Einige simple Wahrheiten über Passworte

Vor einiger Zeit las ich einen Artikel, der empfahl, das Hash-Verfahren für Unix-Passworte zu verändern und diese nicht mehr mittels MD5 zu verschlüsseln. Hintergrund war der Diebstahl von mehr als 90.000 Passworten bei der US-Regierungsberatungsfirma Booz Allen Hamilton, wo unsichere MD5-Hashes verwendet worden waren.

Das ist schon eine Zeit her – heutzutage vergeht kaum ein Monat, wo nicht über einen „Data Breach“ bei einer großen Firma berichtet wird.

Was sind diese „Hashes“ eigentlich?

Passworte werden aus Sicherheitsgründen meist nicht in Klarschrift abgelegt. Täte man das, könnte ein Angreifer, der sich Zugriff auf das System verschafft, direkt das Passwort lesen und für sich verwenden. Da die Zugangskennung für viele Websites die E-Mail-Adresse ist, kann man Zugriff auf weitere Konten verschaffen.

Eine verbesserte Variante wäre eine verschlüsselte Ablage des Passworts – dabei muss man jedoch berücksichtigen, dass eine direkte Entschlüsselung möglich ist. In manchen Systemen kommt deshalb eine Krypto-Hardware zum Einsatz, ohne deren Besitz die Chiffrate nicht entschlüsselt werden können, teilweise wird das eingesetzte Verfahren durch ein anbieterspezifisches „Pepper“ angereichert, ohne dessen Kenntnis man das Klarschrift-Passwort nicht zurückrechnen kann.

Stattdessen wird heutzutage das Passwort meist mittels einer Einweg-Verschlüsselung (das sogenannte „Hashing“) auf ein Chiffrat (den „Hash“) abgebildet und nur letzteres gespeichert. Es gibt aufgrund der Eigenschaften der eingesetzten Verfahren keinen direkten Weg zur Rückberechnung des Passworts aus dem Hash.

Dennoch kann ein eingegebenes Passwort bei Wiedervorlage auf Korrektheit geprüft werden, indem wieder die Einweg-Verschlüsselung angewendet und der resultierende Hash mit dem gespeicherten verglichen wird. Dazu muss ggf. auch nur der Hash übertragen werden, der ja selbst keinen direkten Rückschluss auf das Passwort zulässt. Dieses Verfahren hat gegenüber einer Zweiweg-Verschlüsselung den Nachteil, dass eine spätere Änderung des Hash-Verfahrens nur möglich ist, wenn man eine „sanfte“ Migration durchführt, d. h. bei korrekter Benutzereingabe den Hash ersetzt – eine einfache Batch-Umwandlung aller Hashes ist unmöglich.

Theoretisch könnten verschiedene Klarschrift-Passworte zum selben Hash führen und damit als „richtig“ erkannt werden, aber die Wahrscheinlichkeit dafür ist normalerweise gering. Die für das Hashing eingesetzten Verfahren versuchen dies sicherzustellen, indem die Menge der möglichen Hash-Werte sehr groß gewählt wird (z. B. 128 Bit bei MD5, d. h. 3,4 * 10^38) und die Abbildungen der Klarschrift-Passworte möglichst gleichmäßig (d. h. zufällig) in der Zielmenge verteilt werden, um Hash-Kollisionen zu verhindern.

Im Idealfall sollte die Hash-Funktion auch aufwendig zu berechnen sein, damit nicht durch schnelles „Ausprobieren“ alle möglichen Kombinationen von Buchstaben durchgetestet werden können.

In den Anfängen von Unix wurde als Hash-Verfahren DES mit einem sogenannten „Salt“ verwendet. Das Salt stellt sicher, dass der Passwort-Hash mit einer Art Zufallsfaktor beaufschlagt wird, damit das Chiffrat für gleiche Passworte jedes Mal anders lautet. Damit sind Massen-Angriffe per Rainbow-

Table ausgeschlossen, bei denen man die Chiffrate für alle Worte aus einem Wörterbuch schon vorab kennt und nur noch vergleichen müsste, was den Rechenaufwand massiv verkleinert.

