

Esercizio S6L3: Simone Esposito

Traccia: password cracking.

Sentitevi liberi di utilizzare qualsiasi tool o soluzione alternativa.

L'obiettivo dell'esercizio di oggi è craccare tutte le password.

Le password da craccare sono le seguenti:

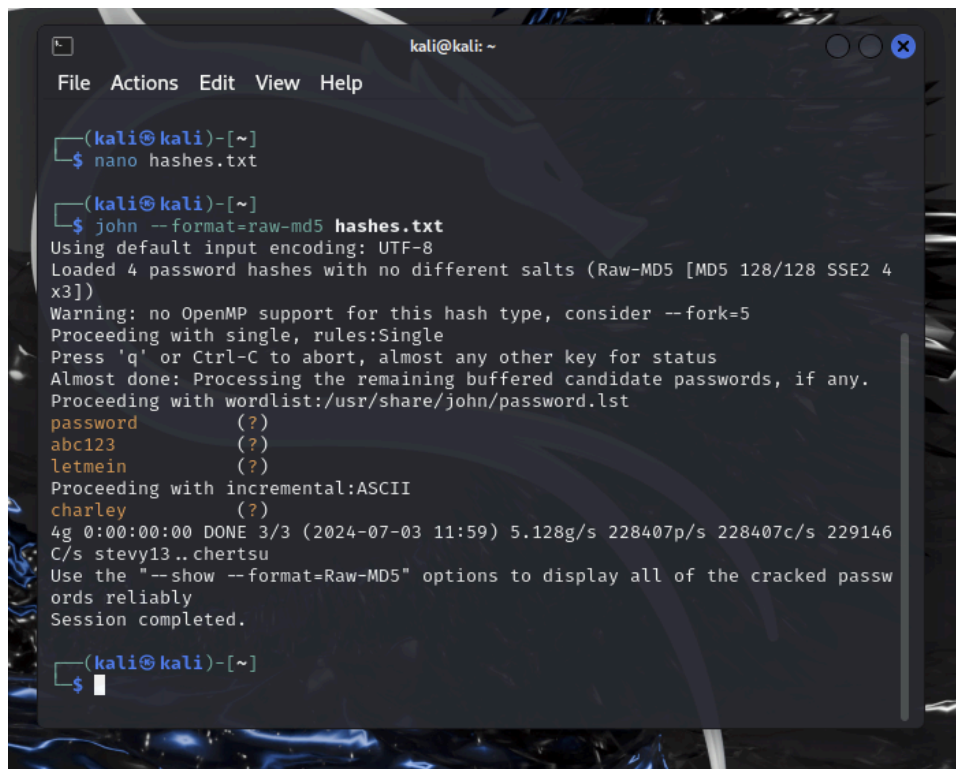
- 5f4dcc3b5aa765d61d8327deb882cf99
- e99a18c428cb38d5f260853678922e03
- 8d3533d75ae2c3966d7e0d4fcc69216b
- 0d107d09f5bbe40cade3de5c71e9e9b7
- 5f4dcc3b5aa765d61d8327deb882cf99

Iniziamo con **John The Ripper**

Crea un file con gli hash: Salva gli hash in un file chiamato hashes.txt, con le password da craccare sono le seguenti:

- 5f4dcc3b5aa765d61d8327deb882cf99
- e99a18c428cb38d5f260853678922e03
- 8d3533d75ae2c3966d7e0d4fcc69216b
- 0d107d09f5bbe40cade3de5c71e9e9b7
- 5f4dcc3b5aa765d61d8327deb882cf99

