is a Peugeot", "The registration year is 200500000000", and "You have a Provisional licence".

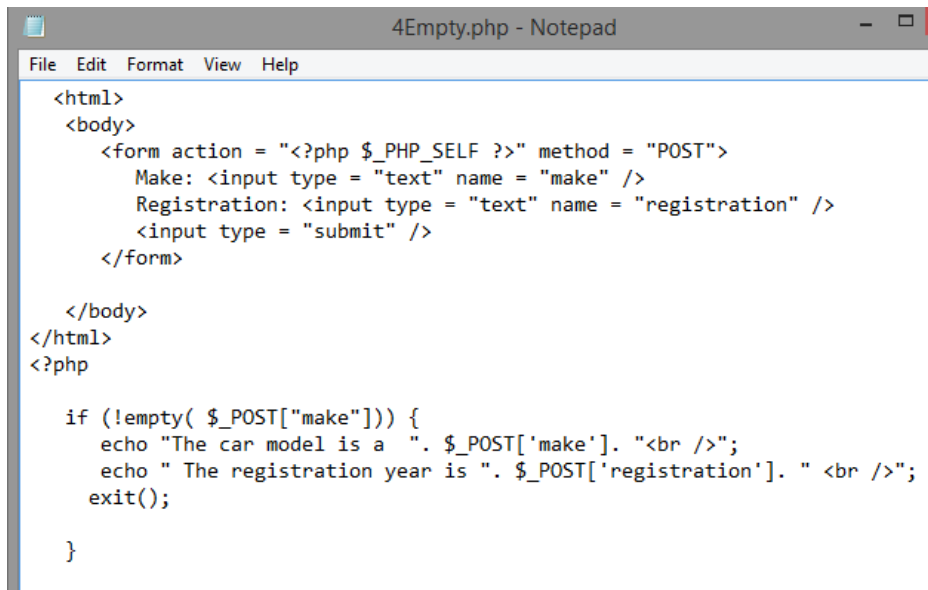
Figure 21 Here is the modified URL

As you can see after I edited the registration field in the browser I was allowed to enter a much larger than specified input where I should have only been allowed to enter a maximum of 4 characters.

4Empty

Validation

The first example of PHP validation I will be displaying is one that I have been using for most of my other scripts, the Empty or !Empty function ensures that my code will only execute after I have verified that a particular field is not Empty or !Empty this example can be seen below.



```

<html>
<body>
  <form action = "<?php $_PHP_SELF ?>" method = "POST">
    Make: <input type = "text" name = "make" />
    Registration: <input type = "text" name = "registration" />
    <input type = "submit" />
  </form>

</body>
</html>
<?php

if (!empty( $_POST["make"])) {
  echo "The car model is a ". $_POST['make']. "<br />";
  echo " The registration year is ". $_POST['registration']. " <br />";
  exit();
}

```

Figure 22 !empty function in use

As you can see I am checking to see if the 'make' field is not empty before I execute the rest of the code, this means that if I enter a registration value without entering a make the code will not run.

