Veilig programmeren

Veilig programmeren lessen voor 2e jaars AO leerlingen

- Lesplan
- LEGENDA
- OWASP 1 - SQL Injection
- OWASP 7 - Cross Site Scripting (XSS)
- Uitwerking OWASP 1 - SQL Injection
- Uitwerking OWASP 7 - CROSS SITE SCRIPTING (XSS)
- Toetsvoorbereiding week 6

Lesplan

Introductie veilig programmeren

In de veilig programmeren lessen wordt kennis opgedaan over Unified Modeling Language (UML), OWASP, SANS, Sessions, Logging van user events. Van de leerlingen wordt verwacht dat zij begrijpen dat de veiligheid van de code cruciaal is en hoe ze de veiligheid kunnen implementeren en waarborgen.

Het ingeleverde huiswerk wordt aan het begin van de les behandeld en de leerlingen krijgen de ruimte om aan te geven waar se tegenaan zijn gelopen en wat ze lastig vonden. Voor het oplossen van het probleem wordt de leerling gevraagd wat hij/zij zelf al heeft gevonden met betrekking tot het probleem en wat een mogelijke oplossing ervoor zou kunnen zijn.

Na het bespreken van het huiswerk wordt een nieuw stukje theorie behandeld, en krijgen de leerlingen de gelegenheid om aan de lesopdrachten **[1]** en huiswerk te werken.

> Het kan zijn dat er een begrippenlijst is toegevoegd aan het einde van de pagina. Deze lijst bevat begrippen die je kunnen helpen bij het begrijpen van de lesstof.
>
> Zie je toch een onbekend begrip? Geef het aan bij de docent!

> **LET OP!!** Neem de LEGENDE door om de pagina's optimaal te gebruiken.

Studiewijzer

Week	Doel	Beschrijving
1	In deze les hebben de leerlingen uitgezocht wat er is gebeurd met Citrix en hoe dit is gebeurd. Hierbij werd van de leerling verwacht dat hij/zij de gestelde vragen in eigen woorden kon uitleggen.	

2	De leerling heeft kennis opgedaan over OWASP 1 - SQL Injectie. Tijdens deze les is er een database genaamd veilig programmeren gemaakt. Deze database bevat een *user* table met daarin een username en een password. Zodra de gegeven SQL statement gebruikt voor je sql indectie, toont deze de data zoals deze opgeslagen is in de database tabel.	
3	De leerling heeft kennis opgedaan over OWASP 7 - Cross Site Scripting (XSS). Tijdens de les is er gewerkt aan de pagina met een input field die een zoekterm accepteert, met daarnaast een zoekknop. Zodra de zoekknop werd ingedrukt, zorgt de code ervoor dat er een tekst verscheen m.b.t. de geschreven zoekterm. Als zoekterm is een stukje JavaScript code ingevoerd. Deze heeft ervoor gezorgd dat de tekst *Er zijn geen resultaten gevonden voor de ingevoerde opdracht* wordt getoond en dat de kleur van de letters zijn veranderd naar blauw.	• Bespreken huiswerk week 2
4	-	
5	De leerling is geïntroduceerd aan OWASP 2 - Broken authentication. Tijdens deze les zijn de voorgaande lessen nogmaals samengevat ter voorbereiding op de toets.	• Herhaling huiswerk week 2 en 3
6	Toetsing opgedane kennis van voorgaande weken.Toets wordt besproken en eventuele vragen beantwoord.	
7		• espreken huiswerk week 6
8	Eindtoets	• Bespreken huiswerk week 7

LEGENDA

Deze pagina dient als een legenda bij het begrijpen van de lessen. Hieronder zie je een overzicht van mogelijkheden:

Belangrijke informatie:

> Belangrijke informatie kun je vinden in deze blauwe ballon.

Lesopdrachten:

> Lesopdrachten moeten uitgewerkt en ingeleverd worden bij de docent en kun je herkennen aan deze oranje ballon.

