# Technische Universität Braunschweig

Institut für Anwendungssicherheit

## Websicherheit

Prof. Dr. Martin Johns

28. Februar 2019

**Nachname:**

**Vorname:**

**Matrikelnummer:**

**Klausurnummer:** 001

**Studiengang:** ☐ Bachelor ☐ Master ☐ Diplom ☐ Frühstudium ☐ Erasmus

**Fachrichtung:** ☐ Informatik ☐ Wirtschaftsinformatik ☐ Mathematik ☐ Physik ☐ Mobilität und Verkehr ☐ Psychologie ☐ Maschinenbau (☐ Mechatronik) ☐ Elektrotechnik ☐ Wirtschaftsingenieurwesen (☐ Bauing ☐ E-Technik ☐ Maschbau) ☐ Finanz- und Wirtschaftsmathematik ☐ IST ☐ Sonstige: _____

**Versuch der Notenverbesserung:** ☐

Die Bearbeitungszeit beträgt 90 Minuten. Die Klausur besteht aus 3 Aufgaben.

| Aufgabe 1 | Aufgabe 2 | Aufgabe 3 | Summe |
|---|---|---|---|
| 50 | 20 | 30 | $\sum$ 100 |
| | | | $\sum$ |

**Note:**

Hinweise: Die Klausur hat 3 Aufgaben. Zwischen den Teilaufgaben befindet sich Platz für die Lösungen. Bitte stellen Sie sicher, dass Sie keine der Aufgaben übersehen und das keine Blätter fehlen. Die Noten werden direkt über das QIS bekanntgegeben, Sie brauchen sich die Klausurnummer nicht merken.

**Aufgabe 1 (50 Punkte)**

a) Welcher Teil einer URL wird nicht an den Server übertragen? [1 Satz] *(2pt)*

b) Eve versendet Emails, welche auf ihre Webseite verweisen. Ihre Webseite sieht genauso aus, wie die von Bank B und erlaubt, unter Angabe des Passworts, eine Passwortänderung vorzunehmen. In den Emails erinnert Eve die Empfänger daran, regelmäßig das Passwort zu ändern. Nennen Sie die vorliegende Angriffsklasse und Angreifertyp? *(2pt)*

c) Wie definiert die Same-Origin Policy gleiche Origins. Der Internet Explorer muss hierbei nicht betrachtet werden. *(3pt)?*

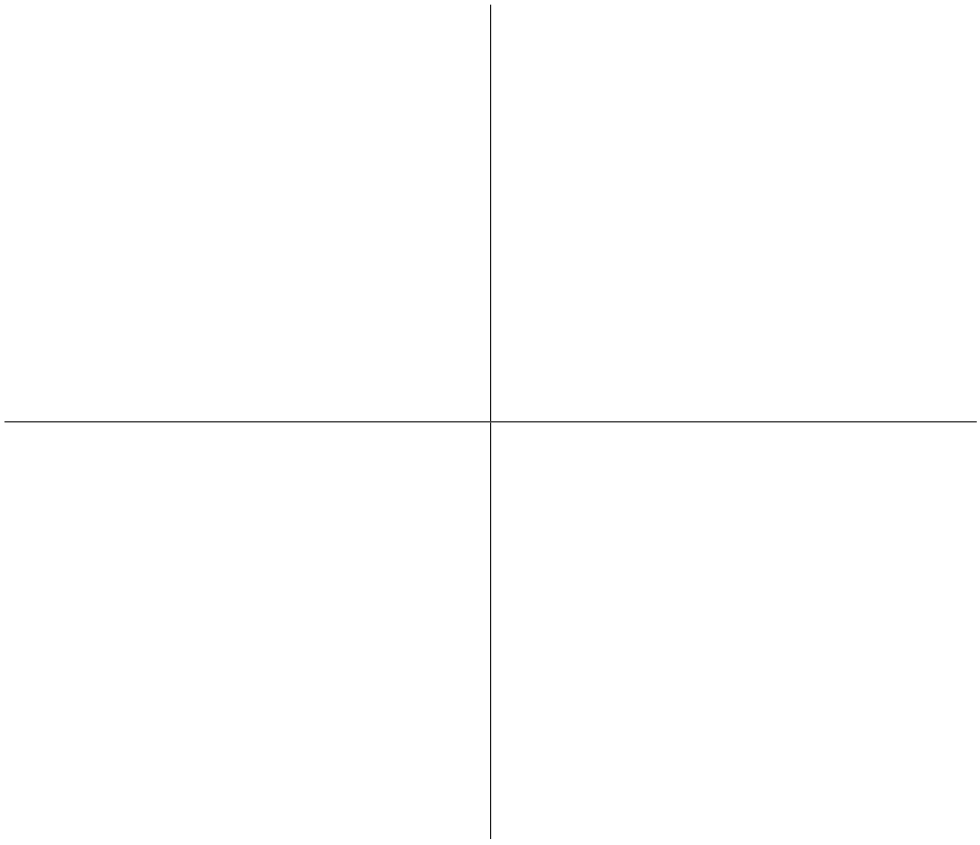
d) Cookies sind der populärste Weg um Authentifizierung in Web Applikationen umzusetzen. Diese Verwendung ist offensichtlich sicherheitskritisch. Wurden Cookies entsprechend mit Sicherheit als Fokus entwickelt? Geben Sie ein Argument, um Ihre Aussage zu begründen. *(5pt)*

