So funktioniert es

- Wie Sie in den Aktivitäten vier und fünf gelernt haben, ist eine kryptografische Hash-Funktion eine einseitige Berechnung; Das ursprüngliche Passwort kann nicht aus seinem Hash rückgerechnet werden. Wenn ein Hacker in einen Computer eindringt und die Passwort-Hashes stiehlt, kann er sie nicht zum Anmelden verwenden, sie könnten jedoch für eine **Rainbow Table** oder einen **Brute-Force-Angriff** (rohe-Gewalt-Angriff) verwendet werden.
- Ein Brute-Force-Angriff erstellt alle Permutationen möglicher Klartext-Passwörter, berechnet den Hash und vergleicht ihn mit dem gestohlenen Hash. Ein Brute-Force-Angriff ist ein langwieriger Prozess, der viel Rechenzeit erfordert.
- Ein Grund dafür, dass dieser Angriff so viel Rechenzeit erfordert, ist die Anzahl der Permutationen für eine Passwortregel.
- Die Permutationen einer Passwortregel stellen alle Möglichkeiten dar, wie das Passwort basierend auf der Anzahl und Art der von der Regel geforderten Zeichen gebildet werden könnte.
- Beispielsweise hat eine Passwortregel mit genau vier Zeichen, die nur Großbuchstaben verwendet, an jeder Position eines von 26 verschiedenen Zeichen. Die Anzahl der Möglichkeiten, ein Passwort basierend auf dieser Regel zu bilden, beträgt 26 x 26 x 26 x 26 = 456.976 oder 26^4 verschiedene Passwort-Permutationen!
- Das obige Beispiel kann wie folgt verallgemeinert werden:

$$\textit{mögliche Zeichen}^{\textit{Passwortlänge}}$$

- Regeln, die lange Passwörter erfordern und aus vielen möglichen Zeichen bestehen, sind für Brute-Force-Angriffe schwieriger, weil es viel mehr mögliche Passwörter zum Testen gibt. Beispielsweise hat eine Regel, die acht Zeichen erfordert, die aus einem Kleinbuchstaben, einem Großbuchstaben, einer Zahl und einem Sonderzeichen (es sind 32) bestehen, $(26+26+10+32)^8$= 6.095.689.385.410.816 mögliche Passwörter!
  Lange Passwörter, die aus vielen Zeichentypen bestehen und häufig geändert werden, verringern das Risiko eines Brute-Force-Angriffs.

Was ist zu tun?

1) Üben Sie mit der Passwortstärke (Permutationen):
   a) Fahren Sie mit Seite 1.3, „possible_passwords.py“, fort und überprüfen Sie den Python-Code. Sehen Sie, wie drei verschachtelte Schleifen jedes mögliche Passwort generieren, basierend auf der Passwortregel, die drei Großbuchstaben erfordert? Führen Sie das Programm aus und beobachten Sie, wie es funktioniert. Sehen Sie, wie die drei verschachtelten Schleifen die Permutationen erzeugen? Wie viele mögliche Passwörter lassen sich aus drei Großbuchstaben bilden? **Hinweis: Dieses Programm dauert etwa zehn Minuten!** Wenn Sie ungeduldig werden, halten Sie die Taste [On] gedrückt, um das Programm zu unterbrechen.
   b) Gehen Sie zur Rechnerseite und sehen Sie sich die Beispielberechnung an. Sehen Sie, dass die Passwortregel, die drei Großbuchstaben erfordert, 17.576 mögliche Passwörter ergibt? Hat diese Berechnung die Anzahl der Passwörter ergeben, die das vorherige Python-Programm generiert hat?
   c) Gehen Sie mit einer Problemaufforderung zur nächsten Rechnerseite weiter und schließen Sie die Übungsberechnung ab. Platzieren Sie den Cursor unter der Problemaufforderung und geben Sie die Berechnung mit den Taschenrechnertasten ein. Haben Sie 3.226.266.762.397.899.821.056 erhalten? Können Sie sich vorstellen, wie Sie die verschachtelten Schleifen im Python-Programm ändern müßten,

um jedes mögliche Passwort basierend auf der neuen Regel zu generieren? Können Sie sich vorstellen, wie lange es dauern würde, das Programm auszuführen?