Inline code:
Wanneer er in een zin een stukje code in een regel (inline) wordt getoond, kun je het herkennen aan de inline code ballon: `# zo ziet een single line comment eruit in Python`

Multi-line code:
Wanneer code **[B1]** wordt gebruikt, kun je deze herkennen aan de grijze ballon met een getal langs de kantlijn. Bijvoorbeeld:

```
"""
Python support multi line codes middels drie aanhalingstekens
"""
```

Begrippen:
Het kan zijn dat er woorden worden gebruikt die mogelijk extra uitleg nodig hebben. Deze worden aangeduid met een rechte haak, gevolgd door een B en dan een nummer. Het nummer bepaalt de volgorde van begrippen. Een voorbeeld hiervan is hierboven te zien, naast het woord "code". Dit is het eerste begrip, daarom zie je een 1 naar de B.
De uitleg van "code" kun je op dezelfde pagina onder titel "Begrippen" terug vinden:

Begrippen

[B1] Code = betekenis begrip

Bronnen:

Bronnen kun je herkennen aan getallen tussen rechte haken (**[1]**). Deze verwijzen naar bronnen in de bronnenlijst. De bronnenlijst is te vinden aan het einde van de pagina, onder het kopje 'Bronnen'.

Bronnen

[1] www.google.com (dit is een slecht voorbeeld!)

OWASP 1 - SQL Injection

Veilig programmeren

Veilig programmeren is een belangrijk onderdeel van Software Development, zie afbeelding:

momdropstable.png

SQL Injection

In deze les gaan we een login form maken met minimale functionaliteit. We gaan kennis maken met SQL Injection en gaan op zoek naar manieren om dit te voorkomen. Let's code!

Benodigdheden voor deze les

- XAMPP
- Sublime Text
- PhpMyAdmin DB
- Browser

> **Opdracht 1**: Wat is SQL?

Step-by-step implementatie login form met SQL injectie

Tijdens deze les gaan we een database opzetten in PhpMyAdmin. Om dit te kunnen doen gaan we eerst XAMPP **[1]** downloaden zodat we php kunnen gebruiken, want we hebben e

Voor deze opdracht gaan we HTML, PHP en MySQL gebruiken. Om deze tools te gebruiken, hebben we een server nodig. Daarom beginnen we met het downloaden van XAMPP **[1]**. Als je XAMPP eenmaal hebt gedownload, open je XAMPP en start je de Apache - en SQL server. Nu kunnen we beginnen met het opzetten van onze database in PhpMyAdmin.

PhpMyAdmin

Open je browser en navigeer naar http://127.0.0.1/phpmyadmin/ . Maak een nieuwe database aan met daarin een tabel *user*. Dit tabel heeft 3 kolommen, namelijk:

- id
- username
- password

Insert op minstens 2 entries in tabel user.

Code

Nu onze servers eenmaal draaien, kunnen we aan de slag! Open de xampp-folder in je document browser en open de folder *htdocs*. Maak een nieuwe folder genaamd *veilig programmeren SQL injecties* aan in folder htdocs.

Open de Sublime Text editor en maak de onderstaande drie files aan en sla deze op in folder *veilig programmeren SQL injeties*:

1. index.html
2. login.php
3. db.php

De onderstaande secties leggen uit wat er wordt verwacht van je code.

index.html

De index.html file is, zoals de extensie van de file al aangeeft, een HTML document. Zorg ervoor dat dit document de HTML-skeleton bevat. De body tag van dit document moet een form bevatten, bestaande uit twee input-fields en een button.

Het form moet gebruik maken van de login.php file.

Let op! Zorg ervoor dat het wachtwoord veld het ingevoerde wachtwoord niet toont!

Als je de HTML-skeleton en form af hebt, zou je deze moeten kunnen zien in je browser. Dat doe je zo:

1. Open je index.html file
2. Click met je rechter muisknop in de file
3. Selecteer "Open in browser"

Merk op dat de zoekbalk een file-path toont. Klik in je zoekbalk en vervang het stukje `file:///C:/xampp/htdocs` met `localhost`. Deze stap is nodig om PHP te kunnen gebruiken.

login.php

De login file is een php document. Hieronder is een code-snippet toegevoegd. Neem deze over in je eigen login.php-file en werk de todo's uit:

```
<?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.

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

$myConn = new DB;

// todo 3: include de variabele met de username in de onderstaande query, zodat deze alle data kan ophalen voor de ingevoerde username.
$query = "SELECT * FROM user WHERE ";

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

// todo 4: vermeldt wat de datatype van variabele $result is. Dit kun je met behulp van een ingebouwde php functie doen.

if (!empty($result)) {
    echo "<br> Login as $username <br>";
    // todo 5: let uit wat de ingebouwde php functie print_r() doet en gebruik het om de result-variabele te printen.
} 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-snippet over in je db.php file en werk de todo's uit.