e) Wie übertragen HTTP GET and POST Requests Parameter? Lassen sich die beiden Übertragunsmöglichkeiten kombinieren und wenn ja, wie? *(8pt)*

f) Stellt die Content Security Policy einen Schutz vor XSS oder eine Entschärfung von potentiellen XSS Schwachstellen dar? Begründen Sie! *(10pt)*

g) Was bezwecken HTTP Strict Transport Security (HSTS) Header? Welches Problem für die Privatsphäre birgt HSTS? Beschreiben Sie einen solchen Missbrauch. *(10pt)*

h) Was ist der Unterschied zwischen Reflected und Persistent (Stored) Cross-Site-Scripting (XSS)? Gibt es noch eine weitere Dimension von XSS? Wenn ja, erklären Sie auch diese. Beschriften Sie die gegebene Matrix und füllen Sie diese mit entsprechenden minimalen Codebeispielen. *(10pt)*



**Aufgabe 2 (20 Punkte)**

a) Alice betreibt eine eigene kleine Webseite auf einem Server mit nur schwacher Hardware. Entsprechend probiert sie möglichst viele Berechnungen in den Browser des Besucher auszulagern. Unter Anderem hat sie die Generierung von CSRF Tokens in eine JavaScript Funktion ausgelagert. Diese JavaScript Funktion erstellt ein verstecktes Formularfeld in das der Tokenwert eingetragen wird. Zusätzlich wird ein gleichnamiger Cookie mit dem selben Wert erstellt. Beide Werte werden Serverseitig abgeglichen. Hat Alice damit einen funktionierend CSRF Schutz? Begründen Sie Ihre Aussage. Sollte ein Angriff möglich sein beschreiben Sie diesen in kurzen Worten. *(5pt)*

b) Der Server von Alice wird im Betrieb von einem Angreifer übernommen. Zum Zeitpunkt der Übernahme verwendet der Server aktiv eine auf der Festplatte verschlüsselt vorliegende Datenbank. Sind die Daten in der Datenbank vor dem Angreifer sicher? Begründen Sie Ihre Antwort. *(5pt)*

```php
function sanitizer($input) {
   $semicolon = preg_match('/;/',$input);
   $hiphen = preg_match('/\'/',$input);
   $capitals = !preg_match('/[A-Z]/',$input);
   if($hiphen) {
     if($semicolon && !$capitals) {
         $input = preg_replace('/[^a-zA-Z]/','',$input);
     } else {
       $input = preg_replace('/[^a-zA-Z\'; -]/','',$input);
     }
   }
   return $input;
}
```

Abbildung 1: Der PHP SQLi Sanitizer

```php
/* code */
$input = sanitizer($_POST['username']);
$query = "SELECT passwords
          FROM users
          WHERE username = '".$input."' LIMIT 1;";
$results = pg_query($query);
/* code */
```