Die anfangs in der Datei /etc/passwd abgelegten Passworte wurden später in eine nur durch einen privilegierten Benutzer lesbare Datei /etc/shadow ausgelagert, damit ein Angriff durch Kopieren und späteres Analysieren schon von vornherein vereitelt würde. Die Systemfunktion für die Passwortprüfung wurde mit einer künstlichen Verzögerung ausgestattet, damit insbesondere von außerhalb des Systems keine Brute-Force-Attacken durchgeführt werden können. Darüber hinaus wird nach drei Fehlversuchen eine längere Wartezeit erzwungen oder der Account sogar gesperrt.

In Windows wurden anfangs Passworte ebenfalls mit einem abgewandelten DES-Verfahren namens LANMAN abgebildet.

Zeit für ein paar große Zahlen

Die Berechnung von DES-Hashes war zunächst so langsam, dass lediglich Wörterbuch-Attacken durchgeführt werden konnten, die im Gegensatz zu Brute-Force-Angriffen den Suchraum stark verkleinern. Hieraus resultiert die oft gegebene Empfehlung, keine normalen Worte oder leichte Abwandlungen davon als Passwort zu benutzen, z. B. indem erzwungen wird, Passworte mit mindestens acht Zeichen, darunter jeweils mindestens Groß- und Kleinbuchstaben, Ziffern und/oder Sonderzeichen.

Diese Empfehlung ist problematisch: vorgezogen werden sollten längere Passworte, die dann auch nur aus Buchstaben und Leerzeichen bestehen dürfen, z. B. ganze Sätze oder deren Anfangsbuchstaben - oft gibt es aber unsinnigerweise Längenbeschränkungen auf z. B. 10 Zeichen.

Warum die Forderung nach langen Passworten besteht, wird schnell klar, wenn man sich die Anzahl möglicher Kombinationen und die daraus resultierende Rechenzeit bei angenommen 1 Million möglicher Tests pro Sekunde für das Erraten eines Passworts ansieht:

Art des Passworts	**Mögliche Kombinationen**	**Rechenzeit bei 1 Mio Tests/Sek**
5 Kleinbuchstaben	11.881.376	12s
8 Ziffern	100.000.000	100s
8 Großbuchstaben	208.827.064.576	2,4 Tage
8 Groß- und Kleinbuchstaben	53.459.728.531.456	1,7 Jahre
8 druckbare ASCII-Zeichen	6.634.204.312.890.620	210 Jahre
10 druckbare ASCII-Zeichen	59.873.693.923.837.900.000	1,9 Millionen Jahre
12 Groß- und Kleinbuchstaben plus Leerzeichen	491.258.904.256.726.154.641	15,5 Millionen Jahre

Wie man sieht, erhöht die Verlängerung des Passworts von 8 auf 10 Zeichen den Rechenaufwand viel mehr als die Hinzunahme weiterer Zeichenklassen.

Mit zunehmender Leistungsfähigkeit von Rechnerhardware wurde das DES-Verfahren unsicher, weil nun auch Brute-Force-Attacken ohne Suchraumbeschränkung möglich wurden. Man begann daraufhin, Passworte mit dem damals vermeintlich sichereren MD5-Verfahren zu hashen. Dazu wurden in die Unix-Passwort-Hashes zur Unterscheidung Präfixe eingebaut, um verschiedene Hash-Verfahren zu erkennen (sogenanntes „Modulares Crypt“ z. B. mit Präfix „1“ für MD5, „$2a$“ für Blowfish). Bei Windows vollzog sich eine Abkehr von DES-basierten LANMAN- hin zu NTLM-Hashes.

Inzwischen gilt auch MD5 als zu unsicher, man weicht meist auf SHA-Hashes aus. Die nächste Stufe steht auch bereits an: der Wettbewerb des NIST zum Thema „Post-Quantenkryptografie“ wurde gerade abgeschlossen.

Wörterbuchattacken

Heutzutage wird das Knacken von Passworten übrigens nicht mehr mittels einfacher Brute-Force-Attacken, sondern mittels vorgefertigter Wörterbücher gemacht. Dadurch werden „typische“ von Menschen oft benutzte Passworte verwendet, also z. B. eher „Snoopy“ als „noSypo“. Zwar deckt man so den Suchraum nicht zu 100% ab, jedoch erhöht sich die Trefferquote – und genau darum geht es.