```
<?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", "");
define("DB_USER", "");
define("DB_PASS", "");

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

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;
□}
}

?>
```

Moment of truth

In het begin heb je in de *user* tabel (van je database) twee gebruikers aangemaakt. Voer de username en wachtwoord van deze gebruiker in en ga na of je de juiste gebruikers data wordt geprint op de pagina.

> **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`.

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

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

Bronnen

[1] XAMPP: https://www.apachefriends.org/index.html

OWASP 7 - Cross Site Scripting (XSS)

Cross-site scripting (XSS) is een fout in de beveilig van websites. Het zorgt ervoor dat websites - die normaliter wel betrouwbaar zouden zijn - onbetrouwbaar worden. Dit komt omdat de website XSS injectie toestaat.

Wat kun je met XSS injecties bereiken?

Het doel van XSS injecties is om permanente verandering aan te brengen aan een website. Een persoon met verkeerde intenties zou JavaScript code kunnen invoeren in een input field, waardoor deze mogelijk in een database tabel wordt opgeslagen. De opgeslagen JavaScript code zou uitgevoerd kunnen worden als de data uit de database tabel bijvoorbeeld gebruikt zou worden om data te tonen op de website. Zo zou de ontwikkelaar van XSS injecties ervoor kunnen zorgen dat de bezoeker van een website ge-redirect wordt naar een andere website.

XSS injectie opdracht

In deze les gaan we aan de hand van een simpele implementatie XSS injectie demonstreren. Let's start coding!

In deze les gaan we een input field binnen een form maken. Als het form af is gaan we met behulp van JavaScript wijzigingen aanbrengen in onze interface. Voor we XSS injectie kunnen toepassen, gaan we eerst het form maken. Dit doen we aan de hand van de volgende stappen:

1. Open de folder waar XAMPP in is opgeslagen en ga naar de htdocs -older. Maak een nieuwe map aan in deze en noem het XSS.
2. Maak een index.php file in de XSS-folder
3. Zorg ervoor dat de index.php file een HTML-skeleton bevat.
4. Maak een form aan in de body tag van index.php. Zorg ervoor dat dit form de index.php file aanroept.
5. Maak binnen het form een input field aan met een placeholder. De waarde hiervan stel je gelijk aan *Zoekopdracht*.
6. Zorg ervoor dat er een knop is naast de input field. We waarde van deze knop moet gelijk zijn aan *Zoek*.

7. Schrijf embedded php code in index.php. Deze code moet uitgevoerd worden zodra de gebruiker op de *zoek* knop drukt. Zorg ervoor dat je php code m.b.v. de isset functie checkt of er een waarde is ingevoerd in het input field. Als de check *true* is, zorg je ervoor dat je de volgende text print:

   De zoekopdracht is: $zoekopdracht
   Geen resultaat gevonden!

> **Opdracht 1:** Wat wordt er op de pagina getoond als je de volgende text invoert:
>
> Coole website <script>alert("XSS voorbeeld")</script>?

> **Opdracht 2:** Wat gebeurd er als je <font color="blue"> invoert?

> **Opdracht 3:** Hoe kun je XSS injecties voorkomen?

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-snippet 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.

Uitwerking OWASP 7 - CROSS SITE SCRIPTING (XSS)

Cross-site scripting (XSS) is een fout in de beveilig van websites. In deze les gaan we een input field binnen een form maken. Als het form af is gaan we met behulp van JavaScript wijzigingen aanbrengen in onze interface. Voor we XSS injectie kunnen toepassen, gaan we eerst het form maken. Dit doen we aan de hand van de volgende stappen:

1. Open de folder waar XAMPP in is opgeslagen en ga naar de htdocs -older. Maak een nieuwe map aan in deze en noem het XSS.
2. Maak een index.php file in de XSS-folder
3. Zorg ervoor dat de index.php file een HTML-skeleton bevat.
4. Maak een form aan in de body tag van index.php. Zorg ervoor dat dit form de index.php file aanroept.
5. Maak binnen het form een input field aan met een placeholder. De waarde hiervan stel je gelijk aan *Zoekopdracht*.
6. Zorg ervoor dat er een knop is naast de input field. We waarde van deze knop moet gelijk zijn aan *Zoek*.
7. Schrijf embedded php code in index.php. Deze code moet uitgevoerd worden zodra de gebruiker op de *zoek* knop drukt. Zorg ervoor dat je php code m.b.v. de isset functie checkt of er een waarde is ingevoerd in het input field. Als de check *true* is, zorg je ervoor dat je de volgende text print:

Ingevoerde zoekterm: $zoekopdracht
Er zijn geen resultaten gevonden voor de ingevoerde zoekopdracht

Uitwerking bovenstaande stappen