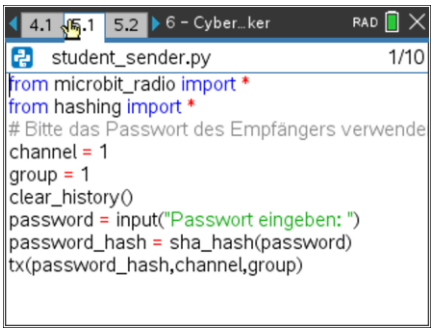
d) Fahren Sie mit der folgenden Diagrammseite fort. Verwenden Sie die Trace-Funktion, um das Diagramm der Anzahl der Passwortpermutationen (y) als Funktion der Passwortlänge (x) zu untersuchen. Was fällt Ihnen an der Form dieser Kurve auf? Da viele Permutationen mehr Rechenzeit erfordern, was sagt die Grafik über die Passwortlänge und die Anfälligkeit für Brute-Force-Angriffe aus?

2) Legen Sie ein Passwort für Ihren micro:bit fest.
   a) Verbinden Sie Ihren micro:bit mit Ihrem Taschenrechner.
   b) Fahren Sie mit Seite 2.1 fort und führen Sie das Programm aus. Erstellen Sie ein Passwort nach der Regel: vier Zeichen, die nur Buchstaben enthalten. *Hinweis:* Um später in der Aktivität „brute_force.py“ Laufzeit zu sparen, **wählen Sie ein Passwort mit einem Anfangsbuchstaben am Anfang des Alphabets**. Zum Beispiel werden „Abba“ oder „Abby“ VIEL früher geknackt als „Zeno“ oder „Zola“. Außerdem ist „ABBA“ schneller als „abba“. Können Sie sich den Python-Code ansehen und erklären, warum? Hinweis: Wenn es zu lange dauert, bis Ihr Rechner fertig ist, halten Sie die Taste [On] gedrückt, um das Programm zu unterbrechen.
3) Der Brute-Force-Angriff:
   a) Tauschen Sie Ihr micro:bit mit einem Gruppenmitglied aus. Sagen Sie ihm nicht Ihr Passwort!
   b) Fahren Sie mit Seite 3.1 fort und führen Sie das Programm aus. Der Hash des Passwortes wird aus der Datei „password.txt“ abgerufen und angezeigt. Dies ist die Datei und der Hash, die ein Hacker zu stehlen versucht, um die Sicherheit Ihres micro:bit zu gefährden! *Hinweis:* Der Hash kann nicht rückgängig gemacht werden; Der Brute-Force-Angriff ist jedoch eine Problemumgehung, die das Klartext-Passwort des gehashten Passworts des micro:bit findet.
   c) Nachdem Sie das Programm in der Python-Shell auf Seite 3.2 ausgeführt haben, drücken Sie die Taste [var]. Ist Ihnen die Variable „hacked_hash“ aufgefallen? Wählen Sie die Variable aus dem Menü aus und sehen Sie, wie sie den 256-Bit-Hash Ihres Passworts an die **Python-REPL-Eingabeaufforderung >>> in der Shell** zurückgibt. Hinweis: Sie verwenden diese Variable in Schritt 5, der Remote-Anmeldeaktivität.
   d) Gehen Sie mit „brute_force.py“ zur nächsten Seite und überprüfen Sie das Programm. Beachten Sie, dass es in den vier verschachtelten Schleifen keine Druckanweisungen gibt. Die Ausgabe jeglicher Art, beispielsweise das Drucken auf dem Bildschirm, ist relativ langsam. Durch das Eliminieren von print-Anweisungen wird das Programm wesentlich schneller. Beim Schreiben von Programmen ist die **Optimierung** von häufig wiederholtem Code unerlässlich. Führen Sie das Programm aus und notieren Sie die Anzahl der Iterationen und die verstrichene Zeit. Vergleichen Sie Ihre verstrichene Zeit mit anderen in Ihrer Gruppe. Das Programm gibt das Klartext-Passwort mit einem Hash zurück, der mit dem gehackten Hash übereinstimmt. Merken Sie sich das Klartext-Passwort, um es in der nächsten Aktivität zu testen.
4) Authentifizierung:
   a) Fahren Sie mit Seite 4.1 „authentication.py“ fort und führen Sie das Programm aus. Geben Sie dreimal das falsche Passwort ein. Können Sie nach der Untersuchung des Python-Codes erklären, wie er wiederholte Authentifizierungsversuche verhindert? Wie erhöhen Begrenzungsversuche die Sicherheit?
   b) Führen Sie das Programm mit dem geknackten Klartext-Passwort erneut aus, um das Ergebnis des Brute-Force-Angriffs zu testen. Konnten Sie es hacken?