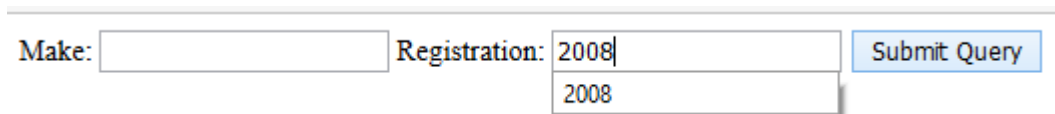


Figure 23 only registration is filled

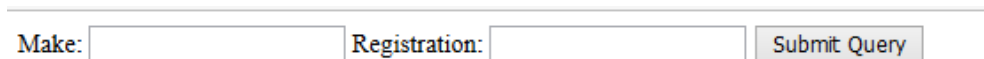
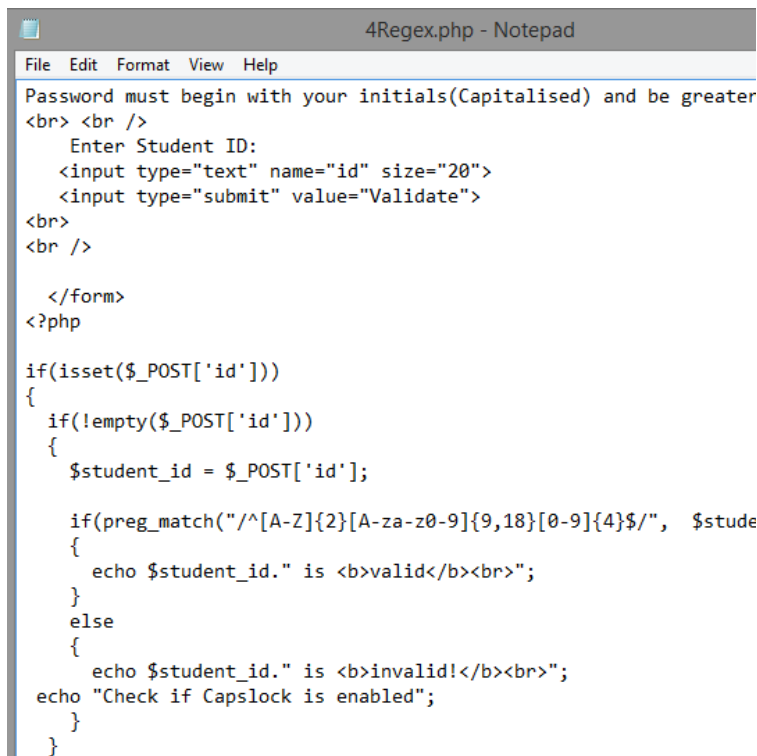


Figure 24 nothing is returned as a result

4Preg_match

For my regular expression validation I designed a login / password system that will only accept a very specific password and will reject anything that doesn't match the expression, the user must enter a password that begins with their initials in capitals, followed by their password (9 to 18 characters) and finished by their student code which is 4 numbers in length.



```

4Regex.php - Notepad
File Edit Format View Help
Password must begin with your initials(Capitalised) and be greater
<br> <br />
    Enter Student ID:
    <input type="text" name="id" size="20">
    <input type="submit" value="Validate">
<br>
<br />

</form>
<?php

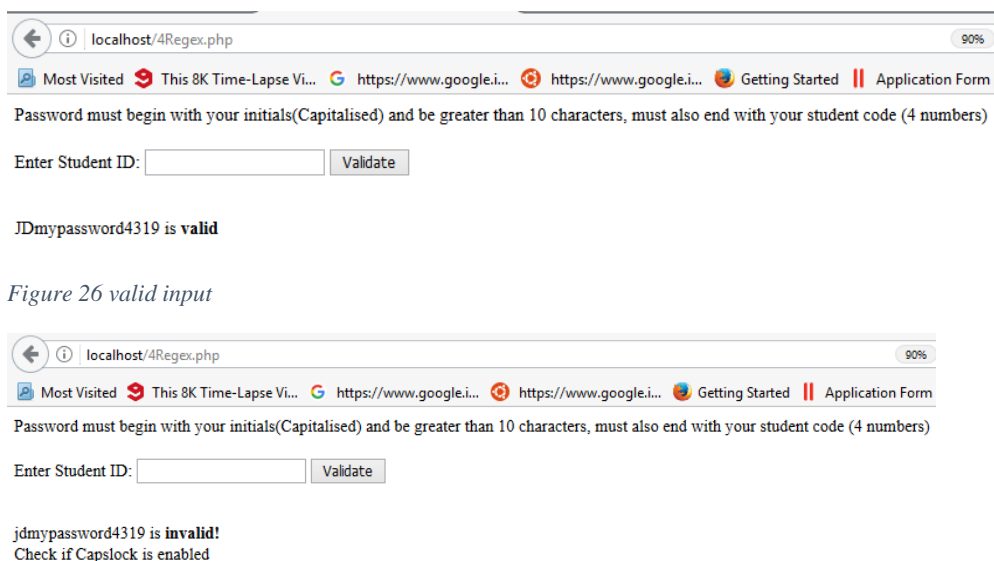
if(isset($_POST['id']))
{
    if(!empty($_POST['id']))
    {
        $student_id = $_POST['id'];

        if(preg_match("/^[A-Z]{2}[A-Za-z0-9]{9,18}[0-9]{4}$/", $stude
        {
            echo $student_id." is <b>valid</b><br>";
        }
        else
        {
            echo $student_id." is <b>invalid!</b><br>";
            echo "Check if Capslock is enabled";
        }
    }
}

