

**Passwortsicherheit im Praxistest**

**Felix Niederer**

Passwortsicherheit im Praxistest

Facharbeit von Felix Niederer

3. Schuljahr, 1. Semester 2017/18

Fachrichtung Physik mit angewandter Mathematik

am Gymnasium der Kantonsschule am Burggraben, St. Gallen

8. Januar 2018

# Abstract

Den meisten Menschen, die einen Computer benutzen ist bewusst, dass die Verwendung von sicheren Passwörter wichtig ist, um von einem potentiellen Angriff geschützt zu sein. Den meisten ist auch klar, dass komplexe und lange Passörter am besten sind. Was hingegen vielen nicht bewusst ist, dass die Sicherheit der Passwörter nicht nur von der Komplexität und Länge, sondern auch vom Inhalt abhängig ist.

Es existieren wörterbuchbasierte Angriffsmethoden, die sehr effizient sind, da sie die Passwörter mit Wörtern aus Listen abgleichen und diese auch nach vorgegeben verändern können (umstellen, Gross-/Kleinschreibung verändern, usw. ). Daher bieten Passwörter aus Fantasiewörtern oder vom Computer generierte Passwörter einen markanten Sicherheitsvorteil.

Der Gegenstand dieser Arbeit soll beweisen, dass die von Menschen generierten Passwörter schneller und erfolgreicher geknackt werden können.

Das Forschungsergebnis zeigt auf, dass es bei zufällig generierten Wörter nicht nur länger dauerte, bis sie geknackt werden konnten, sondern dass auch der Erfolg bei einigem Angriff geringer war.

Daraus ist zu schliessen, dass Passwörter nicht nur aus Sonderzeichen, Zahlen und Buchstaben bestehen sollten sondern dass es ebenso wichtig ist, Fantasiewörter zu verwenden.

# Inhaltsverzeichnis

[1 Sichere Passwörter 5](#_Toc502910234)

[1.1 Was ist ein Hash und warum ist es er eines der wichtigsten Werkzeuge in der Passwortsicherheit? 5](#_Toc502910235)

[1.2 Wo können uns starke Passwörter schützen? 5](#_Toc502910236)

[1.3 Wo können uns starke Passwörter nicht schützen? 6](#_Toc502910237)

[1.4 Definition starkes oder sicheres Passwort 6](#_Toc502910238)

[1.5 Definition schwaches oder unsicheres Passwort 6](#_Toc502910239)

[2. Wörterbuchbasierte Angriffe 7](#_Toc502910240)

[2.1 Problemstellung 7](#_Toc502910241)

[2.2 Lösungsansatz 7](#_Toc502910242)

[2.4 Praxis 7](#_Toc502910243)

[2.5 Ergebnis 8](#_Toc502910244)

[3. Zusammenfassung 10](#_Toc502910245)

[4. Anhang 11](#_Toc502910246)

[4.1 Das zur Einteilung und zum Hashen verwendete Programm 11](#_Toc502910247)

[4.2 Literaturverzeichnis 12](#_Toc502910248)

# 1 Sichere Passwörter

## 1.1 Was ist ein Hash und warum ist es er eines der wichtigsten Werkzeuge in der Passwortsicherheit?

Stellen wir uns vor Webseiten würden unsere Passwörter so speichern wie wir sie eintippen, wenn wir unser Konto erstellen. Nun könnte jeder der Zugriff auf die Datenbank hat, sei es berechtigt (zum Beispiel ein Systemadministrator) oder unberechtigt, sofort auf jedes Benutzerkonto zugreifen, sogar ohne dass dies vom Benutzer bemerkt wird. Da er einfach das Passwort auslesen kann und zum Beispiel auf der Webseite eintippen.

Damit dies nicht geht, sollte (und tut dies auch meist) ein Dienst die Passwörter in Hashform speichern. Zu diesem Zweck wird eine kryptographische Hashfunktion (oder auch kollisionsresistente Einwegfunktion genannt) auf das Passwort angewendet. Eine kryptographische Hashfunktion nimmt wie jede Funktion eine Eingabe und produziert eine Ausgabe. Sie funktioniert jedoch nur in eine Richtung, sprich man kann von der Eingabe eindeutig auf die Ausgabe schliessen, jedoch nicht andersherum. Gibt man nun beim Anmelden sein Passwort ein wird auf dieses nun die gleiche Hashfunktion angewendet und der Output wird mit dem gespeicherten Hash abgeglichen. Da eine Hashfunktion bei gleicher Eingabe immer die gleiche Ausgabe produziert, sind die verglichenen Werte, sofern das eingegeben Passwort korrekt ist, gleich. Gibt man ein falsches Passwort ein, sind die Hashes nicht gleich.

