Gutes Passwort mit PHP erstellen

Das Verfahren lässt sich für beliebige, eindeutige, kurz, prägnante Zeichenketten verwenden. Es kann auch zum Beispiel für die "Passwort vergessen" Funktion genutzt werden.
An ein gutes Passwort sind aber verschiedene Bedingungen geknüpft.

== Abwechslung ==
Kennt einer Angreifer 10 Passwörter, sollte man daraus nicht das 11te schließen können. Zwischen den Passwörtern darf es keine Gemeinsamkeit geben.
Dazu gibt es in PHP die Funktion rand(). Sie liefert eine Zahl.

Quellcode

```
1. $random = rand();
2. echo $random; //1807510776
```

Die Funktion hat einen kleinen Wertebereich. Um die Zahl 1807510776 zu knacken bräuchte man nur 1807510776 Loginversuche.

Quellcode

```
1. for($i=0; $i<99999999999; $i++) {
2. versuchelogin($i);
3. }
```

== Eindeutigkeit ==
Die Passwörter sollten "relativ" eindeutig sein - es liegt auf der Hand, dass ein Passwort, dass von jedem Benutzer verwendet wird, kein gutes ist.
Welche Funktionen gibt es um dies in PHP zu erreichen? Eine Handbuchrecherche nach unique liefert uns die Funktion uniqid.
Dieser löst auch gleich das Problem mit dem kleinen Wertebereich. Sie setzt statt dem Dezimal- auf dem Hexadezimalsystem auf und hat außerdem eine andere Größe.

Die Methode besitzt folgende Signatur:

Quellcode

```
1. string uniqid ([ string $Präfix [, bool $mehr_entropie ]] )
```

Das Handbuch verspricht, dass die Rückgabe der Funktion durch den Parameter $mehr_entropie noch eindeutiger wird. Als Präfix nehmen wir die Zufallszahl.

Quellcode

```
1. $random = rand();
2. $unique = uniqid($random, true);
3. echo $unique; //18075107764921ae0bb96c37.27618105
```

Hierdurch wird eine ID mit 32 Zeichen (ein 128 Bit-Hex-Wert) erzeugt, die nur schwer vorhersehbar ist.

== Streuung ==
Streuen bezeichnet in der Informatik das Abbilden großen Quellmenge in eine kleinere Zielmenge. Da dieser Hashwert kürzer als die originale Datenstruktur ist sind Kollisionen prinzipiell unvermeidlich. Eine gute Hashfunktion zeichnet sich

aber dadurch aus, dass sie wenige Kollisionen erzeugt.
Eine bekannte Hashfunktion ist md5(). Begrenzen wir die Länge auf eine Anzahl haben wir damit unser erstes zufälliges Passwort:

Quellcode

```
1. function simplePassword($length) {
2. $random = rand();
3. $unique = uniqid($random, true);
4. $hash = md5($unique);
5. return substr($hash, 0, $length);
6. }
```

== Volles Alphabet ==
Um den Wertebereich des ganzen Alphabets auszuschöpfen

Quellcode

```
1. function simplePassword($length) {
2. $password = "";
3. while (strlen($password) < $length) {
4. $array[0] = chr(rand(48,57)); // Null bis Neun
5. $array[1] = chr(rand(65,90)); // Großbuchstaben
6. $array[2] = chr(rand(97,122)); // Kleinbuchstaben
7. $password .= $array[array_rand($array)];
8. }
9. return $password;
10. }
```

Alles anzeigen

== Wohlingendes Passwort ==
Ein Verfahren dass indirekt zu mehr Sicherheit beiträgt ist das Verfahren zum mnemonischen Passwort.
Die Mnemonik ist die Gedächtniskunst. Sie soll mit Regeln, Kniffen und Tricks das Erinnerungsvermögen unterstützen und leistungsfähiger zu machen.
Im realen Leben ist die Eselsbrücke eine bekannte mnemonische Technik.
Natürlich können spezielle Algorithmen Passwörter auf diese Art um einen Bruchteil schneller knacken. Allerdings ist der Aufwand der Anpassung immens groß.

Einen Code für wohlklingende Passwörter findet man unter mathias-bank.de/2006/08/15/wohlklingende-passworter/.

== Demo ==
Unter Nutzung des "Vollen Alphabets" habe ich folgende Demo online gestellt: demo.easy-coding.de/php/passwords/.
Den Code dazu findet ihr unter demo.easy-coding.de/php/passwords/download.zip.