```

Figure 25 regular expression

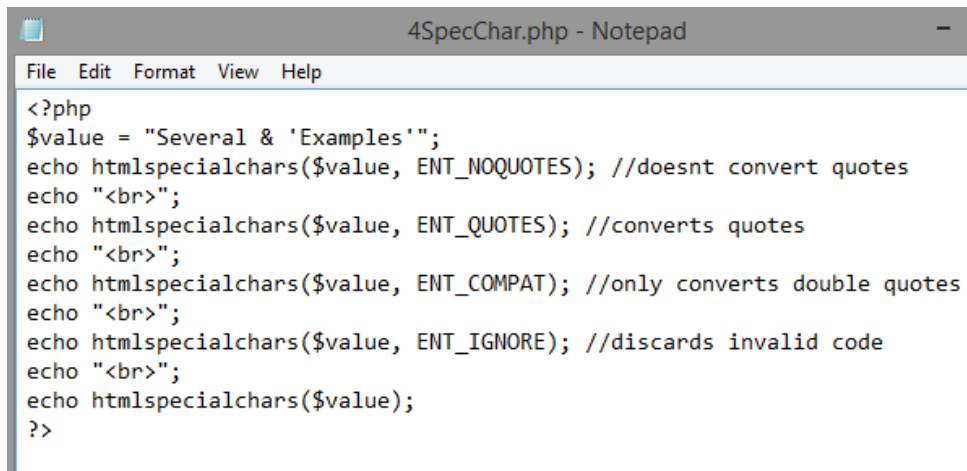
I then tested its functionality with some diverse inputs.



4SpecChar

HTML Special Characters

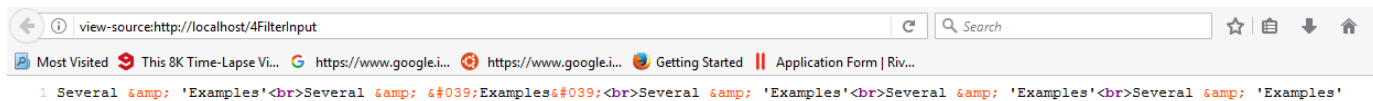
In this example I looked at HTML `specialchars()` function and how it interacts with certain HTML entities, while its output seems normal in the webpage if I look into the page source I can see how `specialchars()` has affected the entities. This is a brief look at the code



```
<?php
$value = "Several & 'Examples'";
echo htmlspecialchars($value, ENT_NOQUOTES); //doesn't convert quotes
echo "<br>";
echo htmlspecialchars($value, ENT_QUOTES); //converts quotes
echo "<br>";
echo htmlspecialchars($value, ENT_COMPAT); //only converts double quotes
echo "<br>";
echo htmlspecialchars($value, ENT_IGNORE); //discards invalid code
echo "<br>";
echo htmlspecialchars($value);
?>
```

Figure 28 4SpecChar file

This script simply prints "Several & 'Examples'" to the webpage, however if I enter the pages source we can see how `htmlspecialchars` has affected it.

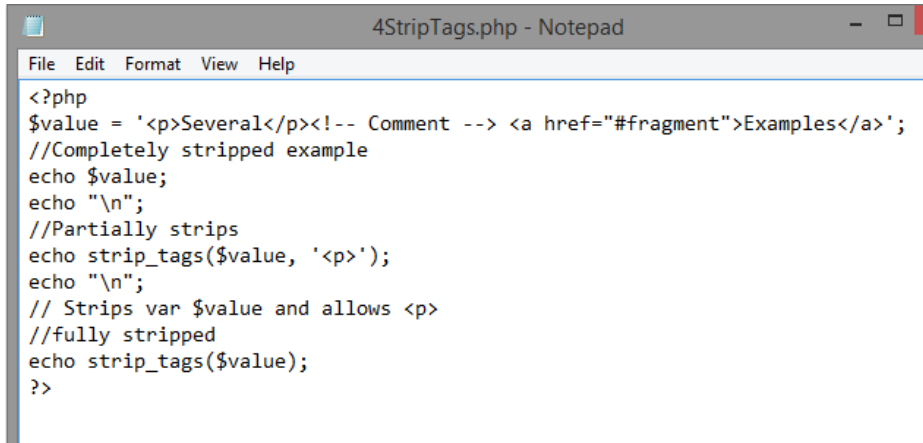


```
1 Several &amp; 'Examples'<br>Several &amp; &#039;Examples&#039;<br>Several &amp; 'Examples'<br>Several &amp; 'Examples'<br>Several &amp; 'Examples'
```

