1 Passwörter speichern

1.1 Was viele Menschen glauben, was beim Login passiert

Wenn Sie sich an einem IT-System anmelden, geben Sie in den meisten Fällen Ihren Benutzernamen und ein Passwort ein. Jede*r, der diese beiden Informationen *weiß*, kann sich als Sie ausgeben (*authentisieren*). Man spricht daher von einem *wissensbasierten System*.

Sie wenden bestimmt Regeln für sichere Passwörter an, unter anderem wissen Sie, dass Sie niemandem Ihr Passwort sagen dürfen. Aber irgendwie muss das IT-System, an dem Sie sich anmelden, ja Ihr Passwort kennen, sonst kann es ja nicht entscheiden, ob das eingegebene Passwort das richtige ist. Man *könnte* also glauben, dass alles ganz einfach ist, und wie folgt abläuft:

1. Die*Der Benutzer*in schickt den Benutzernamen und das Passwort an den Server.
2. Der Server sucht in der Benutzertabelle nach dem Benutzernamen und vergleicht das eingegebene Passwort mit dem gespeicherten.
3. Stimmen beide überein, lässt der Server die*den Benutzer*in rein, sonst nicht.

Diese naive Vorstellung wird auch in der Lach- und Sachgeschichte “Internet” der Sendung mit der Maus präsentiert:

Naiver Passwortvergleich in der Lach- und Sachgeschichte “Internet” der Sendung mit der Maus [https://www.wdrmaus.de/filme/sachgeschichten/internet.php5]

Nun haben sie in der Tabelle `users` ja bereits die Spalte `password` gesehen:

username	**password**
niclas258	`\$2y\$10\$VdoILxrn6CZU/PtClCDJY.eB3s0lzYiGkY7NokYpZJpMsJ2pq/tuK`
rafael54	`\$2y\$10\$HkuheWgNzbw9KLo8N3kOHeM6hh7LTpLwqMOcAY7QpqKzOi7NIYqUm`

Vielleicht haben Sie sich gedacht, dass solche Passwörter sicherlich kein Mensch eingeben würde, aber es sind ja schließlich nur Testdaten. Oder sind das gar nicht die Passwörter? Oder sind sie verschlüsselt?

Aufgabe 8.1 Passwörter im Klartext?

Überprüfen Sie, ob InstaHub die Passwörter im Klartext speichert, indem Sie folgende Tests unternehmen:

1. *Versuchen* Sie, sich an Ihrem InstaHub mit einem beliebigen Benutzernamen und dem zugehörigen Wert in der Spalte `password` anzumelden.
2. *Überprüfen* Sie, ob das von Ihnen verwendete `admin`-Passwort in der Datenbank steht.
3. *Überprüfen* Sie, ob sich der Wert des `admin`-Passworts ändert, wenn Sie das Passwort über das Profil ändern.

Aufgabe 8.2 Admins Probleme mit Passwörtern

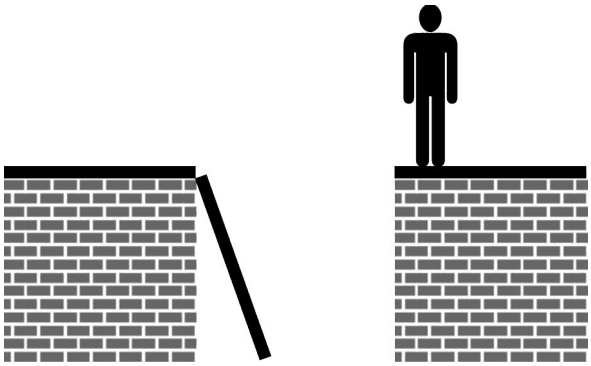
1. *Recherchieren* Sie, wann in der letzten Zeit einem Server unverschlüsselte Passwörter abhanden gekommen sind!
2. *Finden* Sie Gründe, die dafür sprechen, Passwörter nicht unverschlüsselt zu speichern.
3. *Finden* Sie Gründe für die Forderung, dass Passwörter so gespeichert werden müssen, dass es auch den Administratoren nicht möglich ist, deren Klartext wieder herzustellen!