Eseguiamo John the Ripper: john --format=raw-md5 hashes.txt

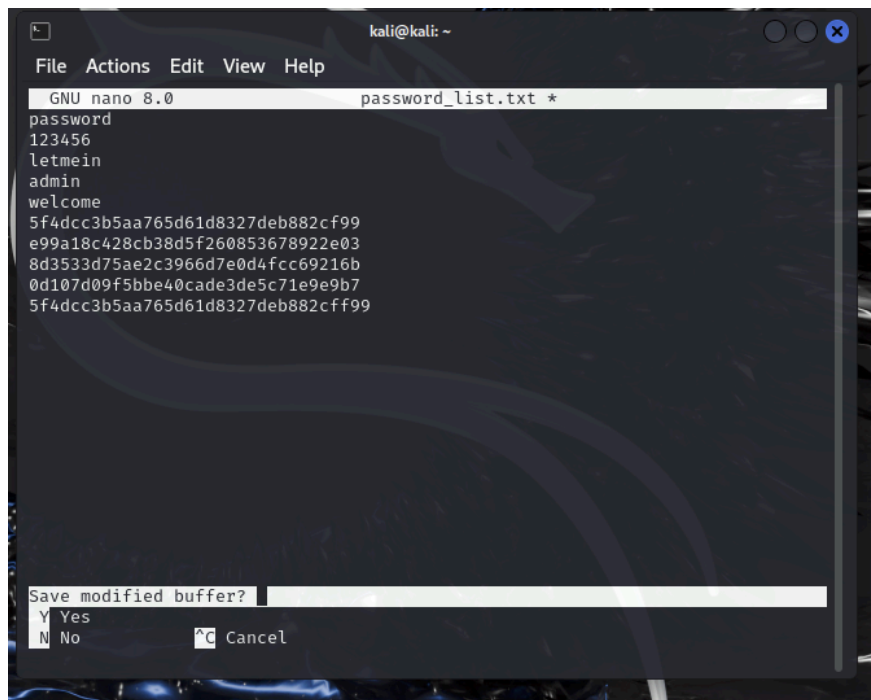


```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ nano hashes.txt  
  
(kali@kali)-[~]  
$ john --format=raw-md5 hashes.txt  
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4  
x3])  
Warning: no OpenMP support for this hash type, consider --fork=5  
Proceeding with single, rules:Single  
Press 'q' or Ctrl-C to abort, almost any other key for status  
Almost done: Processing the remaining buffered candidate passwords, if any.  
Proceeding with wordlist:/usr/share/john/password.lst  
password (?)  
abc123 (?)  
letmein (?)  
Proceeding with incremental:ASCII  
charley (?)  
4g 0:00:00:00 DONE 3/3 (2024-07-03 11:59) 5.128g/s 228407p/s 228407c/s 229146  
C/s stevy13..chertsu  
Use the "--show --format=Raw-MD5" options to display all of the cracked pass  
ords reliably  
Session completed.  
  
(kali@kali)-[~]  
$
```

HASHCAT

Per utilizzare hashcat per craccare gli hash MD5, ho seguito una serie di passaggi:

1. **Preparazione dell'Ambiente:** Ho verificato che hashcat sia installato su Kali Linux e ho verificato la presenza del file wordlist rockyou.txt. Se il file era compresso (rockyou.txt.gz), l'ho decompresso per ottenere il file rockyou.txt.
2. **Creazione del File degli Hash:** Ho creato un file di testo (ad esempio, passwords.txt) contenente gli hash MD5 che voglio craccare. Ogni hash è stato inserito su una nuova riga senza spazi o caratteri aggiuntivi.
3. **Verifica del Contenuto del File:** Ho controllato il contenuto del file degli hash per assicurarmi che fosse corretto e che non ci fossero caratteri non stampabili o spazi extra.
4. **Esecuzione di hashcat:** Ho eseguito hashcat specificando il tipo di hash (-m 0 per MD5), la modalità di attacco (-a 0 per il dizionario), il file degli hash e il file wordlist (ad esempio, rockyou.txt). Ho utilizzato percorsi completi per i file se necessario per evitare errori di percorso.
5. **Interpretazione dei Risultati:** Ho analizzato l'output di hashcat per verificare quali hash sono stati craccati con successo. Gli hash craccati e le relative password sono stati mostrati nella console o salvati in un file specifico da hashcat.



```
(kali@kali)-[~]
$ hashcat -m 0 -a 0 /path/to/passwords.txt /usr/share/wordlists/rockyou.txt

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian Linux, None+Asserts, RELOC, SPIR, LLV
M 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

=====
* Device #1: cpu-penryn-Intel(R) Core(TM) Ultra 7 155H, 1562/3189 MB (512 MB
allocatable), 5MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
```

RAINBOW TABLES

Generiamo le rainbow tables

```
(kali㉿kali)-[~]  
$ sudo rtgen md5 loweralpha-numeric 1 7 0 2400 8000000 0  
rainbow table md5_loweralpha-numeric#1-7_0_2400x8000000_0.rt parameters  
hash algorithm:      md5  
hash length:        16  
charset name:        loweralpha-numeric  
charset data:        abcdefghijklmnopqrstuvwxyz0123456789  
charset data in hex:  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72  
73 74 75 76 77 78 79 7a 30 31 32 33 34 35 36 37 38 39  
charset length:      36  
plaintext length range: 1 - 7  
reduce offset:       0x00000000  
plaintext total:     80603140212  
  
sequential starting point begin from 0 (0x0000000000000000)  
generating...  
163840 of 8000000 rainbow chains generated (0 m 44.9 s)  
327680 of 8000000 rainbow chains generated (1 m 6.1 s)  
491520 of 8000000 rainbow chains generated (0 m 48.0 s)  
655360 of 8000000 rainbow chains generated (0 m 47.9 s)  
819200 of 8000000 rainbow chains generated (0 m 50.5 s)  
█
```