Leider stehen derartige Sammlungen dank der Data Breaches der letzten Jahre in Hülle und Fülle zur Verfügung, die bekannteste Sammlung „Rockyou.txt“ umfasst inzwischen mehrere 100 Millionen Passworte mit Längen zwischen 6 und 20 Zeichen. Wie bereits gesagt: Passworte mit bis zu 6 Zeichen kann man ohne Probleme vollständig prüfen.

Das bedeutet: „Popocatepetl“ und „Narhallamarsch“ sind trotz Ihrer Länge vermutlich auch keine guten Passworte!

Das schwächste Glied

Problematisch sind Hashes u. a. deshalb, weil es davon mittlerweile oft mehrere parallel eingesetzte Varianten gibt. So finden sich auf einem Unix-System z. B. für einen Benutzer oft mindestens zwei verschiedene Hashes für den Unix-Account (z. B. Verfahren MD5 oder Blowfish) und für den SAMBA-Account (Verfahren NTLM und/oder das ältere LANMAN). Die Wahrscheinlichkeit ist hoch, dass die dazu gehörigen Passworte die gleichen sind. Falls ein Angreifer Zugriff auf das System hat, kann er sich das Hash-Verfahren aussuchen, das die beste Performance verspricht. Somit bestimmt das schwächste eingesetzte Hash-Verfahren die Dauer bis zum Kompromittieren des Passworts.

Mittels Tools wie Hashcat können heutzutage durch den Einsatz von handelsüblichen Grafikkarten Rechenleistungen allerdings nicht mehr wie oben angenommen 1 Millionen, sondern ca. **10 Milliarden** Windows-NTLM-Hashes pro Sekunde erzielt werden. Das bedeutet bei Verwendung von beliebigen Passworten mit einer Maximallänge von 8 Zeichen eine maximale Rechenzeit von nur 11 Minuten bis zur Errechnung des Windows-Passworts!

Die Auswirkung von gesteigerter Rechenleistung ist hier unwillkommen: Konnte man früher z. B. unter Windows wenigstens durch den Einsatz von EFS bestimmte kritische Dateien wirklich sicher wähnen, ist dies heute bei kurzen Passworten nicht mehr der Fall: Bis vor einiger Zeit hätte ein Angreifer mit physischem Zugriff auf die Festplatte des Rechners zwar das Passwort des Benutzers zurücksetzen können und damit Zugriff auf alle dem Benutzer zugängliche Dateien erhalten. Der Zugriff auf die EFS-verschlüsselten Dateien wäre ihm jedoch verwehrt geblieben. Heute würde er

einfach das (zu kurze) Windows-Passwort aus dem Hash errechnen und hätte damit kompletten Zugriff.

Vorläufiger Ausweg ist die Nutzung von Passworten, die so lang sind, dass auch mit moderner Hardware kein Knacken möglich ist. Natürlich dürfen diese auch nicht in einem Wörterbuch zu finden sein. Daneben sollten nur Hashes verwendet werden, die noch als sicher gelten, z. B. Blowfish (bei SHA-1 können ca. 3,5 Milliarden Passworte pro Sekunde berechnet werden und dort werden auch schon prinzipielle Bedenken angemeldet, so dass z. B. bei TLS-Zertifikaten heutzutage nur noch SHA-2 verwendet werden soll).

Die moderneren Hash-Verfahren werden teilweise noch künstlich verlangsamt, indem die Anzahl der „Runden“ erhöht wird. Da hierdurch bei vielgenutzten Systemen auch teure (Nutz-)Rechenzeit verbraucht wird, weicht man neuerdings manchmal auf Platz- statt Rechenzeitkomplexität aus, indem Verfahren genutzt werden, die ein Mindestmaß an Speicherplatz benötigen (Beispiel: Argon2). Versucht man dann z. B. auf tausenden Rechenkernen auf einer Grafikkarte parallel viele Hashes zu berechnen, reicht zwar die Rechenleistung, nicht aber der verbaute Speicher dazu aus.