Abbildung 2: Der Aufruf des SQLi Sanitizer

c) Studieren Sie den in PHP geschrieben SQLi Sanitizer in Figur 1. Definieren Sie einen Wert für $\$\_POST['username']$ der den Sanitizer umgeht und Code in den Query Aufruf in Figur 2 injected. Dies soll zur Folge haben, dass die Tabelle 'users' gelöscht wird. Sie dürfen annehmen, dass neben dem Sanitizer keine weiteren SQLi Schutzvorkehrungen getroffen wurden. *(10pt)*

**Aufgabe 3 (30 Punkte)**

Die Quellcode-Ausschnitte 3 bis 8 definieren eine kleine Webanwendung. Diese Webanwendung enthält vier uns bekannte Sicherheitsschwachstellen von denen drei gefunden werden müssen, um volle Punkte zu erreichen. Die vierte Sicherheitsschwachtelle wird weiter unten als Beispiellösung zur Orientierung gegeben.

Jede gefunde Schwachstelle gibt **3 Punkte für das Aufzeigen und korrekte Kategorisieren**. Weitere **5 Punkte gibt es für die Erklärung, wie die Schwachstelle ausgenutzt werden kann**. Abschließend gibt es **2 Punkte für die Erklärung wie die Schwachstelle behoben werden kann**. Es können somit maximal 10 Punkte pro Schwachstelle erreicht werden.

Lediglich XSS, XSSI, CSRF, SQLi, Execution After Redirect, Command Execution und File Inclusion sind gefragt. Bitte halten Sie ihre Lösung kurz, prägnant und im Stil der Beispiellösung. Lediglich die ersten 3 bearbeiteten eigenständigen Schwachstellen werden mit Punkten belohnt.

a) Beispiel:

*Type* *CSRF*

*File* *deletemusic.php*

*Exploit* *Das Formular von deletemusic.php hat keine CSRF protection (i.e. doublesubmit token). Folglich kann ein Angreifer einen (eingeloggten) Nutzer auf eine kontrollierte Seite locken und im Namen/mit den Rechten des Nutzers Musiktitel vom Server löschen.*

*Fix* *Es sollte ein CSRF token verwendet werden, wie es unter Anderem bei viewmusic.php gemacht wird.*

b) 1. Schwachstelle

Type:

File:

Exploit:

Fix:

c) 2. Schwachstelle

Type:

File:

Exploit:

Fix:

d) 3. Schwachstelle

Type:

File:

Exploit:

Fix:

main.php

```
<html> <head>
  <link rel="stylesheet" type="text/css" href="style.css">
        </head>
<?php
 session_start();
?>
<div id='container'> <div id='left-side-space'></div>
<div id='side-bar'> </div>
<div id='header'> <div class='upper-float'></div>
<div class='inner-float'></div>
<div class='text-content'>
  <?php
    if(isset($_SESSION['login']) &&
       $_SESSION['login'] === 'true') {
      echo 'Hello User ' . preg_replace('/[^a-zA-Z]/i','',$_SESSION['username']) .
      ' (<a href="./logout.php">logout</a>|'.
      '<a href="./main.php?nav=welcome">home</a>)';
    } else {
      echo 'Please log in';
    }
?>
</div> </div>
<div id='right-of-header'></div>
<div id='body'>
<?php
      switch ($_GET['nav']) {
           case "welcome":
                include 'welcome.php';
                break;
           case "deletemusic":
                include 'deletemusic.php';
                break;
           case "viewmusic":
                include 'viewmusic.php';
                break;
           case "login":
                include 'login.php';
                break;
           default:
                if(isset($_GET['nav']))
                        echo "The navigation '" . $_GET['nav'] .
                             "' does not exist";
                else
                        include 'login.php';
    }
?>
</div> </div> </html>
```

Abbildung 3: Die Mainpage die das Applikationsframework stellt.