5) Remote-Anmeldung:
   - **Der Empfänger**
     - Ändern Sie das Passwort wie in Aktivität 2 beschrieben. Achten Sie auf gute **Passworthygiene** und verwenden Sie nicht dasselbe Passwort erneut. Wählen Sie ein Passwort am Anfang des Alphabets. Flüstern Sie dem Absender Ihr Passwort zu, damit sich der Absender bei Ihrem micro:bit anmelden kann. *Halten Sie das Passwort* unbedingt vor Hackern *geheim*. Gehen Sie zu „student_receiver.py“, ändern Sie die Gruppe auf die ihnen zugewiesene Nummer und führen Sie das Programm aus, *bevor* der Absender sein Programm ausgeführt hat.
   - **Der Sender**
     - Gehen Sie zu „student_sender.py“, ändern Sie die Gruppe auf die ihnen zugewiesene Nummer und führen Sie Ihr Programm aus, *nachdem* der Empfänger und der Hacker ihr Programm gestartet haben. Senden Sie das Passwort des Empfängers, um sich aus der Ferne bei seinem micro:bit anzumelden.
   - **Der Hacker**
     - Gehen Sie zu „student_hacker.py“, ändern Sie die Gruppe auf die ihnen zugewiesene Nummer und führen Sie dann das Programm aus, bevor der Absender seine Nummer ausgeführt hat. *Hinweis:* Dieses Programm empfängt den vom Absender gesendeten *Passwort-Hash*, um sich beim micro:bit des Empfängers anzumelden.
     - Verwenden Sie den gestohlenen Hash und das Modul brute_force_cracking, um das Passwort des Empfängers aus dem abgefangenen Hash zu knacken. Verwenden Sie die Funktion „brute(hacked_hash)“ des Moduls, um den Hash zu hacken. Geben Sie >>>brute(hacked_hash) ein. Denken Sie daran, dass die Variable „hacked_hash“ über die Taste [var] ausgewählt werden kann.
     - Sobald der Hacker das Passwort des Empfängers geknackt hat, sollte der Empfänger das Programm „student_receiver.py“ erneut ausführen, um zu testen, ob der Hacker in Ihr micro:bit eindringen kann. Der Hacker sollte zu „student_sender.py“ vordringen und das geknackte Klartext-Passwort senden. *Hinweis:* Wenn der Hacker erfolgreich ist, sollte er sich beim micro:bit des Empfängers anmelden können, ohne jemals das Passwort zu erfahren!

Die Programme

Rolle des Senders



Rolle des Empfängers

Rolle des Hackers

Weitere Übungen

- Tauschen Sie Ihre Rolle mit einem Teammitglied. Jedes Teammitglied sollte jede Rolle spielen und versuchen, das Passwort des anderen zu hacken.
- Notieren Sie die Zeit, die zum Knacken eines Passworts benötigt wird, das mit dem Buchstaben „A“ beginnt, und wiederholen Sie den Vorgang für ein Passwort, das mit „B“ beginnt. Fahren Sie mit den Buchstaben bis „E“ fort. Erstellen Sie ein Diagramm mit der Position der Buchstaben, z. B. A = 1, B usw., auf der horizontalen Achse und der Programmausführungszeit auf der vertikalen Achse. Können Sie die Form der Kurve dieses Diagramms vorhersagen?

Prüfen Sie Ihr Verständnis

- Ein Brute-Force-Angriff ist ein rechenintensiver Hack, der den Hash aller Permutationen von Passwörtern für eine bestimmte Passwortregel berechnet und sie mit einem gestohlenen Passwort-Hash vergleicht. Wenn die Hashes gleich sind, wird das Klartext-Passwort für den gestohlenen Hash ermittelt.
- Wenn Sie Ihr Passwort häufig ändern, viele selten verwendete Zeichen verwenden, dasselbe Passwort nicht für verschiedene Konten wiederverwenden und Passwörter nicht weitergeben oder aufschreiben, tragen Sie dazu bei, Ihre Konten vor Hackern zu schützen.

Hilfe

- Überprüfen Sie, ob jeder im Team die ihm zugewiesene Gruppennummer verwendet.
- Stellen Sie sicher, dass der Empfänger und der Hacker ihre Programme starten und warten, bevor der Absender die Nachricht übermittelt.
- Denken Sie daran, dass der Absender versucht, sich beim micro:bit des Empfängers anzumelden und das **Passwort** für den micro:bit **des Empfängers** senden muss.