Figure 29 4SpecChar.php's source

4StripTags³

For this example I took a string that had many tags surrounding it, I have three different examples of how `strip_tags` works, firstly I start off with the full unstripped string printing, followed by a partially stripped string and lastly a completely stripped tag.



```

<?php
$value = '<p>Several</p><!-- Comment --> <a href="#fragment">Examples</a>';
//Completely stripped example
echo $value;
echo "\n";
//Partially strips
echo strip_tags($value, '<p>');
echo "\n";
// Strips var $value and allows <p>
//fully stripped
echo strip_tags($value);
?>

```

Figure 30 `strip_tags` script

```
1 <p>Several</p><!-- Comment --> <a href="#fragment">Examples</a>
```

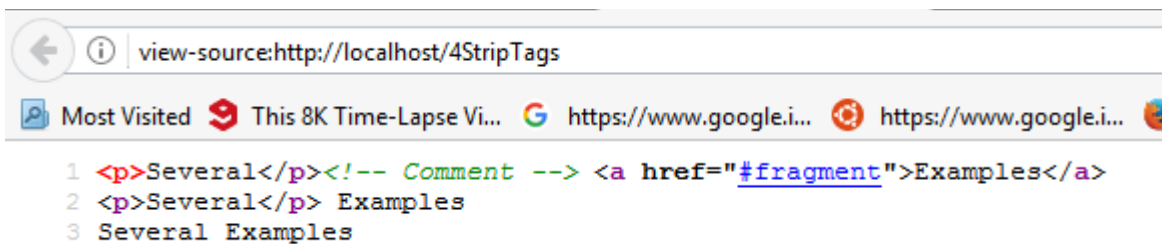
Figure 31 unstripped

```
2 <p>Several</p> Examples
```

Figure 32 partially stripped

```
3 Several Examples
```

Figure 33 fully stripped



```

1 <p>Several</p><!-- Comment --> <a href="#fragment">Examples</a>
2 <p>Several</p> Examples
3 Several Examples

```

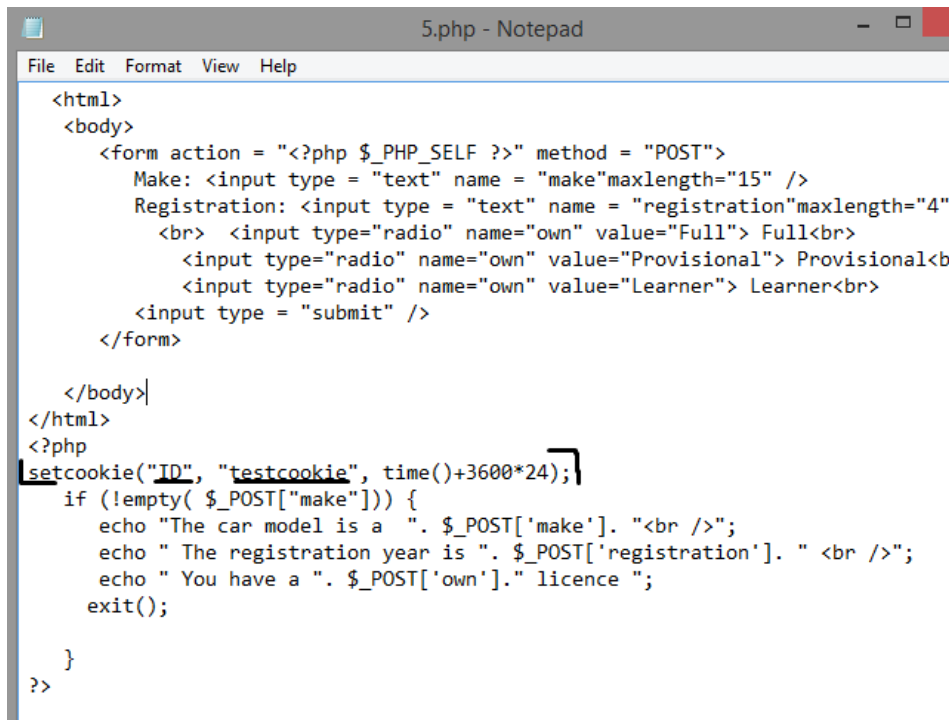
Figure 34 inside the page source

³ https://www.w3schools.com/php/func_string_strip_tags.asp
Strip Tags information

