Live Hacking

Das Mysterium Passwörter und wie man sie (clever) crackt

Tom Gries (TOMO) | GPN21 | Juni 2023 | Raum 208 (HFG) | 90 Minuten

@tomo@chaos.social

@_TomGries_

2023-06-08

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 einem aktuellen Browser (Firefox, Chrome, Edge, Safari). Die komplette Bedienung ist über die Ports 80 bzw. 443 möglich.

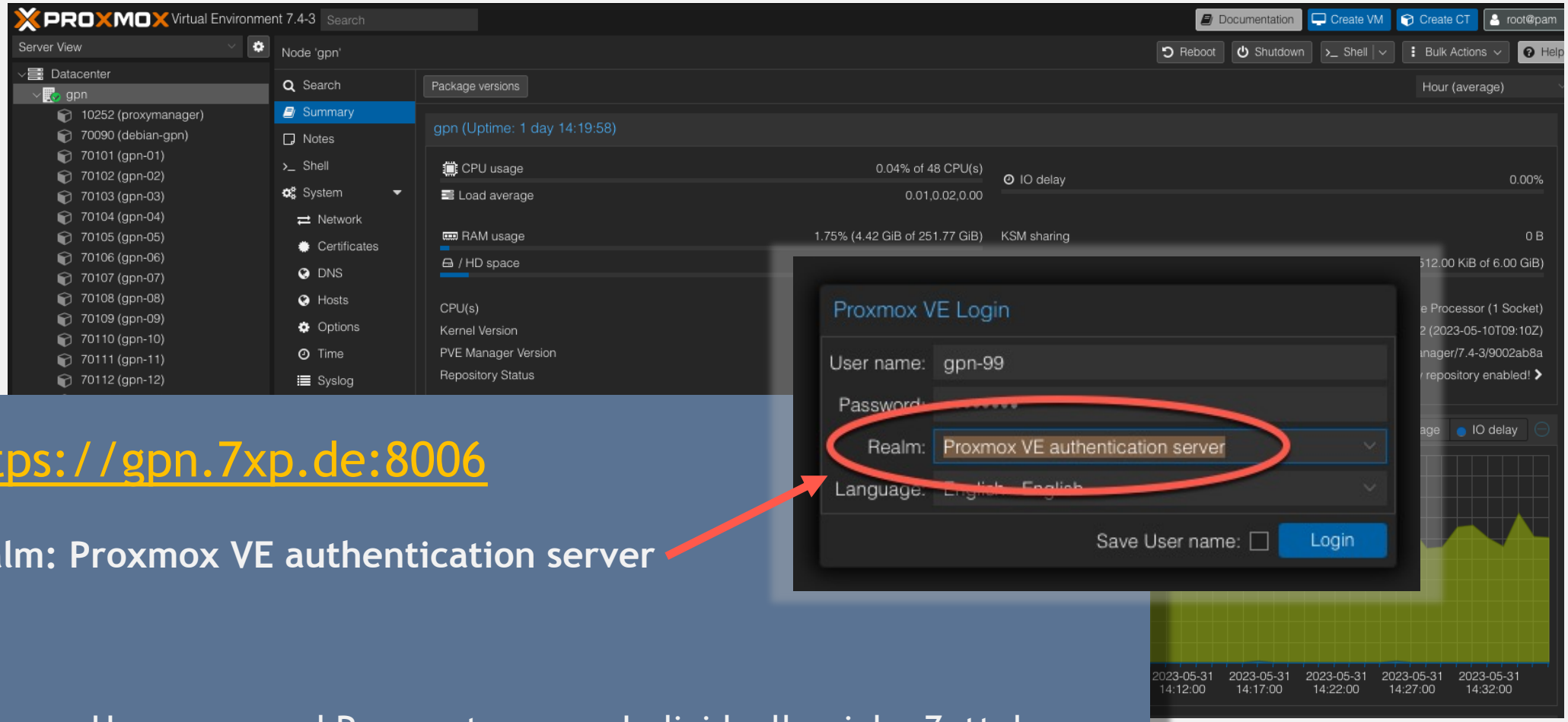
Es gibt keine besonderen Anforderungen an das Betriebssystem (Windows, Linux, Mac OS). Allerdings musst Du Dich etwas mit Linux, der Bash und Kali Tools auskennen.

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.