Verschafft sich nun jemand Zugriff auf die gespeicherten Hashes, kann er sich damit nicht direkt anmelden, da er nicht weiss welche Eingabe diesen exakten Hash produziert. Ihm bleibt nichts anderes übrig als mögliche Eingaben durchzuprobieren und zu sehen welche diesen Hash produziert. (Pornin, 2011) Dies ist umso umständlicher oder gar unmöglich je stärker / sicherer ein Passwort ist. Meist geht dies Hand in Hand mit einem breiten Schlüsselraum. Der Schlüsselraum bezeichnet den Satz aller möglicher Zeichen, welche für die Verschlüsselung der Daten verwendet wurden. (Monaghan, Gopinathan, Naughton, & Sheridan, 2007, S. 1) In diesem Fall, die für die Passwörter Verwendeten Zeichenkategorien, wie zum Beispiel «Zahlen», «Zahlen und Buchstaben» oder «Zahlen, Buchstaben und Sonderzeichen»

## 1.2 Wo können uns starke Passwörter schützen?

Ich möchte hier einige Beispiele geben, wie Passworthashes in falsche Hände geraten können und bei welchen uns stärkere Passwörter auch schützen können, da nur die Hashes gestohlen werden können.

### Passwörter für Webseiten

Passwortdatenbanken können von Hackern geklaut werden, dies ist gerade bei schlecht programmierten Webseiten ein Problem. Ein Beispiel für eine solche Attacke wäre die «SQL injection» bei welcher die Differenzierung des Computers zwischen Nutzereingaben und Befehlen ausgenutzt wird. So kann einem Computer z.B. eine einfache Suchfeldeingabe als Befehl zum Exportieren einer kompletten Datenbank verkauft werden. (Halfond, Viegas, & Orso, 2006, S. 1f)

Ein weiteres Problem sind Mitarbeiter, welche in grossen Unternehmen, problemlos unbemerkt einen Blick in diese Datenbank werfen können.

### Passwörter für Benutzerkonten von eignen PCs oder Netzwerken.

Auch auf Computer werden die Passwörter in Hashform gespeichert. Diese können jedoch teils auf verschiedene Arten extrahiert werden. Zum Beispiel kann sich ein zweiter Computer mit dem Opfercomputer verbinden und sich als Netzwerkkarte präsentieren Diese werden meist auch im gesperrten Zustand eingebunden, der Angreifer kann nun den Datenverkehr abfangen und warten, bis ein Hash übermittelt wird.

## 1.3 Wo können uns starke Passwörter nicht schützen?

Bei allen Angriffen bei welchen die Passwörter direkt gestohlen werden. Dies geschieht meistens nicht beim Betreiber einer Webseite, sondern direkt auf dem Computer der Opfer. Einige Beispiele:

Bei sogenannten «Phishing» Angriffen wird dem Nutzer eine Webseite untergeschoben die ihm suggeriert sie wäre die Webseite eines anderen populären Webdienstes, mit dem Ziel, dass das Opfer seinen Benutzernamen und sein Passwort dort einträgt. Die untergeschobene Webseite wird jedoch vom Angreifer kontrolliert. (Plößl, Federrath, & Nowey, 2005, S. 1)

Eine andere Art von Angriff ist der «Keylogger». Dies ist entweder ein Gerät, welches zwischen Tastatur und Computer des Opfers geschaltet wird oder eine Software. Ein Keylogger zeichnet alle Tastatureingaben des Nutzers auf, auch wenn dieser Passwörter eintippt. (Ladakis, Koromilas, Vasiliadis, Polychronakis, & Ioannidis, 2013, S. 1)

## 1.4 Definition starkes oder sicheres Passwort

Wenn ich von starken oder sicheren Passwörter rede, meine ich Passwörter welche bei einem aktuell als sicher betrachten Hashverfahren sehr viel Zeit in Anspruch nehmen, um geknackt zu werden (auch für Angriffe, bei welchen sehr viel Rechenleistung zu Verfügung steht). Dies beinhaltet alle Massnahmen welche getroffen werden können um den Vorgang zu beschleunigen.

## 1.5 Definition schwaches oder unsicheres Passwort

Wenn ich von schwachen oder unsicheren Passwörter rede, meine ich Passwörter, welche bei einem aktuell als sicher betrachten Hashverfahren nicht viel Zeit in Anspruch nehmen um geknackt zu werden. Dies beinhaltet alle Massnahmen welche getroffen werden können um den Vorgang zu beschleunigen.

