PHP
PHP Security

Primary source: https://www.w3schools.com/php/

Internet Programming 2, Lesson 8
May 2022

PHP Ord and Chr functions
PHP Security

1

```php
<?php
echo('Below is the demo for the ord() function:<br>');
echo "ord('A')= ".ord("A")."<br>";
echo "ord(a)=".ord("a")."<br>";
echo "ord(Z)= ".ord("Z")."<br>";
echo "ord(z)=".ord("z")."<br>";
echo "ord(0)= ".ord("0")."<br>";
echo "ord(9)= ".ord("9")."<br><br>";
echo('Below is the demo for the Chr() function:<br>');
echo "chr(65)= ".chr(65)."<br>";
echo "chr(97)=".chr(97)."<br>";
echo "chr(90)= ".chr(90)."<br>";
echo "chr(122)=".chr(122)."<br>";
echo "chr(48)= ".chr(48)."<br>";
echo "chr(57)= ".chr(57)."<br><br>";
```

Ord and function

Below is the demo for the ord() function:
ord('A')= 65
ord(a)=97
ord(Z)= 90
ord(z)=122
ord(0)= 48
ord(9)= 57

Below is the demo for the Chr() function:
chr(65)= A
chr(97)=a
chr(90)= Z
chr(122)=z
chr(48)= 0
chr(57)= 9

encrypt.php

In this slide deck, we cover:
Avoid $_SERVER["PHP_SELF"] Exploits :

```php
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

Retaining form data in the event of an error:

```php
Surname: <input type="text" name="surname" value="<?php echo $surname;?>">
```

2

1

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Note on PHP Form Security
The $_SERVER["PHP_SELF"] variable can be used by hackers!
If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

3

Assume we have the following form in a page named "test_form.php":

```php
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```php
<form method="post" action="test_form.php">
```

However, consider that a user enters the following URL in the address bar:
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E

In this case, the above code will be translated to:
`<form method="post" action="test_form.php/"><script>alert('hacked')</script>`

This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.
Be aware of that **any JavaScript code can be added inside the <script> tag!** A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

4

How To Avoid $_SERVER["PHP_SELF"] Exploits?
$_SERVER["PHP_SELF"] exploits can be avoided by using the **htmlspecialchars()** function.
The form code should look like this:
```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

The **htmlspecialchars()** function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post"
      action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

The exploit attempt fails, and no harm is done!

5

Validate Form Data With PHP
The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.
When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.
We will also do two more things when the user submits the form:
1.Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP **trim()** function)
2.Remove backslashes (\) from the user input data (with the PHP **stripslashes()** function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).
We will name the function test_input().
Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

6

Example
```php
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Notice that at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.
The next step is to make input fields required and create error messages if needed.

7

Checking for empty required fields and displaying an error message

```html
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  First Name(s): <input type="text" name="firstName" >
  <span class="error">* <?php echo $firstNameErr;?></span>
  <br><br>
  Surname: <input type="text" name="surname" >
  <span class="error">* <?php echo $surnameErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $firstName;
echo "<br>";
echo $surname;
echo "<br>";
?>
```

8

```php
<?php
// define variables and set to empty values
$firstNameErr = $surnameErr = "";              // initialize to empty string so when script runs first time,
                                               // nothing will be displayed – form code on next page

$firstName = $surname = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["firstName"])) {            // check if field has a value
    $firstNameErr = "Name is required";        // if false, set an error message – will be displayed next to input
                                               // box on form –code on previous form page
  } else {
    $firstName = test_input($_POST["firstName"]); // if it exists, assign it to a variable for further processing
          }
  if (empty($_POST["surname"])) {
    $surnameErr = "Name is required";
  } else {
    $surname = test_input($_POST["surname"]);
    }
}
function test_input($data) {
  // as per normal – previous notes
}
?>
```

9

Retaining form data in the event of an error:
- set the value property of input box to echo the value of the variable which will hold its value in the php script -
$firstName and $surname in this case. Note if form is running for first time, this variable will have been set to
the empty string when it was created and initialized – previous page

```html
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  First Name(s): <input type="text" name="firstName" value="<?php echo $firstName;?>">
  <span class="error">* <?php echo $firstNameErr;?></span>
  <br><br>
  Surname: <input type="text" name="surname" value="<?php echo $surname;?>">
  <span class="error">* <?php echo $surnameErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $firstName;
echo "<br>";
echo $surname;
echo "<br>";
?>
```

10

5

Summary

PHP scripts should be in the same file as the form:

```
action="<?php echo $_SERVER["PHP_SELF"];?>"
```

Execute the PHP Script only if the form was submitted:

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
      // script code
}
```

Check that mandatory fields contain data: set error message to display next to form input box

```
if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = $_POST["name"];
}
```

Avoid $_SERVER["PHP_SELF"] Exploits :

```
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```

Retaining form data in the event of an error:

```
Surname: <input type="text" name="surname" value="<?php echo $surname;?>">
```

11