Die Virtualisierungsumgebung (Proxmox)

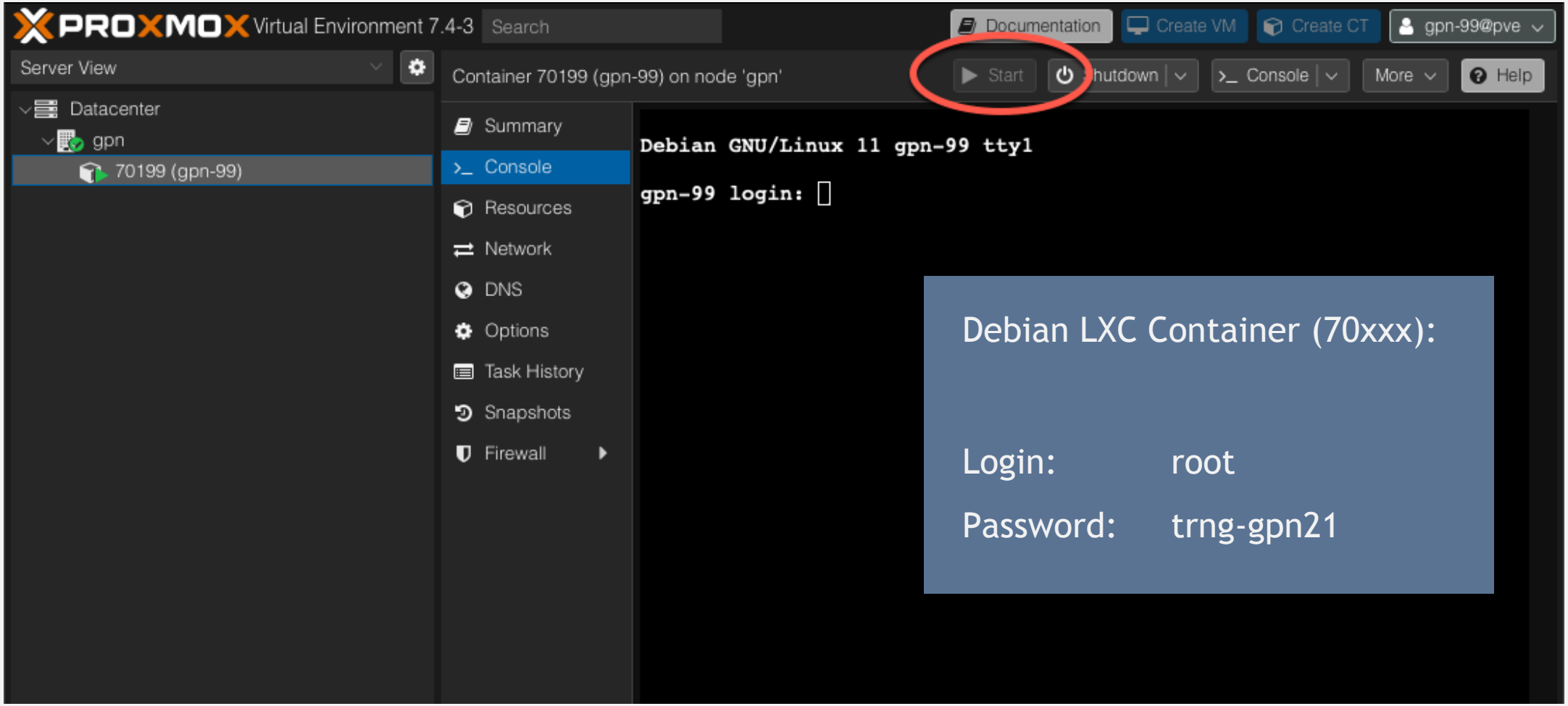


https://gpn.7xp.de:8006

Realm: Proxmox VE authentication server

Proxmox Username und Passwort: Individuell – siehe Zettel

Deine individuelle Proxmoxumgebung



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 Passwort-hashen so viele wie möglich. TIPP: Die Passwortpolicy hat nur Kleinbuchstaben und Ziffern erlaubt.

Du hast 45 Minuten Zeit. Gewonnen hat, wer in dieser Zeit die meisten Passwörter gecrackt hat.

Bevor wir uns EINE Lösung anschauen, lasst uns über

WAHRSCHEINLICHKEITEN

reden!

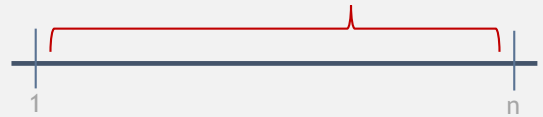
Wie lange dauert das cracken?

Passwortcracken ist endlich. Daher gibt es nur drei wesentliche Möglichkeiten, die man betrachten sollte 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 ÷ 2).

