Create Safe Code - SQL Injection

We gaan een eenvoudige login maken en we gaan zien hoe je een eenvoudige login kunt omzeilen. Hiervoor gaan we SQL injection gebruiken. De opdracht is om deze login te maken en dan zodanig aan te passen dat de login veilig is en niet meer kan wordnen omzeild.

Maak een database met usernaem en wachtwoord en maak een eenvoudige login.

De database kan worden aangemaakt met script dat je hier kunt downloaden.

form.html

Maak een html pagina met een login form. Het form begint met:

`<form class="" action="login.php" method="post">` en heeft twee invoer velden een user en wachtwoord.

Maak het form verder af.

Database

De php code is hieronder toegevoegd. De login maakt gebruik van een een include. De code van de include waarin een verbinding met de database wordt gemaakt staat hieronder ook. let op dat je nog wel de database parameters moet invullen.

login.php

```
<?php


include "includes/db.php";


echo "User:".$_POST['user']."<br>";
echo "Wachtwoord:".$_POST['wachtwoord']."<br>";


$user=$_POST['user'];
$wachtwoord=$_POST['wachtwoord'];
```

```
$myConn = new DB;
$query = "SELECT * FROM login where username='$user' and wachtwoord='$wachtwoord'";
$result = $myConn->executeSQL($query);

if ( $result != 0 ) { // voor deze opdracht moet je deze regel niet veranderen
    echo "<br> Login as $user <br>";
} else {
    echo "<br> Invalid login! <br>";
}

?>
```

db.php

```
<?php

// Define DB Params
define("DB_HOST", "localhost");
define("DB_USER", "");
define("DB_PASS", "");
define("DB_NAME", "login");

class DB{
    protected $dbh;
    protected $stmt;
    protected $resultSet;

    public function __construct(){
        $this->dbh = new PDO("mysql:host=".DB_HOST.";dbname=".DB_NAME, DB_USER, DB_PASS);
        $this->resultSet=[];
    }

    public function executeSQL($query){
        $this->stmt = $this->dbh->prepare($query);
        $result = $this->stmt->execute();
        if (! $result) {
            die('<pre>Oops, Error execute query '.$query.'</pre><br><pre>'.'Result code: '.$result.'</pre>');
        }
        $this->resultSet = $this->stmt->fetchAll(PDO::FETCH_ASSOC);
        return count($this->resultSet);
```

```
}

    public function getRow(){
if (count($this->resultSet) >0 ){
            return array_shift($this->resultSet);
        } else {
            return 0;
        }
    }



}

$DB = new DB;

?>
```

Test of de login werkt.

Voer dan als wachtwoord `' OR '1'='1` in. Kijk wat er gebeurt en en leg uit wat er gebeurt. Hoer heet deze techniek?

Opdracht 1 (P1W1)

Maak de code zodanig veilig dat het truuckje wat hier boven is beschreven niet meer werkt.

Opdracht 2 (P2W2)

1. Breidt de code uit zodat je via de browser een gebruiker en wachtwoord kan toevoegen in de database.
2. Pas de code die je zojuist hebt gemaakt aan zodat de wachtwoorden encypted worden opgeslagen.
3. Pas de code verder zodanig aan dat de controle op het invoeren van een juist wachtwoord plaatsvind op encrypted wachtwoorden.

Opdracht 3 (P2W2)

Voer in de login bij username het volgende in:

```
<script>alert('you are hacked!')</script>
```

DIt sctipt is verder niet gevaarlijk maar je kunt je voorstellen dat je op deze manier wel een script kunt 'injecteren' dat gevaarlijk is. Pas de code aan zodat je beveiligd bent tegen dit soort aanvallen.

Opdracht 4 (P2W2)

Kun je een algemene beschrijving maken van hoe je er voor zorgt dat je via een form (user input) de kans op hacken zo klein mogelijk maakt. Maak deze beschrijving in één of een paar regels.

--

Revision #10
Created 7 November 2019 18:42:35 by Admin
Updated 18 November 2019 21:25:56 by Admin