Passwortsicherheit (Hack)

Lernziele

- Komplexität von Passwörtern berechnen können.
- Sicherheit von Passwörtern einschätzen können.
- Stärken, Schwächen und Unterschiede zu Bruteforce- und Wörterbuch-Angriffen nennen.
- Hashcat-Befehle für Bruteforce und Wörterbuch-Angriffe verstehen.

Passwort als Hash

Passwörter im Klartext sollen möglichst nicht persistent (dauerhaft) gespeichert werden. Die Gefahr ist zu gross, dass ein Angreifer die Passwörter im Klartext auslesen kann (zum Beispiel über eine Injektion). Damit eine Software beim Login ein Passwort überprüfen kann, muss das Passwort aber in irgend einer Form für die Authentifizierung vorhanden sein. Es gibt hier also einen Widerspruch:

- Sicherheit: Das Passwort darf nicht dauerhaft gespeichert werden.
- Praxis: Das Passwort muss dauerhaft gespeichert werden, damit man beim Login das eingegebene Passwort vergleichen kann.

Dieser Widerspruch wird mit einem «Kompromiss» gelöst: Gespeichert wird nicht das Passwort, sondern ein «Abdruck» davon: der *Hash*. «Hash» oder «to hash» kommt aus dem Englischen und kann mit «Gehacktes» bzw. «hacken» übersetzt werden. Die Analogie ist die Folgende:

> *Das Passwort wird «durch den Fleischwolf gedreht» und kommt «zerhackt» wieder raus.*

Es gibt verschiedene Algorithmen um einen Hash zu erstellen. Sie können diese direkt auf einer Linux-Konsole ausprobieren. `MD5` ist beispielsweise ein älterer, sehr bekannter Algorithmus. Für sicherheitskritische Anwendungen sollte er nicht mehr gebraucht werden:

```
echo -n 'geheim' | md5sum      # e8636ea013e682faf61f56ce1cb1ab5c
echo -n 'sicher' | md5sum      # bb5c98f1331c50ea01d8d6fc54bf84d5
```

Viele Anwendungen setzen heute auf `SHA` mit einer entsprechenden Länge:

```
echo -n 'geheim' | sha256sum   # addb0f5e7826c857d7376d1bd9bc33c0c54479...
echo -n 'sicher' | sha256sum   # bdfccb90bbe91a2b3eed18c7280709a96fea8c...
```

Erklärung

Was passiert? Die Algorithmen «zerhacken» die Passwörter in eine zufällig wirkende Zeichenkette. Solange die gleiche Eingabe (also das Passwort) «zerhackt» wird, kommt immer das gleiche Ergebnis als Hash heraus. Für jede andere Eingabe soll aber (möglichst) immer ein anderer Hash heraus kommen. Daher lassen sich statt der Passwörter, die Hashes der Passwörter vergleichen. Es genügt also den Hash des Passwortes in der Datenbank zu speichern, statt das Passwort selbst. Vorteil: Wenn jemand die Passwort-DB stiehlt, bekommt er nur Hashes und keine Passwörter. Wichtig: Aus Hashes lässt sich in der Regel nicht auf die ursprüngliche Eingabe schliessen.

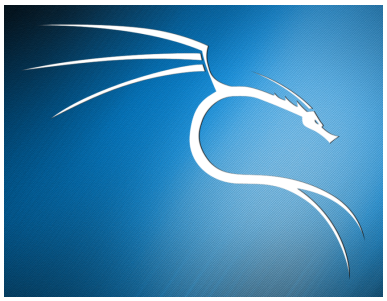
Achtung

Aber Vorsicht: Auch Hashes sind nicht 100% sicher. In diesem Kapitel werden Sie versuchen einige Hashes zu knacken! Im nächsten Kapitel werden Sie dann einige Techniken erlernen, um genau das zu verhindern.

