Modul 183 LB3 - by Team MultiCulti

Varshan Balasunderam, Martin Bissiger, Phetvanxai Nantavong, Yannick Rüfenacht

10.6.2017

Modul 183 LB3 - by Team MultiCulti
Varshan Balasunderam, Martin Bissiger, Phetvanxai Nantavong, Yannick Rüfenacht

10.6.2017

Modul 183 LB3 - by Team MultiCulti
Varshan Balasunderam, Martin Bissiger, Phetvanxai Nantavong, Yannick Rüfenacht

1 Situationsanalyse

1.1 Anforderungen

- Der Auftraggeber verlangt von uns eine Webseite, auf welche man sich mit einem User einloggen kann. Auf diese soll man mit einer Session (Cookies), nach dem verlassen der Seite, ohne Login, wieder zugreifen.
- Eine Registration ist erwünscht aber nicht verlangt.
- Die Hauptfunktion der Webseite soll die Ausführung von Systemkommandos sein, nachdem der Benutzer sich angemeldet hat.
- Der Auftraggeber möchte, dass seine Seite inkl. der Datenbank vor allen möglichen Angriffen gut geschützt ist.
- Ein Log, wo die verschiedenen Error-Meldungen aufnotiert werden, wird verlangt.

2 Umsetzungsmethode

Für die Entwicklung der Webseite verwenden wir HTML, PHP und das CSS für ein nettes Design. Die Entwicklung der Webseite wird mit Hilfe eines XAMPP Servers umgesetzt. Wir haben das Repository und die Dokumentation unseres Projektes auf Github, welches regelmäßig aktualisiert wird. Unser Code setzt voraus, dass bei der Datenbank bereits ein Benutzer mit dem Namen: “multiculti” und dem Passwort: “gibbiX12345” existiert. Unser Datenbankname ist “multiculti”.

2.1 Tools zusammengefasst

CSS, HTML, PHP, mySQL

GitHub, XAMPP, Brackets, Chrome

3 Login

Solange die Session 'user' nicht existiert, kann der Benutzer nicht auf die Inhalte zugreifen. Die Session wird in der Funktion log_user_in erstellt. In der Login-Funktion werden erstmal die eingegebenen Daten (Username und Password) erfasst und in der Datenbank gesucht. Dabei wird ‘Password’ mit einem salt gehasht, durch die PHP Funktion ‘crypt’.

```
/*Kontrolle der Eingabe*/
while($row = $result->fetch_object()){
        if($row->username == $username && $row->password == crypt($password, $row->salt)){
                $logged_in = true;
```

```
        }
}
```

3.1 Schutzmassnahmen

Die Sonderzeichen von ‘Username’ und ‘Password’ werden escaped. Dadurch kann der Angreifer keine SQL-Injections durchführen.
Die Passwortabfrage geschieht mit einem Salt und einem Hash.

3.2 Registration

Wir haben eine Registration erstellt, die es nur registrierten Benutzern erlaubt andere zu registrieren. Das bedeutet, dass man sich nicht selbst registrieren kann. Dies haben wir absichtlich gemacht, um potentielle Gefahren und Sicherheitslücken zu minimieren. Ein Salt wird für den Nutzer erstellt. Dieser wird dem Passwort zugegeben und gehashed.

```
$salt = uniqid(mt_rand(), true);
$password = crypt($password, $salt);
$statement = ConnectionHandler::getConnection()->prepare($query);
$statement->bind_param("sss", $username, $password, $salt);
```

4 Sessions

Die $_SESSION['user'] wird verwendet um bei der Anmeldung die Userdaten abzuspeichern, damit der Benutzer für eine gewisse Dauer angemeldet bleibt und sich somit nicht mehrmals anmelden muss. Die Session wird in der Methode log_user_in($username, $password) definiert und gespeichert.

```
if($logged_in){
$_SESSION['user'] = $username;
}
```

