

Uitwerking OWASP 1 - SQL Injection

Opdracht 1: Wat is SQL?

SQL staat voor Structured Query Language en is een data manipulatie taal.

Opdracht 2: Maak de code snippets compleet door de todo's uit te werken.

index.php

```
<html>
<head>
<title>My pretty page</title>
</head>

<body>
<form action="login.php" method="post">
<input type="text" name="username" >
<input type="password" name="password" >
<input type="submit" name="submit" value="Login">
</form>
</body>
</html>
```

login.php

Let op! Hieronder zie je de content van login.php. De todo's hierin zijn al uitgewerkt.

```
<?php

include "db.php";

// todo 1: print de ingevoerde username en wachtwoord op aparte regels. Geef duidelijk aan welk van de twee
```

de username en het wachtwoord is.

```
echo "User: ".$_POST['username']."<br>";  
echo "Wachtwoord: ".$_POST['password']."<br>";
```

// todo 2: Maak twee variabele aan en sla de ingevoerde username en wachtwoord op in deze variabelen.

```
$username=$_POST['username'];  
$password=$_POST['password'];
```

```
$myConn = new DB;  
$query = "SELECT * FROM user WHERE username='$username'";
```

```
$result = $myConn->executeSQL($query);
```

// todo 3: vermeldt wat de datatype van variabele \$result is. Dit kun je met behulp van een ingebouwde php functie doen.

```
echo "datatype of the result variable is " . gettype($result);
```

```
if (!empty($result)) {  
    echo "<br> Login as $username <br>";
```

// todo 4: let uit wat de ingebouwde php functie print_r() doet en gebruik het om de result-variabele te printen.

```
// onderstaande regel print de raw array  
print_r($result);  
} else {  
    echo "<br> Invalid login! <br>";  
}
```

```
?>
```

db.php

De db.php file bevat een database (DB) class. Deze zorgt voor een database connectie. Neem de onderstaande code-snipper over in je db.php file en werk de todo's uit.

db.php

```
<?php
```

```
// Define DB Params
```

```

// todo 1: zoek uit wat de host, user, password van je database en vul ze hieronder in om de connectie te kunnen
// maken met je db
define("DB_HOST", "localhost");
define("DB_USER", "root");
define("DB_PASS", "");

// todo 2: vul de naam van de database in op de plek van de empty string.
define("DB_NAME", "veilig_programmeren");

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>');
        }

        $row = $this->stmt->fetchAll();

        if(!empty($row)){
            $this->resultSet = $row;
            return $this->resultSet;
        }

        return $this->resultSet;
    }
}

?>

```

Opdracht 3: Als je ervoor hebt kunnen zorgen dat de juiste data print, is het zover! Voer de volgende regel in voor de username, en leg uit wat je te zien krijgt: `' OR '1'='1 .`

Wanneer de bovenstaande SQL snippet wordt ingevoerd, krijg ik de volledige *user*-tabel (met data) te zien.

Opdracht 4: Wat is de naam van de "techniek" die je in opdracht 3 hebt toegepast?

SQL Injection (NL: SQL injectie).

Opdracht 5: Hoe zou je ervoor kunnen zorgen dat de regel uit opdracht 3 niet meer kan gebeuren?

Er zijn meerdere manieren om SQL injectie te voorkomen:

- Deze stap voorkomt SQL injecties niet perse, maar het is belangrijk om field validation toe te passen voor een input field. Zo "forceer" je de gebruiker om een waarde voor bijvoorbeeld gebruikersnaam/wachtwoord in te vullen.
- Je zou Regular Expressions (Regex) kunnen gebruiken om bepaalde karakters buiten te sluiten. Zo zou je ervoor kunnen zorgen dat een bepaalde input alleen letters en numerieke waarden kan bevatten.
- Wat je ook zou kunnen doen is het vervangen van tekens. Zo zou je ervoor kunnen zorgen dat SQL statements niet uitgevoerd kunnen worden.

Revision #3

Created 9 March 2020 17:43:17

Updated 14 April 2020 11:08:11