# 2. Wörterbuchbasierte Angriffe

## 2.1 Problemstellung

Durch wörterbuchbasierte Methoden zum Knacken von Passworthashes ist die Sicherheit eines Passwortes mittlerweile nicht nur von der Länge und Verwendung eines möglichst breiten Schlüsselraumes abhängig, sondern auch von der Abstraktheit. Bei wörterbuchbasierten Angriffen wird der Hash über eine vorgefertigte Liste überprüft. Um die Erfolgschancen zu erhöhen wird ein Regelsatz angewendet, basierend auf der Erfahrung, wie die meisten Menschen ihre Passwörter aufbauen, an welcher Stelle Sonderzeichen oder Zahlen eingesetzt werden müssen. Es können auch verschiedene Wörter zusammengesetzt oder einzelne Zeichen ersetzt werden. Dies erhöht die Geschwindigkeit, mit welcher ein Hash geknackt wird massiv. Auch bei sicheren Hashes wie dem SHA256 Hash. (Narayanan & Shmatikov, 2005)

Dies führt dazu, dass auch von den meisten Plattformen als sicher eingestufte Passwörter, gerade wenn eine nicht mehr aktuelle Hashmethode verwendet wird, nicht sicher sind.

## 2.2 Lösungsansatz

Da wörterbuchbasierte Angriffe auf Listen mit bereits bekannten Passwörter und tatsächlichen Wörterbüchern (daher der Name) zurückgreifen, sowie Regelsätze basierend auf die Art wie die meisten Menschen ihre Passwörter aufbauen, basieren, sollten Passwörter mit vergleichbarer Länge und Schlüsselraum welche weder in einem Wörterbuch stehen noch einem bekannten Schema folgen, bedeutend sicherer sein. Dabei spielt es für den Computer keine Rolle, ob diese sprachähnlich oder völlig zufällig sind (Dies spielt später in der Praxis noch eine wichtige Rolle). Wenn wir uns längere Fantasiewörter ausdenken sollten diese daher bedeutend sicherer sein als existierende Wörter.

Für Passwörter welche nicht aus bekannten Wörtern bestehen sollte es daher länger dauern um geknackt zu werden und eine bedeutend niedrigere Erfolgsrate aufzeigen geknackt zu werden.

## 2.4 Praxis

Um festzustellen, ob und wenn ja, wie weit dem Computer unbekannte Passwörter schwerer fallen zu knacken, habe ich 3,3 Mio. Passwörter, welche aus bereits bekannten Listen stammen, gesammelt. Diese stammen von weakpass.com. Bei den Listen handelt es sich nur um Listen von Passwörtern von Menschen.

Ich habe die Passwörter mit einem Programm[[1]](#footnote-1) sortiert und in verschiedene Kategorien eingeteilt. Die Kategorien sind nach den drei Schlüsselräumen Zahlen, Zahlen und Buchstaben, sowie Zahlen, Buchstaben und Sonderzeichen geordnet. Zusätzlich habe ich noch die Mindestanforderungen an Passwörter von Google, Microsoft und der Raiffeisenbank als Kategorien gewählt.

Als Vergleichsgruppe habe ich, den jeweiligen Kategorien in Anzahl und Einteilungskriterien entsprechende, zufällige Passwörter generieren lassen. Ob man sich schwer zu merkende zufällig generierte Zeichenfolgen oder neue, sich selbst ausgedachte Fantasiewörter verwendet spielt keine Rolle, es kommt nur darauf an, dass die Zeichenketten noch in keinem Wörterbuch erfasst sind.

Danach wurden die Passwörter gehasht. Ich habe die Hashmethoden MD5 gewählt. Diese ist nach aktuellen Standards nicht mehr sicher und sollten in der Praxis nicht verwendet werden, aus Zeitgründen war dies jedoch unumstösslich. Zudem spielt die absolute Zeit keine Rolle, zu beachten ist der relative Zeitunterschied der beiden Gruppen (menschliche und zufällige Passwörter) sowie die Erfolgsrate.

Die gehashten Passwörter habe ich anschliessend von dem Programm Hashcat (Hashcat-Team, 2017) knacken lassen. Dies ist ein Programm welches zum Knacken von Hashes auf im grosse Stil ausgelegt ist. Es kann legal aus dem Internet heruntergeladen werden. Gerade der einfache Zugang zu solchen Werkzeugen macht es wichtig sichere Passwörter zu verwenden.