Wenn man beispielsweise 10000 Kerne und 10 GByte RAM annimmt, Argon2 aber mit einem Mindestspeicher von 100 MByte eingesetzt wird, können nur 100 Kerne gleichzeitig arbeiten, was die Effizienz dieser Hardware um den Faktor 100 reduziert.

Allerdings wird das längst noch nicht bei allen eingesetzten Systemen getan und als Anwender kann man das meist weder erfahren (Security by Obscurity) noch sollte man es voraussetzen.

Ein Passwort für alle Einsatzzwecke?

Eine Warnung also für alle, die immer das gleiche Passwort benutzen, egal ob im Internet (z. B. Foren), für geschützte Programme (z. B. Internet-Banking), für Netzwerk-Appliances (z. B. Internet-Router) oder für die Verschlüsselung von Backups: **Sie begeben sich weitgehend in die Hände derjenigen, denen sie ihr Passwort anvertrauen.**

Schließlich kann man sowieso nie sicher wissen, in welcher Form (Klarschrift oder Hash) das überlassene Passwort gespeichert wird. Tatsache ist, dass z. B. viele Internet-Foren dieselbe Software einsetzen (z. B. phpBB), in dem zumindest überhaupt Hashes verwendet werden (wenn auch bis phpBB 2.0 nur das inzwischen als unsicher geltende MD5). Falls aber in einer solchen Software ein Sicherheitsloch steckt, sind sofort sämtliche damit betriebenen Sites gefährdet. Bei vorhandener Möglichkeit zum Einbruch in ein System und Diebstahl der Daten könnte also bereits früher als beim späteren Bekanntwerden der Lücke auch immer das dort verwendete Passwort bereits kompromittiert worden sein – nämlich, wenn es in Klarschrift oder zweiweg-verschlüsselt abgelegt wurde. Solche noch nicht weithin bekannten Lücken werden 0-Day-Exploits genannt und teuer gehandelt.

Zusammen mit den heutigen Möglichkeiten muss man also immer davon ausgehen, dass das Passwort offengelegt ist - selbst, falls Hashes anstelle von Verschlüsselung zum Einsatz kommt. Wer also bislang nur meinte, dass Datendiebstähle in Online-Auftritten von Sony, KM-Elektronik, Mindfactory, Ashampoo oder Adobe (um nur einige exemplarisch zu nennen) lediglich zur Folge haben, dass jetzt vermehrt Spam im Postfach vorgefunden wird, könnte sehr bald eines Besseren belehrt werden.

P.S.: Im Fall von Adobe hat sich diese Befürchtung schon bewahrheitet!

P.P.S.: Vor einiger Zeit hat das BKA in entsprechenden Untergrund-Foren einen Datenbestand mit 500 Millionen Datensätzen inklusive E-Mail-Adresse und Klarschrift-Passworten entdeckt, davon stammten allein 50 Millionen von DE-Domains. Allein die 2019 bekannt gewordene „Collection #1", ebenfalls eine Sammlung von E-Mail-Adressen, umfasst 773 Mio E-Mail-Adressen und 21 Mio. Passworte. Weitere Leaks sind hier aufgeführt:
https://de.wikipedia.org/wiki/Liste_von_Datendiebst%C3%A4hlen

P.P.P.S.: Angesichts der inzwischen praktisch wöchentlich auftretenden Passwort-Lecks gibt es sogar Websites, auf denen man prüfen kann, ob der eigene Account eventuell gehackt wurde. Zwei Beispiele sind https://haveibeenpwned.com/ und https://sec.hpi.de/ilc/. Man sollte sich aber gut überlegen, wo man ggf. seine E-Mail-Adresse oder Passwort „testen" lässt: Solche Sites könnten potentiell selbst zum Einsammeln missbraucht werden.

Ein Beispiel für Verschlüsselung von Passworten in Anwendungssoftware:

Die Firma VISA versendete im Firmenkundengeschäft monatlich per E-Mail-Umsatzlisten in Form von passwort-geschützten PDFs. Ich hatte das Passwort dafür vergessen. Die Rettung brachte die Software „Advanced PDF Password Recovery" von Elcomsoft: Dieses Programm berechnete innerhalb von 4 Stunden ein Passwort für die Dateien. Nota bene: „ein" Passwort, weil das errechnete Passwort nach meiner Erinnerung nicht dasjenige war, welches ich ursprünglich vergeben hatte. Offensichtlich wird bei PDF zur eigentlichen Verschlüsselung ein Hash des Passworts benutzt, so dass mehrere unterschiedliche Passworte funktionieren können.