5

Cookies

For the next step I will be adding a session id to my car registration site, I will be using this to demonstrate how cookies work and also how web scarab can track user input and session ids. Below is a screenshot of the addition made to the car registration website.



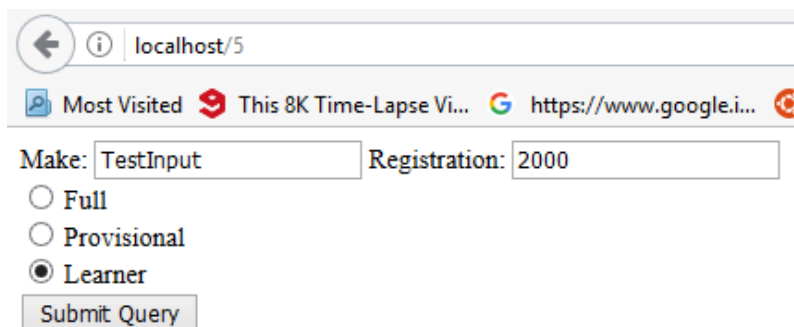
```

5.php - Notepad
File Edit Format View Help
<html>
<body>
  <form action = "<?php $_PHP_SELF ?>" method = "POST">
    Make: <input type = "text" name = "make"maxlength="15" />
    Registration: <input type = "text" name = "registration"maxlength="4"
      <br> <input type="radio" name="own" value="Full"> Full<br>
      <input type="radio" name="own" value="Provisional"> Provisional<br>
      <input type="radio" name="own" value="Learner"> Learner<br>
    <input type = "submit" />
  </form>
</body>
</html>
<?php
setcookie("ID", "testcookie", time()+3600*24);
if (!empty($_POST["make"])) {
  echo "The car model is a ". $_POST['make']. "<br />";
  echo " The registration year is ". $_POST['registration']. " <br />";
  echo " You have a ". $_POST['own']. " licence ";
  exit();
}
?>

```

Figure 35 Image of the file where the cookie is set

Now we can see the effect of setting the cookie by opening web scarab and intercepting requests from the localhost.



localhost/5

Most Visited This 8K Time-Lapse Vi... https://www.google.i...

Make: TestInput Registration: 2000

☐ Full
☐ Provisional
☒ Learner

Submit Query

Figure 36 5.php

The layout looks the same but as I submit query I will intercept the request and will be able to see if the cookie has been set properly.

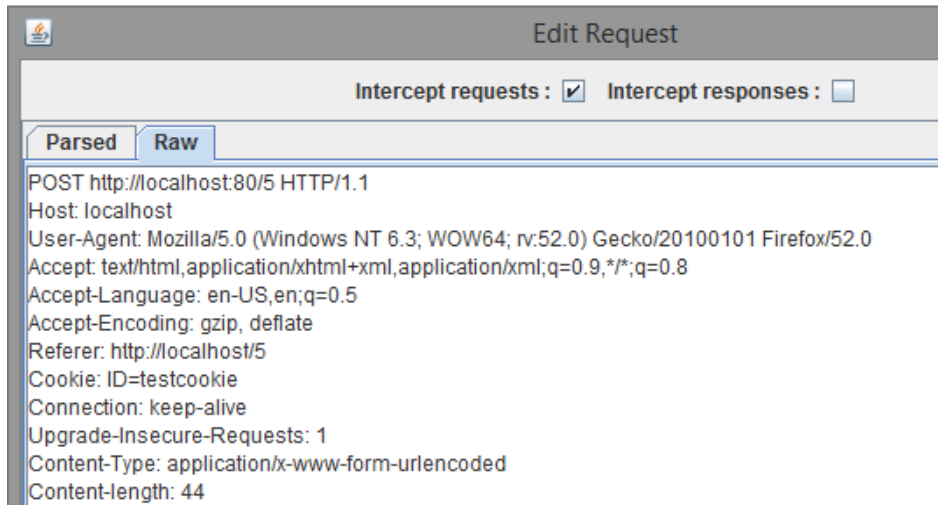


Figure 37 cookie has been properly set

Now that the cookie is working I can look to see if the submit query that I entered was also tracked by web scarab.