Als Angriffsmethode habe ich die standardwörterbuchbasierte Methode verwendet. Als Regelsatz habe ich die von Hashcat zur Verfügung gestellten «best64» Regeln verwendet, da diese eine gute Balance zwischen Geschwindigkeit und Erfolgsrate bieten. Als Wörterbuch kam die «crackstation-human-only» (Hornby) zum Einsatz. Als Hardware kam eine RX 480, eine R9 390 sowie ein Xeon E5-2670 zum Einsatz, dies entspricht etwa der Leistung von 2 durchschnittlichen Gaming Computer. Jedoch spielt die Leistung, aus denselben Gründen wie Hashmethode, keine Rolle.

## 2.5 Ergebnis

|  |  |  |
| --- | --- | --- |
|  | Von Menschen generiert | Computergeneriert |
| Zahlen | 2,3 min / 61,63% | 3,6 min / 28.16% |
|  |  |  |
| Zahlen, Buchstaben | 1.97 min / 24,15% | 1,3 min / 3,84% |
|  |  |  |
| Zahlen, Buchstaben, Sonderzeichen | 0,92 min / 12,49% | 0,92 min / 1,39% |
|  |  |  |
| Raiffeisenbank | 0,90 min / 9,68% | 0,90 min / 0,00% |
|  |  |  |
| Google | 19,17 min / 39,59% | 0,93 min / 0.00% |
|  |  |  |
| Microsoft | 1,37 min / 24,01% | 0,9 min / 0.00% |

Tabelle 1 Die Zeitanzeige steht für die Zeit die Hashcat benötigte, die Prozentzahl steht für den Anteil der Liste der erfolgreich geknackt wurde.

Die Zeiten und Erfolgsraten zeigen deutlich, dass die computergenerierten Passwörter um einiges sicherer sind, als die von Menschen generierten.

Von den von Menschen generierten Passwörtern wurde im Durschnitt 26,9% gefunden, bei den computergenerierten nur 3,2%. Gerade bei den längeren Passwörtern (Microsoft, Google, Raiffeisenbank) wurden, bei den computergenerierten kein einziges mehr gefunden, und das, obwohl zum Beispiel die Google Bedingungen nicht einmal zwingend Sonderzeichen oder Zahlen enthalten.

Daraus kann man schliessen, dass Passwörter zu wählen, welche kein existierendes Wort enthalten bedeutend sicherer ist. Daher ist die Lösungsvariante auf das Problem des Sicherheitsverlustes durch wörterbuchbasierte Angriffe, durchaus sehr effektiv.

Der Vergleich zwischen den von Menschen generierten- und computergenerierten Passwörter zeigt auch, dass es wichtiger ist Passwörter zu wählen welche keine real existierenden Wörter enthalten, als nur Zahlen, Buchstaben und Sonderzeichen im Passwort zu integrieren. Dies sieht man daran, dass die computergenerierten Passörter auch bei einem schmaleren Schlüsselraum den von Menschen generierten teils noch überlegen waren, wie zum Beispiel in den Kategorien «Zahlen, Buchstaben» der computergenerierten und «Zahlen, Buchstaben, Sonderzeichen» der den von Menschen generierten Passwörter. Hier haben die computergenerierten Passwörter eine bedeutend geringere Erfolgsrate, obwohl sie einen schmaleren Schlüsselraum aufweisen.

Die Ergebnisse der computergenerierten Passwörter zeigen zudem sehr schön auf, dass es hilft den Nutzer zu zwingen komplexere Passwörter zu verwenden, da die Erfolgsrate indirekt proportional zum Anstieg der Voraussetzungen steht. Die Kategorien mit den höchsten Bedingungen, «Raiffeisen» (Goss-, Kleinbuchstaben, Zahlen, Sonderzeichen und eine Minimallänge von 6 Zeichen) und «Zahlen, Buchstaben, Sonderzeichen» haben mit Abstand die geringste Erfolgsrate. Ich denke dies ist auf die Verwendung von Sonderzeichen zurückzuführen. Anscheinend senken sie bei den von Menschen generierten Passwörtern die Erfolgsrate stark. Dies ist jedoch immer noch kein Vergleich zu den computergenerierten Passwörtern.

Weitere Tests mit grösseren Regelsätzen, wie zum Beispiel den ebenfalls von Hashcat zur Verfügung gestellten Regelsatz «dive», wären nötig um zu sehen ob diese ein gleichmässigeres Resultat produziert. Dies war mir aus Zeitgründen nicht möglich, da grössere Regelsätze mehr Zeit benötigen.

# 3. Zusammenfassung

Um festzustellen, ob und wenn ja, wie weit dem Computer unbekannte Passwörter schwerer fallen zu knacken, habe ich 3,3 Mio. als Vergleichsgruppe, den jeweiligen Kategorien entsprechend in Anzahl und Einteilungskriterien, zufällige Passwörter vom Computer generieren lassen.