Ähnliche Entschlüsselungsprogramme existieren ebenfalls für die Microsoft-Office-Familie, und auch dort lässt sich die Performance durch Einsatz geeigneter Hardware massiv steigern.

Man kann festhalten, dass die Entwickler von Anwendungsprogrammen auch nur mit Wasser kochen und aufgrund der Tatsache, dass die meisten nicht quelloffen sind, kann man nur schwer abschätzen, welchen Grad an Qualität die jeweils verwendete Verschlüsselung hat. Dies gilt umso mehr für Passwort-Manager, weil dort prinzipbedingt eine Entschlüsselung möglich sein muss.

Was folgt aus diesen Überlegungen?

Man sollte also unbedingt für verschiedene Systeme bzw. Zwecke unterschiedliche Passworte verwenden. Zur Vereinfachung des Umgangs damit gibt es verschiedene Ansätze, z. B. kann man sein Standardpasswort modifizieren, indem man den ersten und letzten Buchstaben des Namens der Webseite oder des Programms einstreut, z. B. „ePASSWORTy" für EBay, „aPASSWORTn" für Amazon. Wichtig ist: Das verwendete System sollte man nur selbst kennen.

Komplexität versus Länge

Dabei gilt es natürlich zu beachten, dass das resultierende Passwort von ausreichender Qualität ist. Hierbei schlägt Länge immer Komplexität, denn ein Passwort, das, wie leider oft empfohlen bzw. erzwungen, zwar eine Kombination aus Groß- und Kleinbuchstaben, Ziffern und Sonderzeichen beinhaltet, aber nur 10 Zeichen lang ist, lässt sich schneller knacken als ein Satz mit mindestens 12 Buchstaben (siehe Tabelle oben). Ausnahmen sind natürlich tatsächlich vorkommende Worte, die man ein einem Wörterbuch finden kann.

Dies ist nicht zu verwechseln mit Empfehlungen, die den uralten Vorgaben des NIST ähneln: der Autor der Regeln, Bill Burr, hat gegenüber dem Wall Street Journal zu Protokoll gegeben, dass er diese Vorgaben heute bereut. Tatsache ist, dass diese eher kontraproduktiv sind, weil man sich solch komplexe Passworte nicht merken kann und diese deshalb aufschreibt oder doch immer das gleiche Passwort verwendet. Durch Untersuchungen aus Millionen von echten Passworten aus Datenlecks wurde klar, dass die meisten davon mittels einfacher Bildungsregeln ebenfalls leicht automatisiert zu knacken sind, wenn sie auf einem Wort aus einem Wörterbuch basieren.

Die Brisanz des Problems, dass die Hashes verschiedener Accounts mit verschiedenen Verschlüsselungen abgelegt werden, wird durch moderne Single-Sign-On-Ansätze wie LDAP sogar noch verstärkt: Da dort die Credentials für verschiedene Systeme abgelegt werden müssen, sind oft genug sämtliche Hashes für Accounts abrufbar, denn jedes authentifizierende Subsystem benötigt ja „seinen“ Hash-Typ. Wenn, wie bei Booz Allen Hamilton, unter den eingesetzten Verfahren auch leicht zu knackende Hashes sind, ist das Ergebnis ebenso klar wie fatal.

In den frühen 90er Jahren hat der Autor dieses Artikels mehrfach Wörterbuchattacken auf Unix-Passwort-Hashes vorgenommen, um unsichere Passworte zu finden. Deren Quote lag erfahrungsgemäß bei ca. 30%. Heute werden Passworte oft bereits beim Ändern gegen ein Wörterbuch verprobt, was angesichts der Möglichkeit zu einer erschöpfenden Brute-Force-Suche allerdings keinen wesentlichen Vorteil mehr bringt. War damals noch eine privilegierte Zugriffsmöglichkeit auf die Datei /etc/shadow für die Hashes notwendig, ist ein Auslesen bei vielen LDAP-Anwendern heute sogar einfacher zu bewerkstelligen.