1.2 Was wirklich beim Login passiert

Das klingt merkwürdig. Man muss das Passwort vergleichen können, aber zugleich soll man es nicht speichern. Wie soll das nur gehen? Zeit für einen kleinen Exkurs:

1.2.1 Falltüren

Es ist sehr einfach durch die Falltür nach unten zu gelangen, während sich der Weg zurück manchmal als ausgesprochen schwierig, wenn nicht gar unmöglich, erweist:



Durch eine Falltür geht es nur in eine Richtung leicht hindurch…

Diese Idee kann man auf das Speichern von Passwörtern übertragen. Wir schicken das Passwort durch die Falltür und speichern das Ergebnis. Bei der Anmeldung schicken wir das eingegebene Passwort ebenfalls durch die Falltür und vergleichen dann das Ergebnis mit dem gespeicherten Wert. Stimmen beide überein, war das Passwort richtig. Klingt erstmal merkwürdig.

Wie Sie wissen, können Computer nur Zahlen speichern. Auch Texte werden intern als Zahlen gespeichert. Und mit denen kann man rechnen. Mathematiker*innen kennen so genannte *Einwegfunktionen*. Die Funktionswerte solcher Einwegfunktionen sind zwar verhältnismäßig leicht auszurechnen, aber es ist fast unmöglich in der Gegenrichtung aus einem Funktionswert zurück zum eingegebenen Wert zu kommen.

Als Beispiel für eine Funktion die einer Einwegfunktion ähnelt, kann die Primfaktorzerlegung dienen. Die Zerlegung der Zahl 21 in ihre Primfaktoren ist leicht: 7 und 3. Die Primfaktorzerlegung wird bei sehr großen Zahlen aber sehr zeitaufwendig. Versuchen Sie doch mal, die Zahl 162.249 in ihre Primfaktoren zu zerlegen! Sie werden ziemlich lange suchen müssen, bis Sie eine Lösung finden. Ihnen bleibt nur der Reihe nach alle Primzahlen ausprobieren. Umgekehrt ist es für Sie aber ein Leichtes, die Aufgabe $433 \cdot 443$ zu lösen. Das Ergebnis ist: 162.249. Wenn man jetzt die Zahlen noch viel, viel größer wählt, wird die Primfaktorzerlegung ein sehr großes Problem.

In der Informatik nennt man diese Funktionen **Hashfunktionen** (engl. *hash*: **Hackfleisch**). Falls Sie sich fragen, warum so etwas Hackfleischfunktion heißt: Spielen Sie die Falltür-Idee mit den Begriffen Katze und Fleischwolf durch …

Nennen wir die Hashfunktion $h(p)$. Dann läuft unsere Anmeldeprozess so ab:

1. Der*die Benutzer*in gibt ihren*seinen Benutzernamen und das Passwort p ein.
2. Der Server bildet $h(p)$ und vergleicht das Ergebnis mit dem gespeicherten Wert.
3. Sind die beiden Werte gleich, war die Authentisierung erfolgreich.

Aufgabe 8.3 Der Wert der Hashtabelle

Durch Umstände, die wir an dieser Stelle nicht weiter ergründen wollen, sind suspekte Individuen aus Ihrem Informatik-Kurs in den Besitz der Passworttabelle des Schulverwaltungsprogrammes gekommen. *Erklären* Sie, warum es ihnen nicht ohne weiteres möglich sein wird, sich am Schulverwaltungsprogramm als Lehrkraft auszugeben!

1.2.2 Gebräuchliche Hashfunktionen

Gebräuchliche Hashfunktionen sind u. a. MD5, SHA-1, SHA-256 oder bcrypt. Letztere wird übrigens von InstaHub verwendet. Die Hashfunktionen bilden aus jeder übergebenen Zeichenkette einen Hashwert fester Zeichenlänge. Egal ob Sie “Hallo Welt!” oder die ganze Bibel hashen - es kommt eine wüste Zeichenkette stets derselben Länge heraus, schon eine geringfügige Änderung der Eingabe führt zu einem völlig anderen Hash-Wert:

Text	**Hash-Wert (SHA-1)**
Die Bibel[1]	`0f1a3d197eb98fb4638e-f028661591602 69b89b8`
“Hallo Welt!”	`726c3e8861ab0652a5043ea5f aff6d3ef33fb209`
“Hallo Welt?”	`dff4b83f415b61a06594d-c57c4972c77da0c833f`

Es gibt also mit Sicherheit mehrere Texte, die zu demselben Hashwert wie “Hallo Welt!” führen. Eine sichere Hashfunktion erkennt man daran, dass man diese so genannten *Kollisionen* nur zufällig finden kann.

