

Password-Cracking - Workshop

ZIP-Datei und Passwort-Hashe

Tom Gries



Dokumenten URL:

<http://docs.tx7.de/TT-PWC>

Autor:

Tom Gries <TT-PWC@tx7.de>
@tomo@chaos.social

Lizenz:

Creative Commons BY-NC-ND

Version:

7.3.0 vom 22.07.2025



Legal Disclaimer - Hackerparagraf

Die hier vorgestellten Methoden und Tools dienen zum Schutz der eigenen Systeme. Das Knacken fremder Passwörter ebenso wie das Eindringen in Systeme kann eine Straftat darstellen. Die vorgestellten Tools können unter den Hackerparagrafen 202c StGB fallen. Entsprechend dürfen Sie nur auf eigene Kennwörter oder Testsysteme losgehen, beziehungsweise sich schriftlich die Erlaubnis des Systembesitzers einholen.

Zudem können Cracking-Tools die getesteten Systeme stark beeinträchtigen oder außer Funktion setzen. Entsprechend vorsichtig sollten Sie bei Produktivsystemen sein.

Was Du für diesen Workshop brauchst

Für diesen Workshop ist ein Computer im Internet mit virtuellen Systemen (Debian, Kali Linux) vorbereitet. Daher brauchst Du nur einen Rechner mit Internetzugang und einen aktuellen Browser (Firefox, Chrome, Edge, Safari). Die komplette Bedienung erfolgt über den HTTPS Port 8006.

Es gibt keine besonderen Anforderungen an das Betriebssystem (Windows, Linux, Mac OS). Allerdings musst Du Dich etwas mit Linux, der Bash und Kali Tools auskennen. Mit googeln kommt man aber auch schon weit.

Dein Auftrag

Melde Dich an Deiner Instanz an, starte den Kali Linux Docker-Container (**./startDockerLiveHacking**) und schau Dich im HomeDirectory von root um. Cracke die ZIP-Datei und von den darin enthaltenen 5.000 Passworthashen so viele wie möglich. TIPP: Die damalige Passwortpolicy hatte nur Kleinbuchstaben und Ziffern erlaubt.

Du hast 45 Minuten Zeit. Versuche in dieser Zeit mehr Passwörter zu cracken als die anderen Teams.

Bevor wir uns EINE Lösung anschauen, lasst
uns über

WAHRSCHEINLICHKEITEN

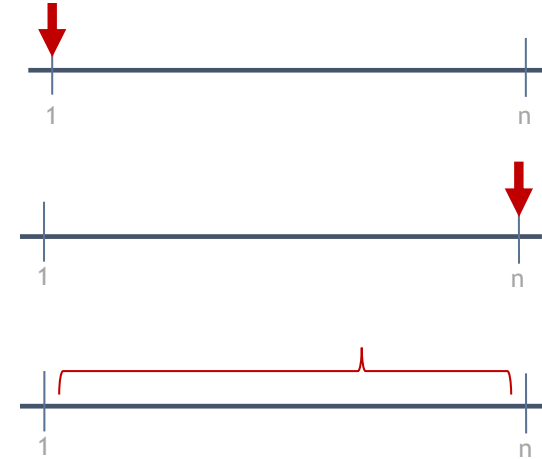
reden!



Wie lange dauert das cracken?

Passwortcracken ist endlich. Daher gibt es nur drei wesentliche Möglichkeiten, die man betrachten sollte beziehungsweise unterscheiden kann:

- Man findet das Passwort im ersten Versuch
- Man findet das Passwort im letzten Versuch
- Es liegt irgendwo dazwischen



Wenn man von einer Gleichverteilung ausgeht (nicht mit der Normalverteilung verwechseln), liegt die Wahrscheinlichkeit ein Passworthash zu cracken bei der halben Gesamtdauer (Gesamtcrackzeit \div 2).



Jetzt aber - EINE Möglichkeit, die Aufgabe zu lösen ...