Danach wurden die Hashes mit einem Programm namens Hashcat (Hashcat-Team, 2017) geknackt. Wichtig sind hierbei vor allem die Erfolgsrate welche Hashcat mit einer wörterbuchbasierten Attacke, dem mitgelieferten Regelsatz «best64» und einer Passwortliste (Hornby) erreicht.

Nach dem Vergleichen der Erfolgsraten ist klar, dass die computergenerierten Passwörter um einiges sicherere sind, als die von Menschen generierten. Dies zeig, dass es nicht nur wichtig ist Passwörter zu wählen welche Sonderzeichen Zahlen und Buchstaben enthalten, sondern auch, dass man keine existierenden Wörter verwendet. Kombiniert man nun neue Zeichenfolgen mit einem breiten Schlüsselraum, kann Hashcat keine oder nur noch vereinzelte Passwörter finden. Das alleinige Verwenden eines bereiten Schlüsselraumes reicht dazu nicht aus, auch wenn dies die Sicherheit eines Passwortes erhöht.

# 4. Anhang

## 4.1 Das zur Einteilung und zum Hashen verwendete Programm

Dieses Programm habe ich zur Einteilung und zum Hashen der Passwörter verwendet.

Es wurde in C# geschrieben.

Aufgrund der Grösse, wurde der Quellcode auf eine Externe Plattform ausgelagert.

<https://github.com/Xelef2000/Passwortsortierer>

# 4.2 Literaturverzeichnis

1,4\_Mio\_most\_common\_passwords. (kein Datum). Abgerufen am 12. 12 2017 von von weakpass.com/wordlist

3,000,000+ combo list. (kein Datum). Abgerufen am 12. 12 2017 von weakpass.com/wordlist

Bowes, R. (kein Datum). 10\_million\_password\_list\_top\_1000000. Abgerufen am 12. 12 2017 von weakpass.com/wordlist

Bowes, R. (kein Datum). 10k\_most\_common. Abgerufen am 12. 12 2017 von weakpass.com/wordlist

Halfond, W., Viegas, J., & Orso, A. (2006). *A Classification of SQL Injection Attacks.* Georgia Institute of Technology College of Computing. Georgia, USA: Georgia Institute of Technology College of Computing.

Hashcat-Team. (7. 11 2017). Hashcat. *4.0.1*. Abgerufen am 10. 12 2017 von hashcat.net

Hornby, T. (kein Datum). Checksums human-only. (CrackStation, Hrsg.) Abgerufen am 12. 12 2017 von Checksums: crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm

Ladakis, E., Koromilas, L., Vasiliadis, G., Polychronakis, M., & Ioannidis, S. (2013). You Can Type, but You Can’t Hide: A Stealthy GPU-based Keylogger. *EuroSec’13.* Prague, Czech Republic: ACM Association for Computing Machinery.

Microsoft Corporation. (10 2016). MD5-Klasse. (msdn.microsoft.com, Hrsg.) Abgerufen am 13. 12 2017 von msdn.microsoft.com/de-de/library/system.security.cryptography.md5(v=vs.110).aspx

Monaghan, D., Gopinathan, U., Naughton, T., & Sheridan, J. (10. 9 2007). Key-space analysis of double random phase encryption. (O. T. Society, Hrsg.) *Applied Optics, 46*, S. 6641-6647.

Narayanan, A., & Shmatikov, V. (7. 11 2005). Fast Dictionary Attacks on Passwords Using Time-Space. In T. U. Austin (Hrsg.), *CCS '05 Proceedings of the 12th ACM conference on Computer and communications security* (S. 364-372). Alexandria: ACM Association for Computing Machinery. doi:10.1145/1102120.1102168

Plößl, K., Federrath, H., & Nowey, T. (2005). Schutzmöglichkeiten gegen Phishing. *Sicherheit 2005*(Beiträge der 2. Jahrestagung des GI-Fachbereichs Sicherheit, Lecture Notes in Informatics (P-62)), S. 161-164.

Pornin, T. (11. 01 2011). *Why passwords should be hashed*. (StackExchange, Herausgeber) Abgerufen am 10. 12 2017 von IT Security Community Blog: http://security.blogoverflow.com/2011/11/why-passwords-should-be-hashed/

Wook Shin, S., Lee, M.-K., Moon, D., & Moon, K. (1. 10 2009). Dictionary Attack on Functional Transform-Based Cancelable Fingerprint Templates. *ETRI Journal, 31*, S. 628-630. doi:10.4218/etrij.09.0209.0137

1. Siehe 4.1 [↑](#footnote-ref-1)