```
<!DOCTYPE html>
<html>
<head>
<title>Mooie demonstration</title>
</head>
<body>
<h1>Voer je zoekopdracht hieronder in</h1>
```

```
<form action="index.php" method="get">
    <input type="text" name="search" placeholder="Zoekterm invoeren " />
    <input type="submit" name="submit" value="Zoek">
</form>

<?php
    if (isset($_GET["search"])) {
        echo "Ingevoerde zoekterm: " . $_GET["search"] . "<br/>";
        echo "Er zijn geen resultaten gevonden voor de ingevoerde zoekopdracht!";
    }
?>
</body>
</html>
```

> **Opdracht 1:** Wat wordt er op de pagina getoond als je de volgende text invoert:
>
> Coole website <script>alert("XSS voorbeeld")</script>?

De pagina toont de text *Ingevoerde zoekterm: Coole website*. De script tags en de JavaScript functie tonen niet. Deze zorgen er wel voor dat er een pop-up box met de text *XSS voorbeeld* verschijnt.

> **Opdracht 2:** Wat gebeurd er als je <font color="blue"> invoert?

De text *Er zijn geen resultaten gevonden voor de ingevoerde zoekopdracht* kleurt blauw.

> **Opdracht 3:** Hoe kun je XSS injecties voorkomen?

XSS injecties kun je voorkomen door de user input te beperken tot een bepaalde tekenreeks. Bijvoorbeeld alleen letters en getallen. Dit resultaat zou je bereiken door Regular Expressions (RegEx) toe te passen.

Toetsvoorbereiding week 6

Tijdens de toets wordt er geen nieuwe kennis getest. Er wordt vanuit gegaan dat de leerling de lesstof tot nu toe eigen heeft gemaakt en kan een opdracht uitwerken. Hieronder staat een lijst van de toegestane middelen, termen waarmee te leerling bekend moet zijn, benodigdheden en de onderwerpen die mogelijk getoetst zullen worden.

Let op! Het is niet mogelijk om deze toets te herkansen.

Toegestane middelen tijdens de toets

- Het gebruik van internet is toegestaan.
- Voor de toets is kennis van HTML, PHP en MySQL vereist.
- Het gebruik van lesmateriaal op roc.ovh, (gemaakte) huiswerk en communicatie middelen is/zijn **niet** toegestaan.

Termen

- Zero-day vulnerability
- Software patch
- IDE
- Regular expressions
- OWASP

Benodigdheden voor de toets

- De leerling is bekend met XAMPP en kan de Apache – en SQL server draaien.
- De leerling is in staat om met behulp van HTML en PHP een simpele webapp te maken.
- De leerling is bekend met PhpMyAdmin database en kan hiermee overweg (zie sectie database voor meer details).

- De leerling is maakt gebruik van een IDE naar keuze, die zowel HTML als PHP ondersteunt.

Database

- De leerling is in staat een database connectie te maken (PDO/mysqli_connect).
- De leerling is in staat om een database en tabellen aan te maken a.d.h.v. de gegeven eisen.
- De leerling is in staat om data toe te voegen aan database tabellen.
- De leerling is in staat om data op te halen uit de database.
- De leerling is in staat om opgehaalde (raw) data te tonen in de browser (print_r).

SQL Injectie

- De leerling weet wat SQL is.
- De leerling kan een SQL injectie demonstreren/simuleren middels.
- De leerling weet hoe SQL injecties verholpen/voorkomen kunnen worden.
- De leerling is in staat een oplossing te implementeren om SQL injecties te voorkomen.

Cross Site Scripting (XSS) injectie

- De leerling weet wat XSS is.
- De leerling kan een XSS injectie demonstreren/simuleren middels.
- De leerling weet hoe XSS injecties verholpen/voorkomen kunnen worden.
- De leerling is in staat een oplossing te implementeren om XSS injecties te voorkomen.