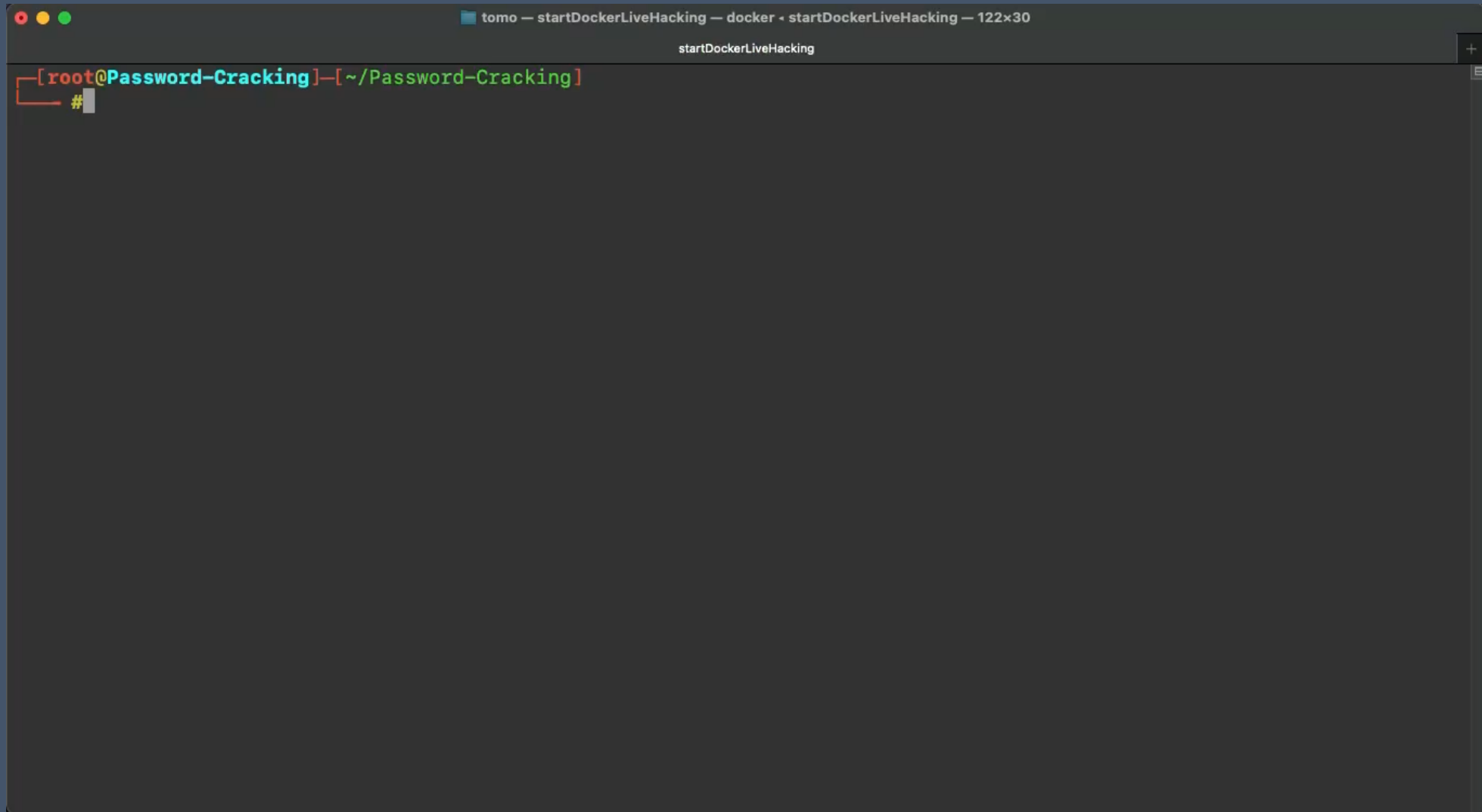
Wie lange dauert das cracken?

Die 5.000 Passworthashe sind aus dem Jahr 2002. Bei der Wörterbuchattacke wird RockYou von 2009 verwendet. Es beinhaltet über 14 Millionen Einträge, davon ca. 3 Millionen mit 8 Zeichen. Das Ausprobieren sämtlicher Kombinationen dauert auf dem bereitgestellten Server weniger als 2 Minuten.