Vorbereitung: Hashcat auf Kali Linux

Um Passwörter zu «knacken», bzw. «wiederherzustellen» wird einiges an Software benötigt. Lesen Sie sich dazu kurz ein und machen Sie die Schritte zur Vorbereitung.

Hashcat ist ein Softwareprojekt zur «Wiederherstellung» von verlorenen Passwörtern. Das Projekt ist unter der Seite `hashcat.net` erreichbar. Hashcat ist nicht das einzige quelloffene Programm in diesem Bereich.



Kali Linux ist eine Linux-Distribution, welche sich auf «Penetration-Testing» (Sicherheits-Tests) spezialisiert hat. Das Projekt ist unter der Seite `kali.org` erreichbar. Die Distribution bringt eine grosse Anzahl vorinstallierter Werkzeuge mit, unter anderem ist auch Hashcat vorinstalliert. Da die Installation von Hashcat nicht einfach ist, nutzen wir für diese Übung Kali Linux.

Aufgabe 1

Kopieren Sie *Kali Linux* von der Modulablage auf Ihre lokale Lernumgebung. Starten Sie das System. Der Benutzer ist `root`, das Passwort `toor`.

Testen Sie ob Sie Netzwerk haben und das Programm Hashcat installiert ist. Falls nicht, konfigurieren Sie das System noch entsprechend.

Aufgabe 2

Auf der Modulablage finden Sie die Dateien `passwordhashes.tgz` und `dictionary_german.tgz`. Die Dateien sollten bereits in Ihrem Kali Linux entpackt vorhanden sein. Falls nicht, legen Sie die Archive im Dateisystem Ihres Kali Linux ab und entpacken Sie diese. Schauen Sie sich kurz den Inhalt einiger Dateien an, wir brauchen diese später! Was sehen Sie in den Dateien?

.....

Hinweis

Hashcat ist eine hochspezialisierte Software welche die Hardware stark in Anspruch nimmt. Dies um möglichst gute Resultate zu erzeugen. Passwörter «knacken» benötigt Rechenpower pur! Hashcat kann über die Schnittstelle `OpenCL` parallel mehrere Grafikkarten als CPU zum Rechnen einzusetzen.

Bedenken Sie Folgendes: Sie arbeiten für diese Übung auf einer kleinen VM, ohne spezielle Treiber oder Hardware. In der Industrie stehen einiges grössere Maschinen für Sicherheitstests zur Verfügung! Entsprechend sind die in dieser Übung erzielten Resultate und Leistungen lediglich ein erster Einstieg in die Materie.

Komplexität eines Passwortes

Bevor Sie sich am Knacken von Passwörtern in der Praxis versuchen, sollten Sie ein Gefühl der Grössenordnung bekommen. Die Komplexität von Passwörtern lässt sich berechnen. Je komplexer ein Passwort, desto schwieriger ist es zu knacken.

Kombinationsmöglichkeiten von Zeichen

Jedes Passwort kann grundsätzlich über diese zwei Merkmale beschrieben werden:

- **Zeichensatz**: Ein Passwort wählt einige Zeichen aus einer vorgegebenen Liste aus. Beispiel: die Zeichen «qwert» werden aus dem Alphabet von Kleinbuchstaben als Passwort ausgewählt.
- **Länge**: Jedes Passwort hat eine Länge. Beispiel: Das Passwort «qwert» hat die Länge fünf.

Diese beiden Merkmale sind jeweils nicht geheim und damit grundsätzlich auch einem Angreifer bekannt. Obwohl ein Angreifer das Passwort eines Einzelnen nicht kennt, kann er bereits einige Aussagen treffen.

Ein einfaches Beispiel:

- **Zeichensatz**: Alphabet A-Z (Gross- & Kleinschreibung) sowie Ziffern 0-9.
- **Länge**: 8 bis 12 Zeichen.