Passworte regelmäßig wechseln?

Die häufig in Firmen vorgeschriebene Vorschrift, das Passwort regelmäßig zu wechseln, führt übrigens anstelle von erhöhter Sicherheit eher dazu, dass man die Passwörter notieren muss, um sie sich merken zu können – oft genug werden die Notizzettel unter der Schreibtischauflage versteckt. Tatsächlich führt ein etwaiger Zwang zu tatsächlichem Wechsel (technisch durch Führung einer Liste von vorangehenden Passworten realisiert) meist nur dazu, dass die Nutzer Passworte wie „Sommer1“, „Sommer2“ usw. verwenden.

Obwohl also diese Regel noch oft anzutreffen ist und Abweichungen davon auch bei manchen aktuellen Sicherheitsprüfungen unsinnigerweise noch immer moniert werden, ist sogar das Bundesamt für Sicherheit in der Informationstechnik nach langen Jahren von dieser Empfehlung abgerückt (siehe 2020er-Ausgabe des BSI-Grundschutz-Kompendiums).

Daumenregeln

1. Passworte sollten aus mindestens 10, besser 12 oder mehr Zeichen bestehen, bei denen der gesamte druckbare Zeichensatz einbezogen werden kann (d. h. Groß- und Kleinbuchstaben, Sonderzeichen und Ziffern bunt gemischt). Länge ist allerdings der Vorzug vor Komplexität zu geben, insbesondere, wenn man sich längere Passworte anhand von Sätzen besser merken kann. Benutzer sollten lange Passworte wählen, Anbieter sollten sie ermöglichen.

2. Passworte sollten in keinem denkbaren Wörterbuch vorkommen.

3. Anwender sollten unterschiedliche Passworte für verschiedene Systeme und Zwecke verwenden. Das ist unpraktisch, muss aber sein. Mittels individueller Regeln lassen sich

solche Passworte leichter merken. Die Alternative, nämlich sogenannte „Passwort Safes“ wie KeePass, werfen ihrerseits wieder Fragen nach der Sicherheit auf (z. B. mangelnde Prüfbarkeit von kommerziellen Produkten mangels offenem Quellcode), genauso wie das Abspeichern von Passworten im Browser selbst oder das schriftliche Notieren.

4. Übrigens: Gerade, wenn der Quellcode solcher Tools offen liegt, sind ggf. Angriffe durch Schadsoftware umso einfacher. Die Verwendung von Passworten sollte, wenn möglich, ganz vermieden werden (z. B. durch Public-Key-Verfahren). Falls nicht darauf verzichtet werden kann, sollte die Übermittlung ausschließlich über gesicherte Wege erfolgen (d. h. verschlüsselt via TLS o. ä.). Die verwendeten Endgeräte müssen frei von Viren und Trojanern und Überwachungssoftware sein (bei Rechnern in Internet-Cafés z. B. nicht sicherstellbar).

5. Vorzugsweise sollten auf den zu sichernden Systemen nur sichere Hash-Verfahren eingesetzt werden, d. h. insbesondere nicht MD5 oder DES, inzwischen auch nicht mehr SHA-1, sondern Argon2 mit hohen Speicheranforderungen. Darauf hat der Benutzer allerdings meist keinen Einfluss, außerdem werden veraltete Verfahren zur Sicherstellung von Rückwärts-Kompatibilität oft noch immer eingesetzt.

6. Wo immer es aufgrund des Risikos sinnvoll erscheint, sollten Anbieter Zwei-Faktor-Verfahren einsetzen. Bei Transaktionen auf Zahlungsverkehrskonten ist dies in der EU inzwischen sogar gesetzliche Vorgabe (PSD2). Anwender sollten optional angebotene Zwei-Faktor-Verfahren (z. B. FIDO oder TOTP) nutzen.

Fazit

Wir leben in modernen Zeiten und müssen uns daran gewöhnen, dass alte Weisheiten über die Wahl von Passworten und die Sicherheit von Daten vor dem Hintergrund aktueller Entwicklungen keinen Bestand mehr haben. Jeder Anwender sollte selbst durch geeignete Wahl von Passworten und Einhaltung bestimmter Grundregeln einer Kompromittierung vorbeugen.