login.php

```php
<?php
   include('users.php');

   if(isset($_SESSION['login']) &&
      $_SESSION['login'] === 'true') {
      //if user is logged in redirect back to main page
      header('Location: ./main.php?nav=welcome', true, 302);
      die();
   }
   if(isset($_POST['btnsubmit']) &&
      isset($_POST['username']) &&
      isset($_POST['password'])) {
      if($passwords[$_POST['username']] == $_POST['password']) {
         $_SESSION['login'] = 'true';
         $_SESSION['username'] = $_POST['username'];
         //print_r($_SESSION);
         header('Location: ./main.php?nav=welcome', true, 302);
         die();
      } else {
         header('Location: ./main.php?nav=login', true, 302);
         die();
      }
   }
?>
<div class='login-left-space'></div>
<div class='login-top-space'></div>
<form action='./main.php?nav=login' method='POST'>
      <table>
        <tr>
          <td>User:</td>
          <td> <input type='text' name="username"> </td>
        </tr>
        <tr>
          <td>Pwd:</td>
          <td> <input type='text' name="password"> </td>
        </tr>
        <tr>
            <td> <input type='submit' name="btnsubmit" value='submit'> </td>
        </tr>
      </table>
</form>
```

Abbildung 4: Die Funktionalität zum Einloggen in die Applikation.

welcome.php

```php
<?php
  if(!isset($_SESSION['login']) ||
     $_SESSION['login'] === 'false') {
     //if user is not logged in redirect back to main page
     header('Location: ./main.php?nav=login', true, 302);
     die();
  }
?>

<div class='welcome-left-space'></div>
<div class='welcome-top-space'></div>
<table>
  <tr>
    <td>
      <a href='./main.php?nav=viewmusic'>Search our music library</a>
    </td>
  </tr>
  <tr><td></td></tr>
  <tr><td></td></tr>
  <tr>
    <td>
      <a href='./main.php?nav=deletemusic'>Delete music tracks</a>
    </td>
  </tr>
</table>
```

Abbildung 5: Die Homepage der Applikation nach Login.

viewmusic.php

```
<div class='search-left-space'></div>
<div class='search-top-space'></div>

<?php

$old_double = -1;
if(isset($_COOKIE['doublesubmit'])) {
  $old_double = $_COOKIE['doublesubmit'];
}

$double = random_int(1000000,9999999);
setcookie('doublesubmit', $double);

if(!isset($_SESSION['login']) ||
   $_SESSION['login'] === 'false') {
   // if user is not logged in redirect back to main page
   header('Location: ./main.php?nav=login', true, 302);
   die();
}

?>

<form action='./main.php?nav=viewmusic' method='post'>
      <label> Track Title: </label>
      <input type='text' name='search'>
      <?php
        echo '<input type="hidden" value="'.
             $double . '" name="doublesubmit">';
      ?>
      <input type='submit' name='submit'>
</form>

<div class='search-results'>
<?php
  if(isset($_POST['search']) &&
     $old_double != -1 &&
     $old_double === $_POST['doublesubmit']) {
     $results = trim(shell_exec("ls -a *" . $_POST['search'] . "*.mp4 | sort"));
  foreach( explode("\n", $results) as $result) {
    echo "<div class='search-element'>";
       echo "Title: " . $result . "<br>";
       echo "</div>";
       echo "<div class='search-buff'></div>";
  }
}
?>
</div>
```

Abbildung 6: Die Funktionalität zum Durchsuchen vorhandener Musiktitel.

deletemusic.php

```php
<div class='search-left-space'></div>
<div class='search-top-space'></div>

<?php

if(!isset($_SESSION['login']) ||
   $_SESSION['login'] === 'false') {
   header('Location: ./main.php?nav=login', true, 302);
}

?>


<form action='./main.php' method='get'>
      <label>Delete Title: </label>
      <input type='text' name='delete'>
      <input type='hidden' name='nav' value='deletemusic'>
      <input type='submit' name='submit'>
</form>


<?php
if(isset($_GET['delete'])) {
  $name = preg_replace('[^a-zA-Z]', '', $_GET['delete']);
  shell_exec("rm " . $name . ".mp4");
}
?>
```

Abbildung 7: Die Funktionaliät zum Löschen von Musiktiteln.

users.php

```php
<?php

$passwords['admin'] = 'easy';

?>
```

Abbildung 8: Die Datei, in der die Nutzerdaten aufbewahrt werden.