Jetzt aber - EINE Möglichkeit, die Aufgabe zu lösen …

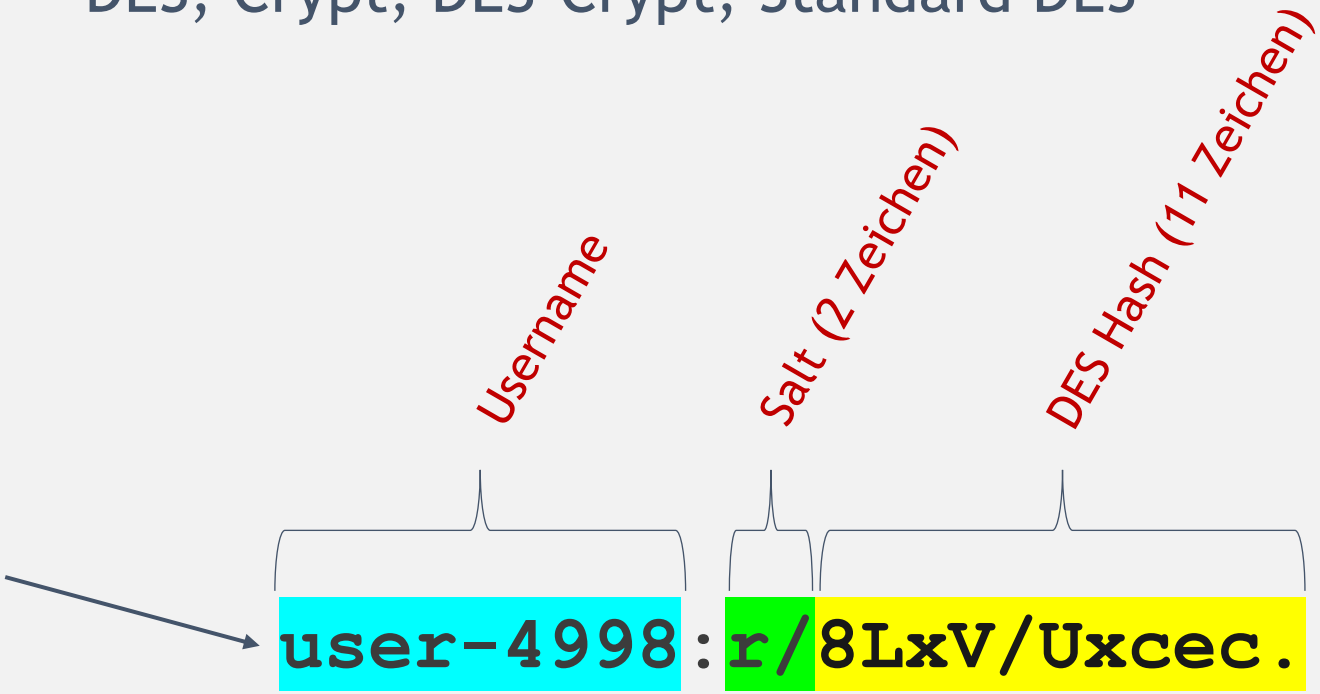
… Details im Docker Container

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



DES illustrated: https://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

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

Hash Identifier nth (Python): https://github.com/HashPals/Name-That-Hash

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-Betriebssystems 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 (descrypt, 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.

⇒ 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'
```

⇒ 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 ≈ 6,6 x 10^15 vs.
64^11 = 73.786.976.294.838.206.464 ≈ 73.787 x 10^15 ≈ Faktor 11.000

Benchmarks

John Benchmark - Yoga C940 (docker)

```
┌──(root㉿kali-docker)-[~]
└─# date
Fri Jan 13 00:02:13 UTC 2023

┌──(root㉿kali-docker)-[~]
└─# john --format=descrypt --test
Will run 8 OpenMP threads
Benchmarking: descrypt, traditional crypt(3) [DES 512/512 AVX512F]... (8xOMP) DONE
Many salts:     40798K c/s real, 5577K c/s virtual
Only one salt:  22370K c/s real, 3035K c/s virtual


┌──(root㉿kali-docker)-[~]
└─# john --format=LM --test
Will run 8 OpenMP threads
Benchmarking: LM [DES 512/512 AVX512F]... (8xOMP) DONE
Raw:    52490K c/s real, 6975K c/s virtual


┌──(root㉿kali-docker)-[~]
└─# john --format=NT --test
Benchmarking: NT [MD4 512/512 AVX512BW 16x3]... DONE
Raw:    58049K c/s real, 58341K c/s virtual
```

John Benchmark - Hetzner AX41 (docker)