5 Systemkommandos

Die Systemkommandos sind verfügbar, sobald sich ein Benutzer angemeldet hat. Der Benutzer hat dann die Optionen zwischen “echo”, “type”, “dir”, “help”. Zudem haben wir eine Textbox eingebaut, wo man die Kommandos näher definieren kann zB. “Echo text>test.txt”.

```
input class="commandline" type="text" placeholder="System command"
name="syscommand"/> 
....
<fieldset>
<input type="radio" id="print" name="sysopt" value="print" checked>
<label for="print">print</label>
```

```
<input type="radio" id="cd" name="sysopt" value="cd">
<label for="cd">cd</label>
<input type="radio" id="dir" name="sysopt" value="dir">
<label for="dir">dir</label>
<input type="radio" id="help" name="sysopt" value="help">
<label for="help">help</label>
</fieldset>
```

Im controller.php wird die Option “radio” mit der Eingabe aus dem Textfeld verbunden zB. “Echo text>test.txt”.

```
$sysopt = $_GET['sysopt'];
$command = $sysopt . ' ' . $_GET['syscommand'];
```

Dann wird diese Option durch shell_exec verarbeitet und das Ergebnis wird dann in einer Variable gespeichert.

```
$output = shell_exec($command);
$_SESSION['output'] = $output;
```

Schließlich wird diese wieder in dem Hauptdisplay (index.php) angezeigt

```
if(isset($_SESSION['output']))
{
        echo $_SESSION['output'];
        unset($_SESSION['output']);
}
Else
{
        echo 'Output';
}
```

6 Logs

Die Fehlermeldungen wurden so konfiguriert, dass sie in ein separates File aufgezeichnet werden.

```
ini_set('log_errors', 1);
ini_set('error_log', 'log/errors.txt');
error_reporting(E_ALL);
```

7 SSL/TLS

Die Anfrage auf eine offizielle Zertifizierung dauert uns zu lange, deshalb kreieren wir unser eigenes SSL Zertifikat (Self Signed Certificate). Die Erstellung des Zertifikates wurde von XAMPP abgelöst, dadurch mussten wir selber keine Daten konfigurieren.§

Jedoch wurde die Website standardmäßig durch HTTP aufgerufen, weshalb wir bei XAMPP die Server-Datei: httpd-xampp.conf editiert haben. Unser Ziel war es, unsere Website mit HTTPS aufzurufen, indem die Webseite sofort von HTTP auf HTTPS umgeleitet wird.

Source code:

```
<IfModule mod_rewrite.c>
   RewriteEngine On

   # Redirect /xampp folder to https
   RewriteCond %{HTTPS} !=on
   RewriteCond %{REQUEST_URI} MultiCulti
   RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]

   # Redirect /phpMyAdmin folder to https
   RewriteCond %{HTTPS} !=on
   RewriteCond %{REQUEST_URI} phpmyadmin
   RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]

   # Redirect /security folder to https
   RewriteCond %{HTTPS} !=on
   RewriteCond %{REQUEST_URI} security
   RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]

   # Redirect /webalizer folder to https
   RewriteCond %{HTTPS} !=on
   RewriteCond %{REQUEST_URI} webalizer
   RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]
</IfModule>
```

Dieser Code ist das Ergebnis von einer Internetrecherche.
Quelle: https://gist.github.com/nguyenanhtu/33aa7ffb6c36fdc110ea8624eeb51e69

8 Github Repository

Auf unserem Github Repository ist der Source Code, sowie auch eine Kopie von dieser Dokumentation zu finden.

9 Frameworks, Credits

Datei	**Quelle**
ConnectionHandler.php, config.php	ICT - Berufsbildung Bern
Jquery-3.2.1.min.js	https://jquery.com/
CSS - Reset	http://meyerweb.com/eric/tools/css/reset/
Bilder & Icons	https://unsplash.com, http://flaticon.com