Ein Angreifer kann nun bereits die Anzahl aller theoretisch möglicher Passwörter berechnen. Wenn die Grösse des Zeichensatzes (Anzahl Zeichen) z ist und die Länge des Passwortes l, dann lautet die Formel für die Anzahl möglicher Kombinationen k:

$$k = z^l$$

Aufgabe 3

Berechnen Sie die Anzahl möglicher Passwörter nach dem obigen Beispiel. (Tipp: Sie müssen die verschiedenen Längen beachten).

.....

.....

.....

Hinweis

Gängige Regeln wie «Das Passwort muss mindestens eine Zahl enthalten» machen die Berechnung noch komplizierter. Wir belassen es beim obigen Beispiel, wo das Passwort eine Zahl enthalten kann aber nicht muss. Dies genügt um ein grobes Gefühl für die Grössenordnung zu bekommen.

Aufgabe 4

Was bringt prinzipiell eine höhere Sicherheit: «Den Zeichensatz um X-Zeichen erweitern» oder «Die Passwortlänge um X erhöhen»? Warum?

.....

.....

.....

Achtung

Die momentane Empfehlung für sichere Passwörter ist mindestens 12 mal aus etwa 70 Zeichen auswählen. Das heisst der Suchraum muss über ca. $\mathbf{10^{22}}$ liegen, damit das Passwort sicher ist.

Wortkombinationen

Passwörter bestehen oft aus zusammengesetzten Wörtern statt zufällig gewählten Zeichen. Solche Passwörter verlieren aus verschiedenen Gründen schnell Ihre Komplexität und sind damit einfacher knackbar.

Ein Beispiel:

Ein Hacker weiss, dass der Benutzer Passwörter aus jeweils drei Wörtern bildet. Jedes Wort wird am Anfang gross geschrieben. (Der Benutzer bildet sich ein, das Passwort so sicherer zu machen, da auch Grossbuchstaben verwendet werden.) Hier drei Beispiel nach diesem Schema:

RegenschirmApfelBerg, SahneMilchButter, DerDieDas

Merkmale: Alphabet A-Z Gross- & Kleinschreibung. 9 - 20 Zeichen.

Nach vorherigem Vorgehen wären dies 62 Zeichen insgesamt. Dies gäbe folgende Rechung:

$\sum_{l=9}^{20} 62^l$ = *ca. 700 Quintilliarden.*[1] *Eine Zahl mit 35 Nullen!*

Also ziemlich sichere Passwörter? Nicht wirklich!

Sobald der Angreifer das Wissen über den Aufbau hat, kann er die Rechnung vereinfachen. Da die Passwörter aus Wörtern statt Buchstaben aufgebaut werden, sind Wörter die kleinste Einheit. Damit hat jedes Passwort nur eine Länge von drei! Der Zeichensatz wird dafür grösser, da es viel mehr Wörter gibt als Buchstaben im Alphabet. Der zentrale Wortschatz der Deutschen Sprache besteht gemäss Duden aus 70'000 Wörtern. Dies ergäbe folgende Anzahl möglicher Passwörter:

$70000^3 = 3.43x\mathbf{10^{14}}$ = *343 Billionen. Eine Zahl mit 14 Nullen, also bedeutend weniger!*

Wie Sie sehen sind diese Passwörter unsicher, obwohl bis zu 20 Buchstaben verwendet werden!

[1] http://mathe-abakus.fraedrich.de/mathematik/grzahlen.html

Passwörter knacken

Folgend werden einige Angriffs-Konfigurationen mit `hashcat` vorgestellt. Sie werden diese Konfigurationen für die Übungen auf der nächsten Seite benötigen!

Hinweis
Hängen Sie (nach dem Knacken) `--show` an die Kommandos an, um die Passwörter zu sehen.

Bruteforce

Erklärung
Der begriff *Bruteforce* kommt aus dem Englischen und kann mit «stumpfe Gewalt» übersetzt werden. Hier werden keine «schlauen Tricks» verwendet, sondern es wird einfach «dumm» jede erdenklich mögliche Kombination durchprobiert!