```
┌──(root㉿kali-docker)-[~]
└─# date
Fri Jan 13 00:18:59 UTC 2023

┌──(root㉿kali-docker)-[~]
└─# john --format=descrypt --test
Will run 12 OpenMP threads
Benchmarking: descrypt, traditional crypt(3) [DES 256/256 AVX2]... (12xOMP) DONE
Many salts:     102445K c/s real, 8537K c/s virtual
Only one salt:  47812K c/s real, 3991K c/s virtual


┌──(root㉿kali-docker)-[~]
└─# john --format=LM --test
Will run 12 OpenMP threads
Benchmarking: LM [DES 256/256 AVX2]... (12xOMP) DONE
Raw:    85979K c/s real, 7197K c/s virtual


┌──(root㉿kali-docker)-[~]
└─# john --format=NT --test
Benchmarking: NT [MD4 256/256 AVX2 8x3]... DONE
Raw:    73254K c/s real, 73254K c/s virtual
```

John Benchmark - Hetzner AX101 (docker)

```
┌──(root㉿Password-Cracking)-[~]
└─# date
Sat Mar 11 16:47:54 UTC 2023

┌──(root㉿Password-Cracking)-[~]
└─# john --format=descrypt --test
Will run 32 OpenMP threads
Benchmarking: descrypt, traditional crypt(3) [DES 256/256 AVX2]... (32xOMP) DONE
Many salts:     253034K c/s real, 7909K c/s virtual
Only one salt:  61046K c/s real, 1908K c/s virtual


┌──(root㉿Password-Cracking)-[~]
└─# john --format=LM --test
Will run 32 OpenMP threads
Benchmarking: LM [DES 256/256 AVX2]... (32xOMP) DONE
Raw:    93880K c/s real, 2936K c/s virtual


┌──(root㉿Password-Cracking)-[~]
└─# john --format=NT --test
Benchmarking: NT [MD4 256/256 AVX2 8x3]... DONE
Raw:    78209K c/s real, 78209K c/s virtual
```

Hashcat Benchmark - GeForce GTX 1660 Ti

```
$ hashcat -b -m 1500 -w 3

   Hashtype: descrypt, DES (Unix)
   Speed.Dev.#3.....:      786.8 MH/s

$ hashcat -b -m 3000 -w 3

   Hashtype: LM
   Speed.Dev.#3.....:    19241.4 MH/s

$ hashcat -b -m 1000 -w 3

   Hashtype: NTLM
   Speed.Dev.#3.....:    35765.0 MH/s
```

Mit hashcat 6.x im September 2019

Hashcat Benchmark - GeForce GTX 1660 Ti

```
$ hashcat -b -m 1600 -w 3

   Hashtype: Apache $apr1$ MD5, md5apr1, MD5 (APR)
   Speed.Dev.#3.....:     9808.9 kH/s