Wie viele Passwörter werden insgesamt aufgedeckt? Mit Wörterbuchattacke und zusätzlich mit Brute Force?

- ☐ A: Bis zu 250 (5 %)
- ☐ B: Zwischen 250 (5 %) und 750 (15 %)
- ☐ C: Zwischen 750 (15 %) und 1.500 (30 %)
- ☐ D: Über 1.500 (30 %)

EINE Möglichkeit, die Aufgabe zu lösen ...



A terminal window titled "tomo — startDockerLiveHacking — docker - startDockerLiveHacking — 122x30". The window shows a shell prompt `[root@Password-Cracking]~[/Password-Cracking]` with a cursor. The terminal is dark-themed with light-colored text.

```
tomo — startDockerLiveHacking — docker - startDockerLiveHacking — 122x30
startDockerLiveHacking
[ root@Password-Cracking ] ~ [ /Password-Cracking ]
#
```



Zusammenfassung der wesentlichen Schritte

ZIP-Archiv cracken

```
zip2john CompanyPasswords.zip > zip-hash 2>/dev/null  
john zip-hash
```

Ermitteln des Hashverfahrens

```
nth --text `head -1 CompanyPasswords.txt | cut -d ":" -f2`
```

Wörterbuchangriff

```
john --wordlist=./rockyou.txt CompanyPasswords.txt
```

Zeichensatz optimieren (Häufigkeitsanalyse)

```
john --make-charset=~/.john/DES-5000.chr CompanyPasswords.txt
```

Brute Force Angriff

```
john --format=descrypt --incremental=DES-5000 CompanyPasswords.txt
```