Hier ist ein beispielhafter Aufruf von `hashcat`:

```
hashcat -m 0 -a 3 geheim.txt ?l?l?l?l --force
```

- `-m 0`: Der Hash-Typ wird auf `md5` eingestellt.
- `-a 3`: Der Bruteforce-Modus wird mit diesem Parameter eingestellt.
- `geheim.txt`: Dateiname der Datei mit den Hashwerten.
- `?l?l?l?l`: Maske/Muster für die zu prüfenden Wertebereiche. Ein Fragezeichen steht für eine Stelle im Passwort, der Buchstabe für den Zeichentyp. Diese Maske bedeutet: «Teste alle Passwörter mit der Länge vier (4 mal `?`) und kleinen Buchstaben (`l`). Sie finden eine detailliertere Beschreibung dazu auf der Webseite des Programms.[2]
- `--force`: Führt den Angriff auch dann aus, wenn die Treiber nicht optimal konfiguriert sind.

Wörterbuch

Erklärung
Sie haben bereits gemerkt, dass die Komplexität von Passwörtern sehr schnell zunehmen kann, sobald diese lange genug sind und die Zeichenauswahl zufällig ist. Solche Passwörter können auch in der Industrie nur schwerlich geknackt werden. Der Suchraum ist einfach zu gross, als dass man alle Kombinationen durchprobieren könnte!

Daher versucht man «schlauere» Methoden als Bruteforce anzuwenden. Die meisten Passwörter bestehen nicht aus wirklich zufälliger Auswahl von Zeichen. Oft kommen z.B. Wörter darin vor. Daher versucht man längeren Passwörtern mit vordefinierten Wortlisten auf die Schliche zu kommen. Diese Wortlisten können sehr gross werden und viele Kombinationen abbilden. Hier kommt es nun tatsächlich auf das «Geschick» des Erstellers solcher Listen an. In dieser Übung verwenden wir eine Wortliste von `md5this.com`. Diese Liste ist weder besonders gross noch gut, reicht aber für eine erste Demonstration völlig aus.

```
hashcat -m 0 -a 0 geheim.txt /path/to/dictionary_german.dic --force
```

- `-a 0`: Der Modus für Wörterbuch-Angriff wird eingestellt.

[2] Hashcat-Notation: https://hashcat.net/wiki/doku.php?id=mask_attack#built-in_charsets

Praxis-Übung «Passwörter knacken»

In der folgenden Tabelle sind einige der Hash-Dateien gelistet, welche Sie vorher auf Ihr System kopiert haben. Aufgabe 3 ist vorgelöst.[3]

	Datei (md5.txt)	Zeichensatz	z	l	k	geknackt	Zeit
1)	4_smallletters	a-z		4			
2)	4_AlphaNum	a-zA-Z0-9		4			
3)	to4_AlphaNumSpecial	?l?u?d?s	95	1-4	82'317'120	100%	2-3 Sek.
4)	6_smallletters	a-z		6			
5)	8_smallletters	a-z		8			
6)	8_AlphaNum	a-zA-Z0-9		8			

Aufgabe 5

Ergänzen Sie die leeren Stellen der oberen Tabelle. Notieren Sie jeweils den korrekten `hashcat`-Befehl um das Problem anzugehen. Tipp: Ab Aufgabe 6 sollten Sie evtl. langsam ein «Wörterbuch» zu Hilfe nehmen...

1) .....

2) .....

3) hashcat md5_to4_AlphaNumSpecial.txt -m 0 -a 3 ?1?1?1?1 -1 ?l?u?d?s --force --increment
oder: md5_to4_AlphaNumSpecial.txt -m 0 -a 3 ?a?a?a?a --force --increment

4) .....

5) .....

6) .....

Aufgabe 6

Knacken Sie die Passwort-Hashes des Fulla-Servers!

.....

[3] Der Zeichensatz bei Aufgabe 3 ist in Hashcat-Notation angegeben.