16.4 Sicherheit Passwörter

Passwörter (bzw. Authentifizierung) sind das Sicherheitskonzept der ersten Stunde in der Informatik und daran hat sich bisher noch nicht viel geändert. Ob nun eine Anmeldung über Fingerabdruck, Gesichtserkennung oder eine Zeichenkette passiert – das Prinzip bleibt immer das gleiche. PHP Entwickler_innen setzen deshalb auf komplexe Passwörter für SQL-Datenbanken, Admin-Oberflächen, FTP, SSH usw. die sie auch in regelmäßigen Abständen ändern. Was aber, wenn man eine Benutzerverwaltung anbietet, wo ein Benutzer sein Passwort selbst wählen kann?

Passwörter verschlüsseln

Das man Passwörter (in plain Text) nicht mittels `$_GET` versendet, sollte doch klar sein. Da kann man das Passwort gleich ausdrucken und im Dorf auf die Kirchentüre nageln. Selbst eine base64 Verschlüsselung macht das Passwort vielleicht unleserlich – aber definitiv nicht 'unknackbar'. Passworttransfer mit `$_POST` oder als SESSION-Variable bieten schon eine relativ hohe Sicherheit – 100%ige Sicherheit ist in der Informatik nie garantiert!

Deshalb sollte man Passwörter nicht im Klartext abspeichern – sondern nur den HASH des Passwort-Strings. Ein HASH ist ein stark verschlüsselter String eines Passworts der mittels eines Verschlüsselungsalgorithmus (z. B. MD5, SHA1 usw.) generiert wurde. Diesen HASH kann man dann in einer gut geschützten Datenbank (oder aber auch in einem wirklich gut geschützten externen File wie JSON, XML) abspeichern. Wenn ein_e Hacker_in es schafft einen HASH zu ermitteln, dann hat er_sie noch lange nicht das Passwort in den Händen. Gut, es gibt Programme die einen HASH zurückkonvertieren – aber diese können schon Monate bis Jahre an so einem gehashten Passwort rechnen.

In PHP gibt es die **Password Hashing API** mit Funktionen zur Passwortsicherheit:

PHP

`password_hash()`

Erzeugt einen HASH – das Beispiel übergibt der Variable `$passwort` einen String, der ungefähr so aussieht:

`$2y$10$iO0G8miW/d6f8ALNVhNvOe67eVVnuzu2Fu8XlYfyIqdhj0DbP5h8O`

```php
$passwort = password_hash("sehrgeheim", PASSWORD_DEFAULT);
```

PHP

`password_verify()`

Die Funktion vergleicht das Passwort mit dem HASH und gibt (bei Übereinstimmung) TRUE bzw. 1 zurück. Im Beispiel ist der HASH in der Variable `$passwort` gespeichert.

```php
if(password_verify("sehrgeheim", $passwort)) {
   echo 'Passwort stimmt!';}
```

Im Beispiel wird nur der HASH gespeichert. Die Prüfung erfolgt dann im PHP Code, indem die Eingabe mit dem HASH über die Funktion **`password_verify()`** *verglichen wird.*

Selbstverständlich kann die Sicherheit noch erhöht werden, indem man das Passwort vorher zusätzlich noch mit openSSL verschlüsselt, dem Hash ein Salt mitgibt oder sogar eine eigene Kryptographie hinzufügt.