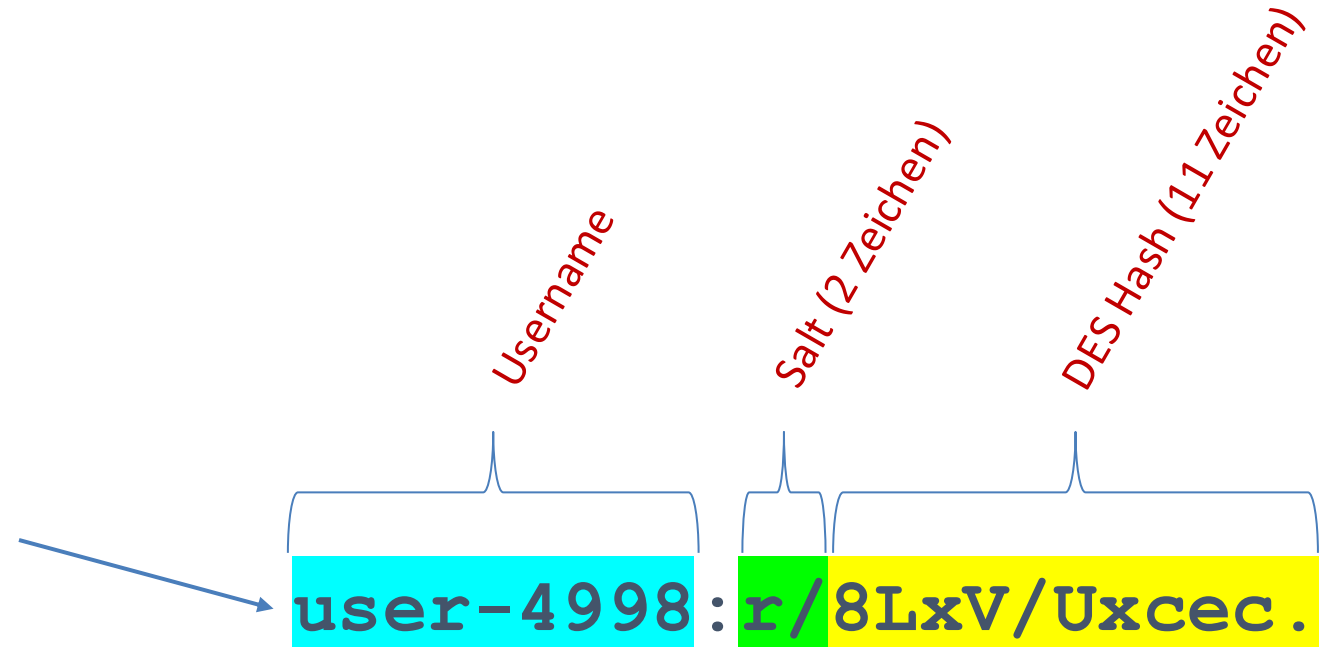


5.000 Passworthashe cracken

```
user-0001:p5pW88uib3DbA
user-0002:pcPHFxgkuhoMQ
user-0003:pzEgwiFCNiMtw
user-0004:q.0GnJiGgGQlA
user-0005:q/B4MmqJRVKMq
user-0006:q1Oek8e12bt62
.      .      .      .      .
.      .      .      .      .
.      (+ 4.990)      .
.      .      .      .      .
user-4997:t/KGjE/vdEu/w
user-4998:r/8LxV/Uxcec.
user-4999:sQcsckBbkRc8Y
user-5000:t74t2w2PKj09g
```

Ermitteltes Hashverfahren:

DES, Crypt, DES-Crypt, Standard DES





Übersicht einiger Hashverfahren

Hash Type	Published	Prefix	# Salt	# Hash
DES Crypt	1975	No Prefix	2	11
LM-Hash (LAN Manager)	1988/1989	No Prefix	No salt	32 (2x16)
NTLM Hash	1993	No Prefix	No salt	32
MD5	1991	\$1\$	up to 8, default = 8	22
SHA1	1995	\$sha1\$	up to 64, default = 8	28
Blowfish (bcrypt)	1999	\$2\$ \$2a\$ \$sb\$ \$2x\$ \$2y\$	22	31
SHA256 (part of SHA-2 family)	2001	\$5\$	up to 16, default = 8	43
SHA512 (part of SHA-2 family)	2001	\$6\$	up to 16, default = 16	86



Noch etwas Geschichte zum Abschluss ...

Die ersten Passwörter wurden im Klartext gespeichert. Die ersten Passworthashe wurden in den späten 1970er und frühen 1980er Jahren eingeführt.

Eine der ersten Passworthashfunktionen war zum Beispiel die sogenannte "crypt"-Funktion, die im UNIX-Betriebssystem um 1979 eingeführt wurde. Diese Funktion verwendete den Algorithmus "DES" (Data Encryption Standard), um Passwörter zu hashen. DES wurde Anfang der 1970er von IBM mit "Unterstützung" der NSA entwickelt. 1976 war der erste Einsatz.

Eins der bekanntesten Passwort-Crackprogramme ist John the Ripper (1996). Es lief auf vielen Unix Rechnern im Hintergrund und versuchte, die lokalen Passwörter zu cracken. Bei Erfolg erhielt der User eine Email.



Infos zu DES (decrypt, traditional)

- ⇒ DES war schon damals gegen differenzielle Kryptoanalyse resistent, obwohl diese Methode erst 1991 veröffentlicht wurde. Die DES-Entwickler beziehungsweise die NSA kannten die differentielle Kryptoanalyse schon 15 Jahre vor der offiziellen Veröffentlichung.
- ⇒ Der gesamte String besteht aus 13 Zeichen. Die ersten beiden Zeichen sind das Salt. Die verbleibenden 11 der eigentliche Hash. DES Hash ist theoretisch kollisionsresistent: $95^8 = 6.634.204.312.890.625 \approx 6,6 \times 10^{15}$ vs. $64^{11} = 73.786.976.294.838.206.464 \approx 73.787 \times 10^{15} \approx \text{Faktor } 11.000$
- ⇒ DES kann nur einen Hash über maximal 8 Zeichen erzeugen und hat eine Schlüssellänge von 56 Bit (Kompromiss von NSA und IBM: 48/64 Bit).

```
>>> des_crypt.hash("12345678", salt="x.")  
'x.KxRJcXiV/jk'
```

```
>>> des_crypt.hash("123456789", salt="x.")  
'x.KxRJcXiV/jk'
```

Anmerkungen oder Fragen?