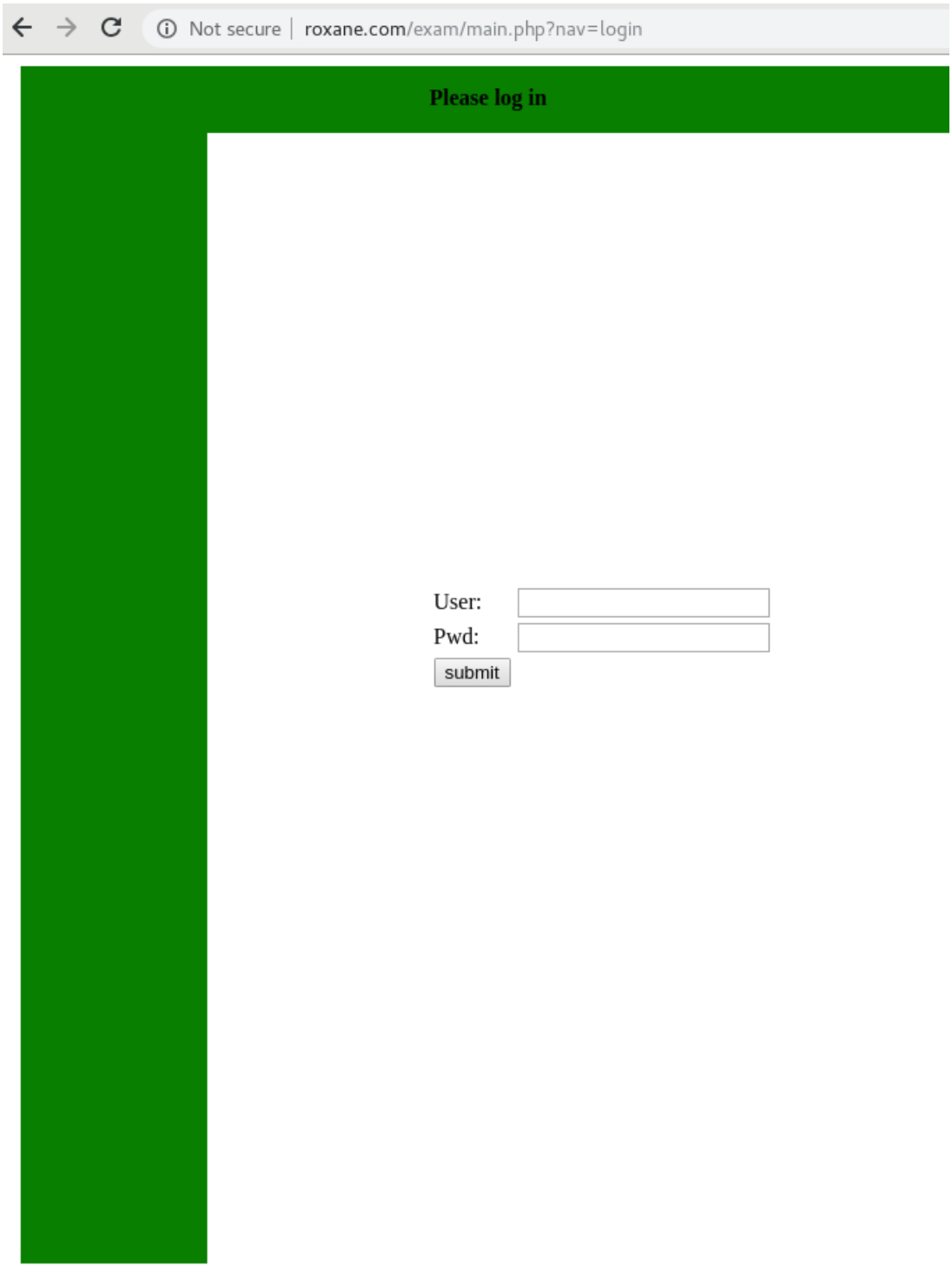



Abbildung 9: Dies ist die Login Seite

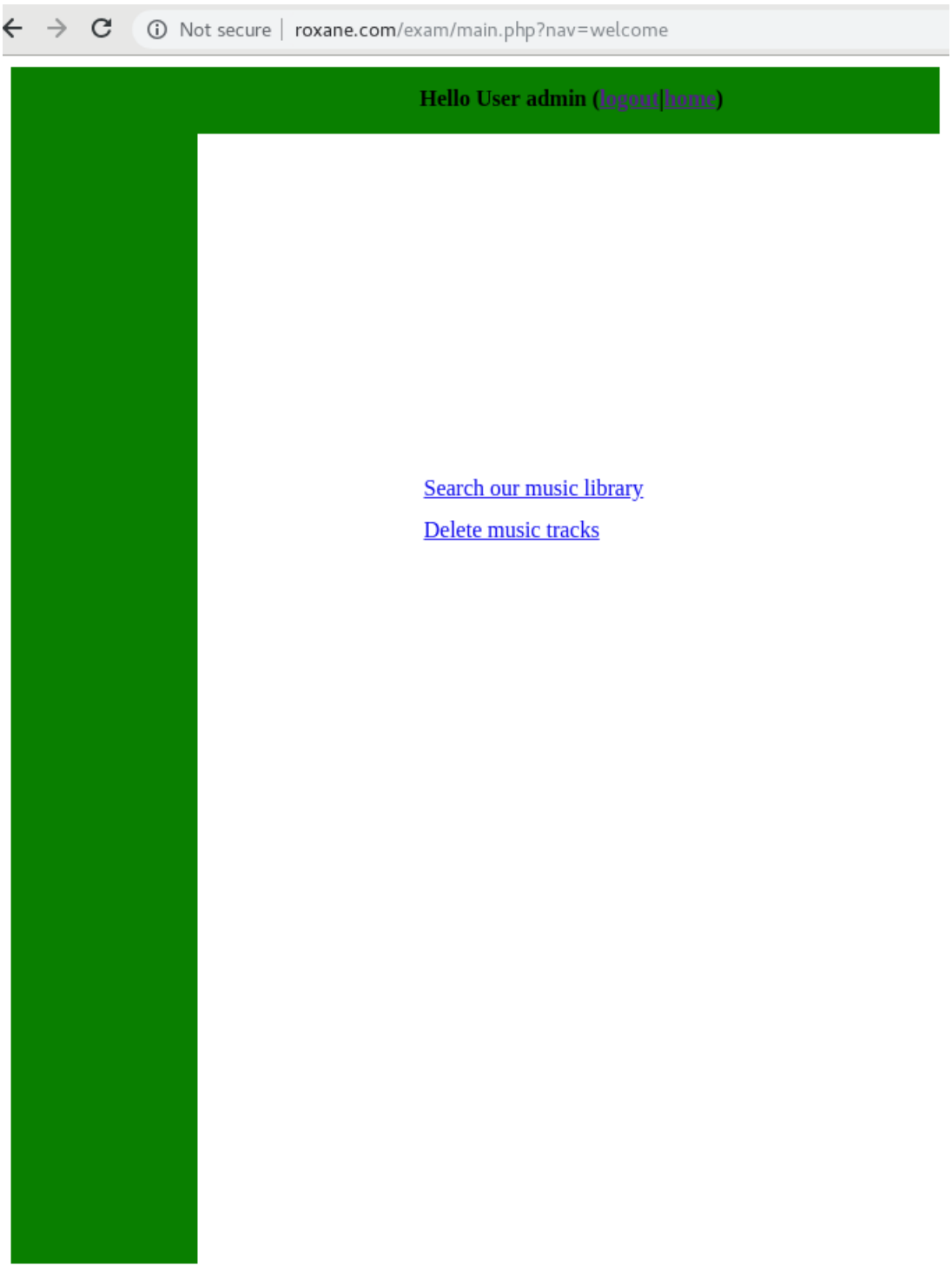