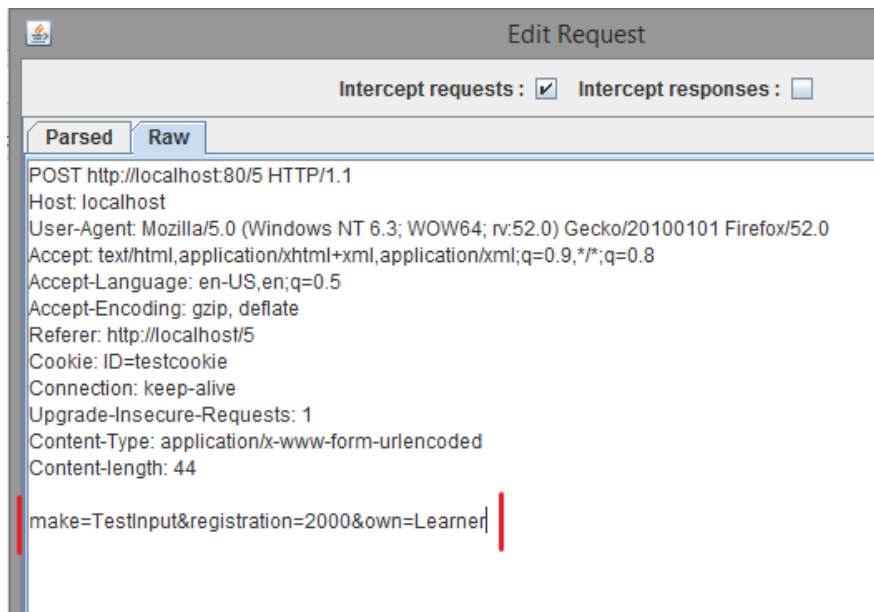


Figure 38 the submit query input is also retrieved

Now that I have explored setting cookies I will look at using cookies in certain situations.

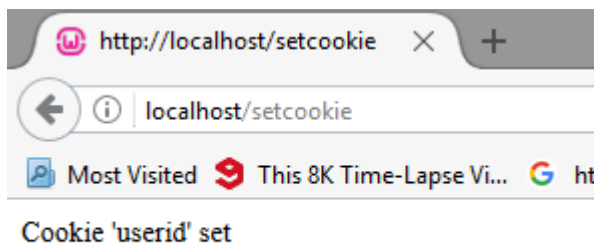


Figure 39 visiting the lab file 'setcookie'

Here I will explore the lab file 'setcookie' to show multiple cookies running in web scarab at once.

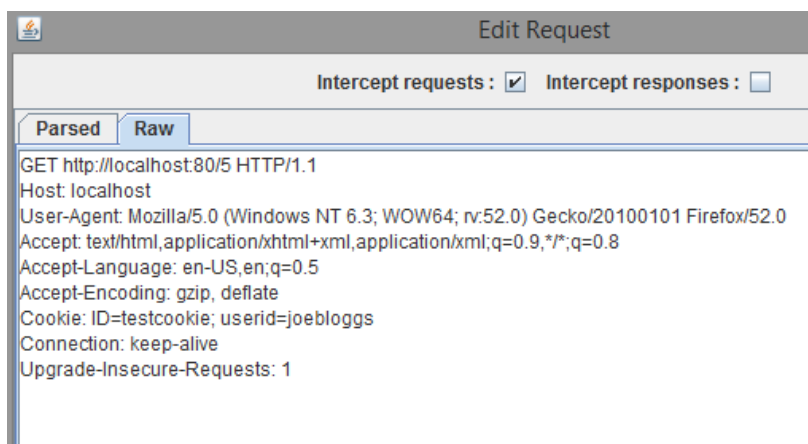


Figure 40 now two cookies are operating in the browser

6

Sessions

In this section I will be looking at the creating a session and carrying out some tests on the session.

