# Password security

Wie wählt man richtige Kennwörter?  
Wichtig ist eine gewisse Länge zB >10 Zeichen(Groß + Klein + Ziffer + Sonderzeichen ist mittlerweile nicht mehr unbedingt Standard)  
auf keinem Fall Wörter die im Lexikon stehen würden, nicht einmal Textpassagen (wegen brute Force)  
zB Anfangsbuchstaben eines Satzes   
Ich bin ein Schüler der Abteilung Informatik und 2001 geboren.

ergibt

Ib1SdAIFu2001g.

Das ist bereits eine gute Grundlage, ein Angreifer der die Datenbank mit Benutzernamen und Kennwörtern knackt hat aber dann trotzdem die Authentifizierungsdaten.

Deshalb:

# Hashing

Kennwörter sollen nie im Klartext gespeichert werden!!!   
Sie werden mit einer Einwegverschlüsselung umgewandelt und der verschlüsselte Wert wird gespeichert. Die Mischung aus Verschlüsselung und Abbildung auf eine gewisse Menge an Zeichen nennt man hashing. Beim Hashen wird aus einem beliebig langen Text eine Folge von Zeichen, aus denen man nicht auf den Text rückschließen kann. Minimalste Änderungen im Original ergeben eine vollkommen anderen hash-Wert. Die Wahrscheinlichkeit, dass zwei Texte denselben Hashwert ergeben (Kollissionen) sollte verschwindend gering sein (konnte bei md5 aber erzeugt werden, deshalb ist es nicht mehr unbedingt sicher)

Soll geprüft werden, ob ein Passwort richtig ist, wird das zu prüfende Passwort gehasht und mit dem gespeicherten gehashten Original vergleichen. Man muss also beim Überprüfen den hash-Algorithmus durchlaufen. Ein gutes Hashverfahren sollte schnell sein, zu schnell braucht es auch nicht sein, damit ein Bruteforce Angriff möglichst ausgebremst wird. Es macht für den Benutzer keinen merklichen Unterschied ob er nach der Eingabe nach 1ms oder nach 200ms weiß ob das Kennwort richtig ist, ein brute Force-Angriff würde dadurch aber um den Faktor 200 länger und damit uninteressant.

Da man vom hash nicht auf das Original zurückrechnen kann (Einweg: nur Original->hash geht) werden von Hackern riesige rainbow-tables verwendet: diese enthalten jeweils Klartetxt Kennwörter und dazugehörige hashes. Man sucht nach dem gleichen hash in der Tabelle und hat dann das Klartextkennwort. Das gibt es auch als online-Dienst.

Ob ein Kennwort gut ist, kann man auch mit „John the Ripper“ testen, einer Software die verschiedenste Schlüsselverfahren kennt und auf diese BruteForce oder lexikalische Angriffe anwendet.

Wie das (ab 2025) mit Quantencomputern aussieht, ist eine andere Sache.

Hashverfahren: md5 und sha-1 gelten nicht mehr als sicher

sha … secure hash algorithm

sha256 sollte noch bis 2022 sicher sein. 256 heißt, dass der Hashwert 256bit also 32 Byte lang ist

oder ripemd-160 oder whirlpool. Derzeit wird weiter nach hash-Verfahren gesucht, die noch sicherer sind und gegen die auch ein Quantencomputer keine vernünftige Chance hat (postquantenkryptographie).

## zusätzlich features

1. damit gleiche Kennwörter nicht den gleichen hash ergeben wird das Kennwort vor dem hashen gesalzen, dh mit einer zusätzlichen Zeichenkette verlängert. Damit wird ein lexikalischer Angriff schwierig, weil zusätzlich zu den Wörtern alle möglichen Salze dazuprobiert werden müssen.  
   Gleiche Kennwörter mit gleichem Salz ergeben aber immer noch gleiche hashes. Somit könnte jemand, der zB eine Datenbank mit hashes bekommt, jene hashes die gleich sind zuerst zu knacken probieren. Bei Erfolg hätte er dann gleich den Zugang mehrerer Benutzer geknackt. Deshalb wird das salt jedesmal geändert und das salt im hash mitgespeichert. Das ist übrigens keine Sicherheitslücke, ein brute force Angriff muss dann trotzdem erst wieder jeden Hash knacken und beim lexikalischen Angriff muss das ganze Lexikon mit dem Salt vermsicht durchprobiert werden. Rainbow tables sind dann sowieso wertlos (es gibt natürlich schon rainbow tables mit den meistverwendeten salts)
2. Pfeffern: zum Ganzen kommt vor dem hashen noch ein geheimer Code dazu, der irgendwo anders im Server gespeichert ist, so dass der Diebstahl der Datenbank noch wertloser wird. Oder überhaupt wieder ein eigener Pfeffer für jeden Benutzer, wobei diese Infos in einer Tabelle irgendwo im sicheren Bereich des Servers, außerhalb der Datenbank gespeichert wird.
3. mehrere Iterationen machen, dh das gehashte nochmal hashen usw, mit einem sleep dazwischen, damit brute force Angriffe länger dauern
4. Und sql-injection verhindern, in dem prepared statements oder EingabeValidierung oder eingebaute Funktionen wie real\_esacpe\_string in php verwendet werden.  
   Erklärung einer Art von injection:  
   am Server folgt ja zum Überprüfen der accountdaten evtl eine Query der Form  
   SELECT \* FROM tablexy WHERE user=‘x‘ AND password=‘y‘;  
   wobei x und y die eingebenen Account daten sind.  
   Dann kann man als user zB eingeben: 1=1 OR irgendwas  
   Das wird am Server in obigem query verwendet und ergibt dann  
   SELECT \* FROM tablexy WHERE 1 = 1 OR user=‘x‘ AND password=‘y‘;  
   was natürlich immer true ist  
   Beim prepared statement werden SQL und Variablen getrennt behandelt und exekutiert, somit kann SQL Code in der Variable nicht Teil des SQL Statements werden.  
   Bei real\_esacpe\_string wird dieser mit Anführunsgzeichen versehen und so als String zusammengehalten.  
   Es gibt aber noch andere sql injections zB solche, die einen neuen Benutzer mit admin Rechten erzeugen.

php Funktionen: password\_hash und password\_verify verwenden

$h=password\_hash($original);

…….

If(password\_verify($h,$eingegebenesKennwort)==true)  
 alert(ok);  
else  
 alert(„Fehlerhafte Authentifizierung“)

Lexikon: gibt’s zum Herunterladen: die 10000, 100000 usw bekanntesten Kennwörter  
Nummer 1 und 2 sind 123456 und password

In anderen Programmiersprachen gibt es auch standardisierte Verschlüsselungsverfahren

zB scrypt oder Argon-2