Abbildung 10: Dies ist die Home Seite


Abbildung 11: Dies ist die Seite zum durchsuchen der Musiktracks

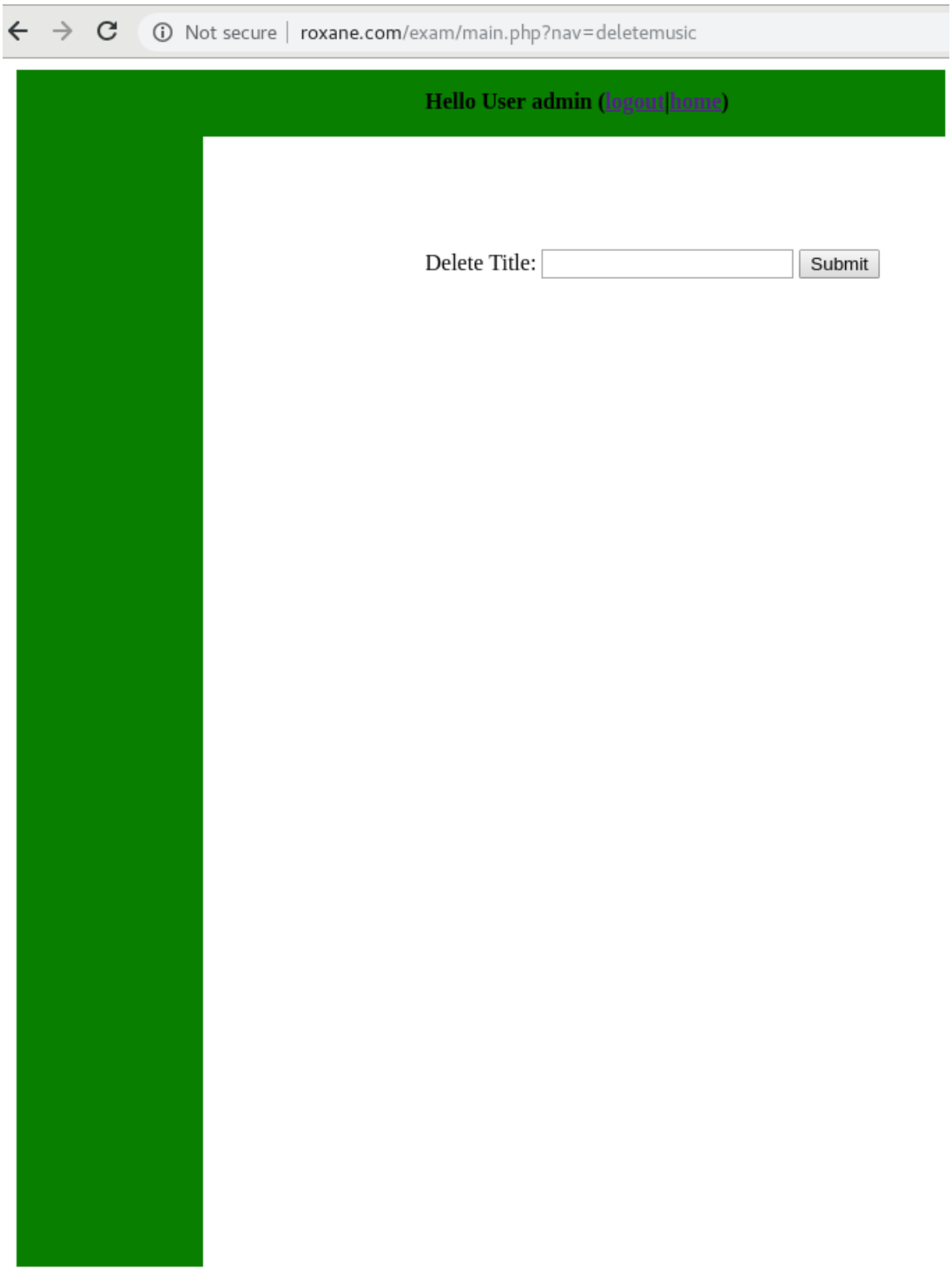



Abbildung 12: Diese ist die Seite zum löschen von Musiktracks