I have created a script that creates a session and performs a mathematical function, in this case it gets the power of two, and then gets the power of the result.

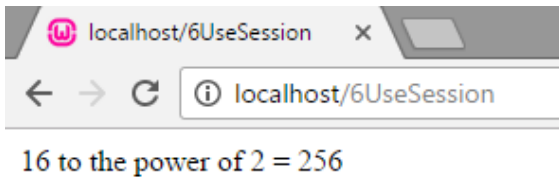


Figure 41 power of 16

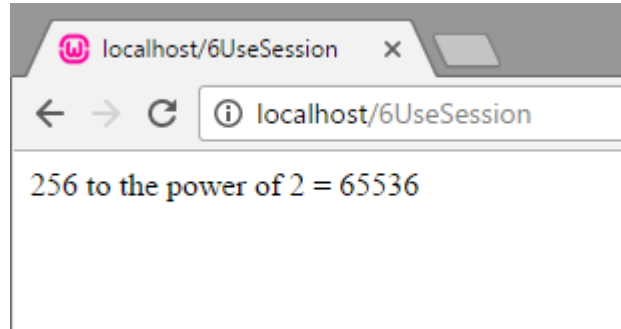


Figure 42 power of 256

Here we can see that the calculation is performed when the page is loaded, then the variable is updated with the result of the last calculation becoming the variable for the next calculation. This occurs every time the page is refreshed and shows that our session exists until the browser is closed or it times out.

Session ID / Hijacking

I managed to access the session id variable using web scarab and by starting the session with the file 6UseSession.php.

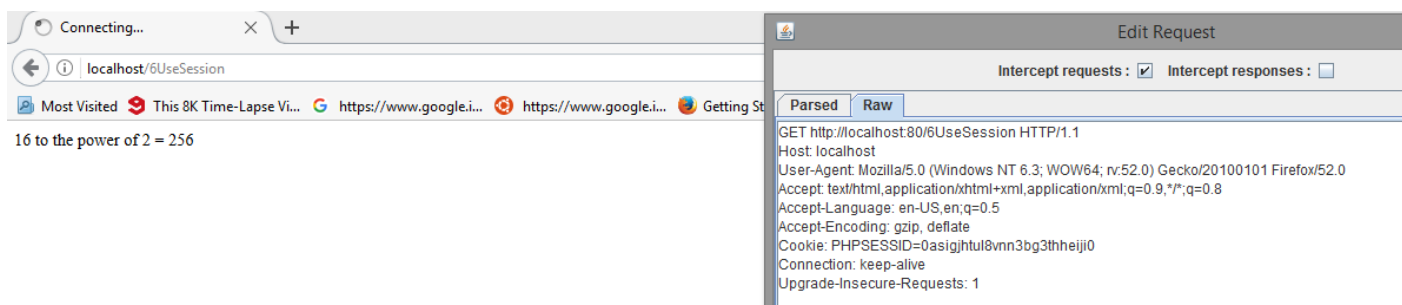


Figure 43 localhost session in web scarab

Hijacking

Now using web scarab I can find and manipulate the cookie field for the browser, for this test I will be using a Firefox and Chrome browser and will be attempting to swap sessions, I will do this by getting the session ID of both sessions and then copying them and pasting them in each other's cookie fields using web scarab, the Firefox browser will have a large calculation while the chrome browser will have a relatively small calculation, if it works the calculations should have swapped browsers.

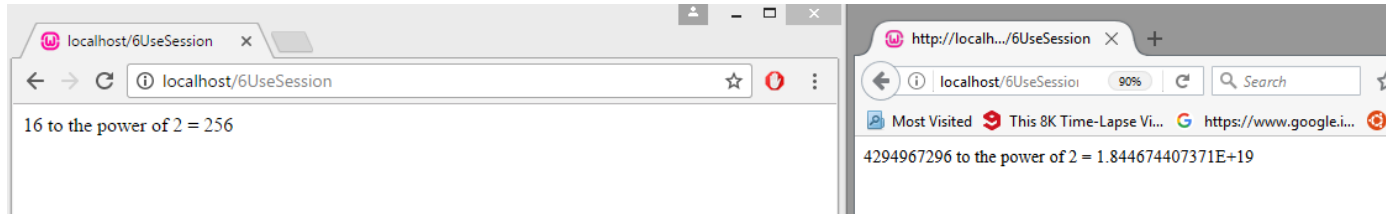


Figure 44 Chrome on the left, Firefox on the right

Now using web scarab I will intercept the cookie when I refresh both pages and then I will swap their session IDs and see if the calculations swap browsers.

