Silas Meier, Rino Vanoni, Basil Gallati

Modul 183 LB 3

17. Juni 2018

Sichere Webapplikation – Dokumentation

Environment Setup

XAMPP

1. **XAMPP Installation**  
   <https://www.apachefriends.org/de/download.html>  
   Apache Web Server starten
2. **Modul-Ordner erstellen**  
   Dateien kopieren nach *[XAMPP-Ordner]/htdocs/Modul183\_lb3*
3. **Datenbank-Erstellung**  
   XAMPP MySQL Datenbank starten, *localhost/phpmyadmin* im Browser aufrufen  
   Datenbank mit Namen *modul183\_lb3* erstellen  
   *skript.sql* in die soeben erstellte Datenbank importieren

**Mögliche weitere Anpassungen:**

* Sollte das Logging nicht funktionieren, weil Schreibrechte fehlen:  
  *sudo chmod -R 777 auf [XAMPP-Ordner]/htdocs/Modul183\_lb3*

SSL

XAMPP unterstützt die Stimulierung/Verwendung von SSL Zertifikaten für eine „localhost-Projekte“ bereits ohne Zusatzeinstellungen. Dazu muss bloss, statt der üblichen Adresse (<http://localhost/modul183_lb3/>*),* <https://localhost/modul183_lb3/> aufgerufen werden. Da das Zertifikat unbekannt ist, wird sie der Browser als unsicher erkennen. Für ein Testing kann eine Ausnahmeregel hinzugefügt werden.

„V-Host Projekte“ müssen in XAMPP konfiguriert werden. Diese Beschreibung kann als Anleitung dazu dienen: <http://magento.xonu.de/tutorials/ssl-in-der-xampp-konfiguration-von-apache-aktivieren/>

* **https-vhost.conf, wichtige Anpassungen:**  
  *SSL Engine on  
  SSLCertificateFile  
  SSLCertificateKeyFile*

Die Funktionalität der Seite/Applikation an sich muss nicht angepasst werden.

Unterschied zwischen einem Beispiel-Virtual-Host über *http* und einem über *https*:

# http

<VirtualHost 127.0.0.2:80>

DocumentRoot "C:/xampp/htdocs/myproject/web"

DirectoryIndex index.php

<Directory "C:/xampp/htdocs/myproject/web">

Options All

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

# https

<VirtualHost 127.0.0.2:443>

DocumentRoot "C:/xampp/htdocs/myproject/web"

ServerName myproject

SSLEngine on

SSLCertificateFile "conf/ssl.crt/server.crt"

SSLCertificateKeyFile "conf/ssl.key/server.key"

<Directory "C:/xampp/htdocs/myproject/web">

Options All

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

Beispiel-Quelle: <https://ourcodeworld.com>

Funktionalität

Login

Das Ausfüllen der Login-Maske löst ein Login aus, wenn das Passwort mit dem Benutzer übereinstimmt.

if(isset($\_POST['login'])) {

$user = $\_POST['userLogin'];

$password = $\_POST['passwordLogin'];

fileLog("User hat sich eingeloggt: ", $user);

login($user, $password);

fileLog("Session wurde gesetzt: ", $user);

}

function login($user, $password){

if(password\_verify($password, findPasswordByUser($user))){

$\_SESSION['userid'] = $user;

echo "Login was successful!";

}else{

echo "Password is incorrect";

}

}

Session

Die Session wird bei jedem erfolgreichen Login erstellt.

if(password\_verify($password, findPasswordByUser($user))){

$\_SESSION['userid'] = $user;

Die Applikation dient nur als Beispiel für ein sicheres Login, könnte aber problemlos erweitert werden. Ein sicheres Logout wäre in diesem Fall zwingend notwendig. Über folgenden Befehl könnte die Session nach einem Logout wieder geschlossen werden:

bool session\_destroy ( void )

Systemkommando

Asdasdasd

Log

Ein Log wird in einer separaten txt-Datei geführt. Bei jeder Registration und jedem Login, sowie bei jeder Session-Vergabe wird ein Event in das dafür vorgesehene Log mit dem Benutzernamen und der Zeit eingetragen. Das Passwort wird selbstverständlich nicht festgehalten.

function fileLog($info, $text) {

$log = "[" . $date = date('Y-m-d h:i:s a') . "] - " . $info . $text."\r\n";

$myFile = fopen("logs/log.txt", "a");

fwrite($myFile, $log);

fclose($myFile);

}

Passwort (Persistenz)

* Mindestlänge von 9 Zeichen, Zahl, Gross- und Kleinbuchstaben
* Hash

function hashPassword($oldPassword) {

return password\_hash($oldPassword, PASSWORD\_BCRYPT, ["cost" => 12]);

}

Die Funktion password\_hash generiert ein automatisch „gehashtes“ Salt das über Bcrypt mit dem Blowfish Algorithmus verschlüsselt wird. Die Verwendung von Peppers wird nicht immer empfohlen. Ein Beispiel dafür kann [dieser Kommentar](https://stackoverflow.com/questions/16891729/best-practices-salting-peppering-passwords) sein. Da die Argumente dieses Kommentars einleuchten und das Hinzufügen eines Peppers das Salt sogar unsicherer machen kann, wird von einer Implementierung eines Peppers abgesehen. Würde eine solche Erweiterung aber angestrebt, sollte das Pepper nicht im Source Code gespeichert werden, sondern optimalerweise auf einem *Hardware Security Module*.

SSL/TLS

Die SSL Einstellungen werden in dieser Arbeit lokal über XAMPP vorgenommen und wurden deshalb im „Environment Setup“ erklärt.

Sicherheitsmassnahmen (vs. Brute-Force, DDOS)

Sadsad

Sichere Registration

* Login und Registration mit Prepared-Statements, um SQL-Injection vorzubeugen.

$conn = connectToDb();

$stmt = $conn->prepare( "INSERT INTO user(user, password)VALUES (?,?);");

$hashedPassword = hashPassword($password);

$stmt->bind\_param('ss', $name, $hashedPassword);

$stmt->execute();

$conn = connectToDb();

$stmt = $conn->prepare("SELECT password FROM user WHERE user = ?;");

$stmt->bind\_param( 's', $user);

$stmt->execute();

User Experience & Graphic User Interface

* Schlichtes, übersichtliches Design mit [skeleton](http://getskeleton.com/) CSS Boilerplate (ausbaufähig)
* HTML 5 Passwort-Pattern Abfrage (min. 9 Zeichen, min eine Zahl, ein Grossbuchstaben, ein Kleinbuchstaben)

<label for="passwordRegister">Password</label><input id="passwordRegister" type="password" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{9,}""

required title="8 or more characters, at least one number, and one uppercase and lowercase letter " name=„passwordRegister">