$ hashcat -b -m 8900 -w 3

   Hashtype: scrypt
   Speed.Dev.#3.....:     857.6 kH/s

$ hashcat -b -m 3200 -w 3

   Hashtype: bcrypt $2*$, Blowfish (Unix)
   Speed.Dev.#3.....:     10260 H/s
```

Mit hashcat 6.x im September 2019

Hashcat Benchmark - über 100 GH/s

hashcat
@hashcat

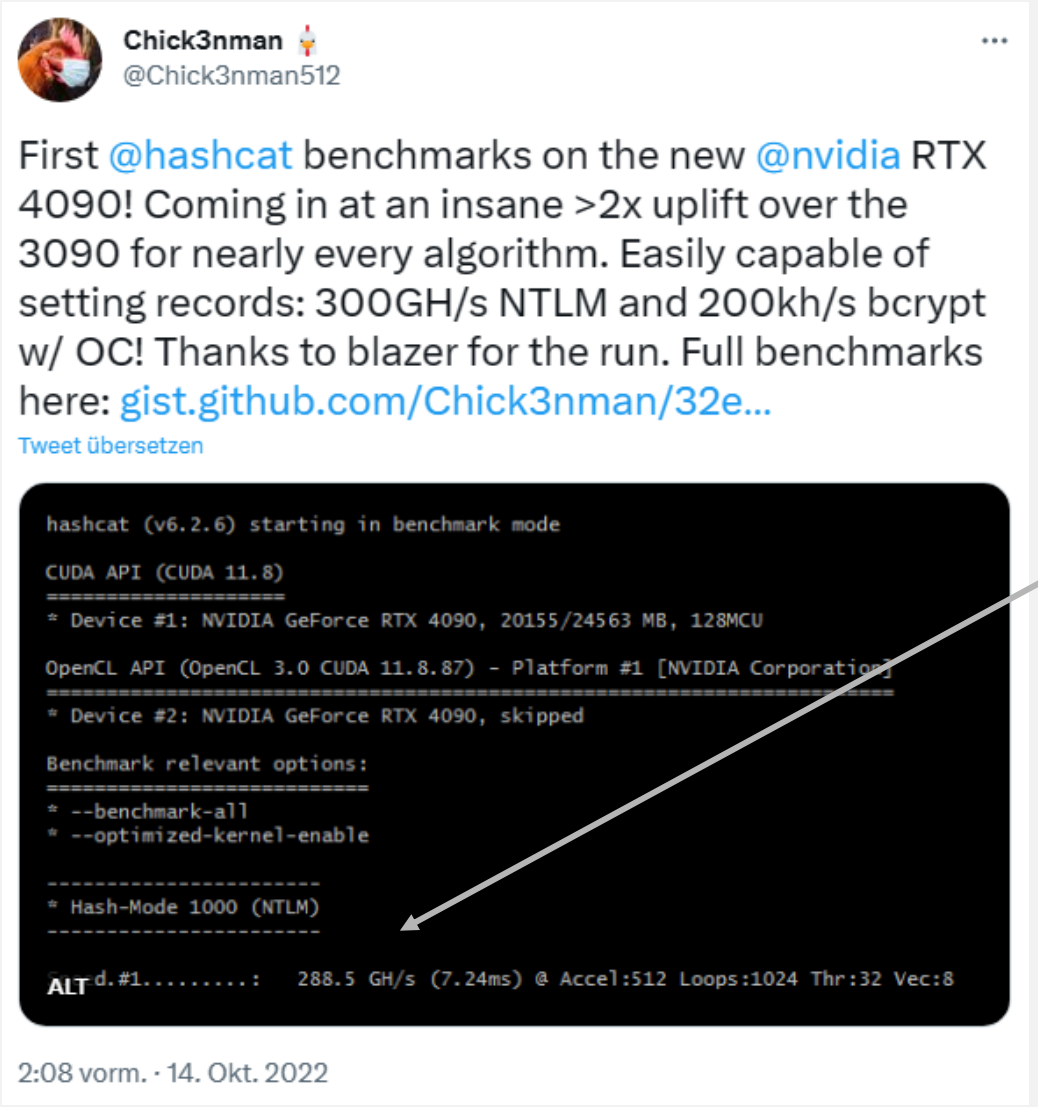
hand-tuned hashcat 6.0.0 beta and 2080Ti (stock clocks) breaks NTLM cracking speed mark of 100GH/s on a single compute device

Tweet vom Februar 2019

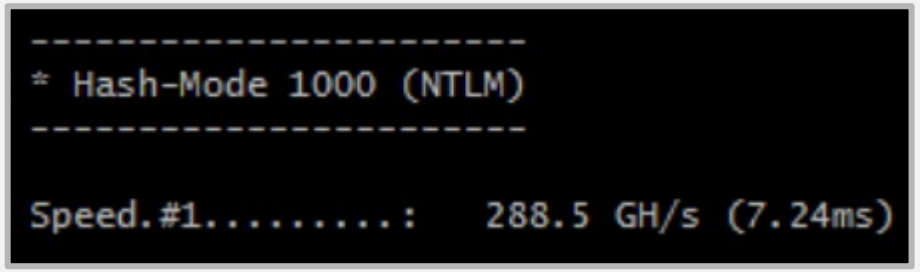
C:\Windows\System32\cmd.exe

```
d:\tools\hashcat-6.0.0>hashcat64 -b -m 1000 -u 1024 -n 512 --opencl-vector-width 8 --force -O

OpenCL Platform #1: NVIDIA Corporation
======================================
* Device #1: GeForce RTX 2080 Ti, 2816/11264 MB allocatable, 68MCU

Benchmark relevant options:
===========================
* --force
* --optimized-kernel-enable
* --opencl-vector-width=8
* --kernel-accel=512

Hashmode: 1000 - NTLM

Speed.#1.........:   102.8 GH/s (10.48ms) @ Accel:512 Loops:1024 Thr:32 Vec:8

Started: Wed Feb 13 22:57:19 2019
Stopped: Wed Feb 13 22:57:26 2019

d:\tools\hashcat-6.0.0>_
```

Hashcat Benchmark - Stand heute (2022/23)



288,5 Milliarden Hashe/Sek.
Tweet vom 14.10.2022



Mehr Benchmarks:
https://t.co/Bftucib7P9

Das war …

Live Hacking

Das Mysterium Passwörter und wie man sie (clever) crackt

Tom Gries (TOMO) | GPN21 | Juni 2023 | Raum 208 (HFG) | 90 Minuten

@tomo@chaos.social

@_TomGries_