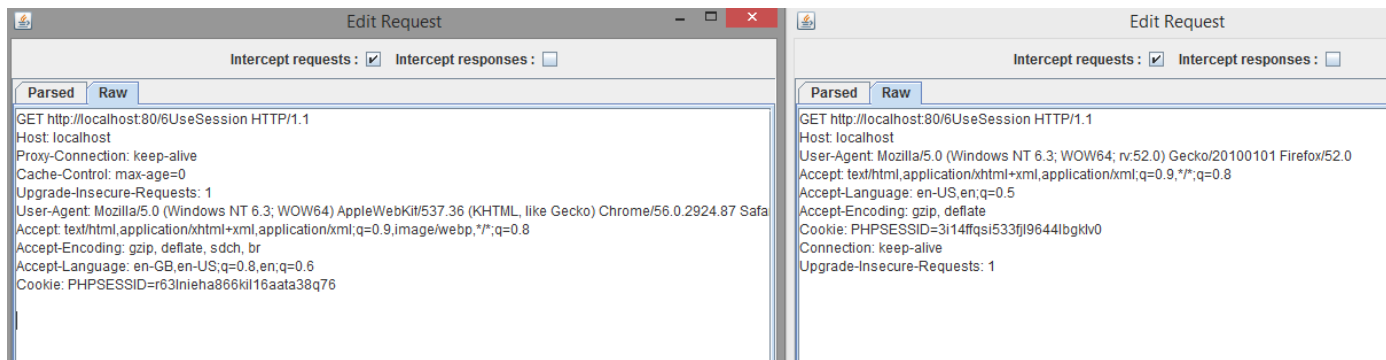


Figure 45 Chrome on the left, Firefox on the right

Now to swap the session IDs and see if the sessions migrate.

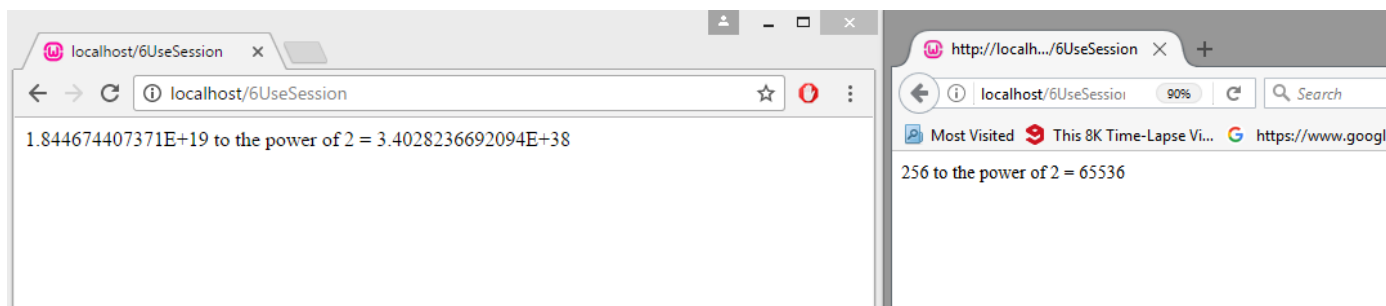


Figure 46 Chrome on the left, Firefox on the right

The session IDs have been swapped and the sessions have been swapped with Chrome now having the large calculation and Firefox having the small calculation. This shows how easy it may be to hijack session IDs and potentially modify the end user's browser.

Bibliography

<http://www.diffen.com/difference/GET-vs-POST-HTTP-Requests> Date Accessed: 29/03/2017, GET vs POST comparison.

https://www.w3schools.com/html/html_forms.asp Date Accessed: 30/03/17, written by W3.CSS, HTML Forms.

https://www.w3schools.com/php/func_string_strip_tags.asp Date Accessed: 30/03/17, written by W3.CSS, Strip Tags.