Aufgabe 8.4 Kollisionen

Nehmen wir an, als (völlig ungeeignete) Hashfunktion wird verwendet “Nimm die ersten zwei und die letzten zwei Zeichen.”

1. *Ermitteln* Sie den Hashwert des Passwortes `Geheim!`.
2. *Finden* Sie ein kollidierendes Passwort zu `Geheim!`.
3. *Erklären* Sie, warum es problematisch ist, wenn sich Kollisionen anders als durch Ausprobieren finden lassen!

1 Luther-Ausgabe 1912

1.2.3 Angriffsmöglichkeiten auf Ihr Hackfleisch

Auf den ersten Blick haben wir alles getan, um die Passwörter unserer Benutzer*innen zu schützen. Aber…

Bei aller Sorgfalt kann es überall passieren, dass die Tabelle mit den gehashten Passwörtern abhanden kommt. Wir müssen also damit rechnen, dass sich jemand in aller Ruhe zu Hause hinsetzen kann und die Tabelle unter die Lupe nimmt.

Die*der Angreifer*in kann nun sehr viele verschiedene Passwörter ausprobieren und $h(p)$ bilden. Wenn der Hashwert in der Tabelle enthalten ist, hat sie*er ein Passwort geknackt. Dieser Angriff kann sowohl in Form eines *Wörterbuchangriffs* als auch in Form eines *Brute-Force-Angriffs* erfolgen.

Aufgabe 8.5 Angriffsarten

Erklären Sie, was ein Wörterbuchangriff und was ein Brute-Force-Angriff ist.

1.2.4 Salz und Pfeffer erschweren Angriffe

Der Einsatz einer Hashfunktion alleine reicht also leider nicht aus. Wir müssen das entstandene Hackfleisch noch ordentlich würzen. Hierzu verwenden wir Salz (*salt*) und Pfeffer (*pepper*). Informatiker*innen mögen offenbar keine Zwiebeln.

*Informatiker*Innen mögen offenbar Hackfleisch, aber nur ohne Zwiebel! (Bild von Andreas Lischka auf Pixabay)*

Der oben beschriebene Angriff auf die Hashtabelle war ja erfolgversprechend, weil man jeden erzeugten Hashwert in vielen Zeilen und vielen Tabellen suchen konnte. Wenn wir dieses Problem beheben können, sind wir sehr viel weiter. Man kann sich das Vorgehen wie folgt vorstellen:

1. An das Passwort wird ein von dem*von der Benutzer*in abhängiger Text (das *salt*) angehängt, z. B. der Benutzername. Wenn man das Passwort jetzt hasht und speichert, muss die*der Angreifer*in also den Hashwert für jede*n Benutzer*in einzeln ermitteln. Das kostet viel Zeit.
2. An das um das *salt* verlängerte Passwort hängt der Server vor dem hashen aber noch einen Text an, ein geheimes *pepper*. Er bildet also $h(passwort+salt+pepper)$. Jetzt kann der*die Angreifer*in nicht einmal mehr für mehrere Server zugleich die Hashwerte ermitteln, sondern muss dies für jeden Server einzeln tun. Und dafür braucht er*sie auch noch Kenntnisse über *pepper* und *salt* und vor allem: sehr viel Zeit!

Aufgabe 8.6 Salz und Pfeffer

Ein Server verwendet zur Speicherung der Passwörter SHA-256. Als *salt* wird der Benutzername verwendet. Das *pepper* ist `Kvillbuche`. Der Hashwert wird in der Form $h(p\&salt\&pepper)$ ermittelt. Wobei $\&$ für die Aneinanderreihung von Zeichenketten steht. 2. Der*Die Nutzer*in `nicolas258` meldet sich mit dem Passwort `TaKaLu908#`an. *Ermitteln* Sie (z. B. auf www.hashgenerator.de) den zugehörigen Hashwert! 3. *Überprüfen* Sie Ihre Lösung, indem Sie den folgenden QR-Code entschlüsseln:

Lösung Aufgabe 8.6 als QR-Code