

Práctica Kali Linux Criptografía

Enrique Ulises Báez Gómez Tagle

21 de octubre de 2024

Índice

1. Crear Carpetas	3
2. Investigación de los comandos	3
3. Creación de archivos	4
4. Cifrar archivos	4
5. Obtener Contraseñas	5
6. Resultados y Desafíos Encontrados	6
7. Conclusión	7

1. Crear Carpetas

```
1 mkdir openssl ccrypt
```

```
(eubgo@KRAKEN)~$ neofetch
neofetch
OS: Kali GNU/Linux Rolling on Windows 10 x86_64
Kernel: 5.15.153.1-microsoft-standard-WSL2
Uptime: 28 secs
Packages: 481 (dpkg)
Shell: bash 5.2.21
Terminal: Relay(11)
CPU: Intel i7-10750H (12) @ 2.592GHz
GPU: 2166.00.00.0 Microsoft Corporation Basic Render Driver
Memory: 322M1B / 7827M1B

(eubgo@KRAKEN)~$ ls
(eubgo@KRAKEN)~$ mkdir openssl ccrypt
(eubgo@KRAKEN)~$ ls
ccrypt  openssl
(eubgo@KRAKEN)~$
```

2. Investigación de los comandos

```
1 ccrypt --h
2 openssl -h
```

```
(eubgo@KRAKEN)~$ ccrypt --h
ccrypt 1.11. Secure encryption and decryption of files and streams.

Usage: ccrypt [mode] [options] [file...]
ccencrypt [options] [file...]
ccdecrypt [options] [file...]
ccat [options] file...

Modes:
-e, --encrypt      encrypt
-d, --decrypt      decrypt
-c, --cat          cat; decrypt files to stdout
-x, --keychange    change key
-u, --unixcrypt    decrypt old unix crypt files

Options:
-h, --help          print this help message and exit
-V, --version       print version info and exit
-L, --license       print license info and exit
-v, --verbose       print progress information to stderr
-q, --quiet         run quietly; suppress warnings
-f, --force         overwrite existing files without asking
-m, --mismatch      allow decryption with non-matching key
-E, --envvar var    read keyword from environment variable (unsafe)
-K, --key key       give keyword on command line (unsafe)
-k, --keyfile file  read keyword(s) as first line(s) from file
-P, --prompt prompt use this prompt instead of default
-S, --suffix .suf   use suffix .suf instead of default .cpt
-s, --strictsuffix  refuse to encrypt files which already have suffix
-F, --envvar2 var   as -E for second keyword (for keychange mode)
-H, --key2 key      as -K for second keyword (for keychange mode)
-Q, --prompt2 prompt as -P for second keyword (for keychange mode)
-t, --timid         prompt twice for encryption keys (default)
-b, --brave         prompt only once for encryption keys
-y, --keyref file   encryption key must match this encrypted file
-r, --recursive     recurse through directories
```

```
(eubgo KRAKEN)-[~]
$ openssl -h
help:

Standard commands
asn1parse      ca          ciphers      cmp
cms            crl         crl2pkcs7    dgst
dhparam        dsa         dsaparam     ec
ecparam        enc         engine       errstr
fipsinstall    gendsa     genpkey       genrsa
help           info        kdf          list
mac            nseq       ocsdp        passwd
pkcs12         pkcs7      pkcs8        pkey
pkeyparam      pkeyutl    prime        rand
rehash         req        rsa          rsautl
s_client       s_server   s_time       sess_id
smime          speed      splac        srp
storeutl       ts         verify       version
x509

Message Digest commands (see the 'dgst' command for more details)
blake2b512     blake2s256 md4           md5
rmd160         sha1        sha224       sha256
sha3-224       sha3-256   sha3-384     sha3-512
sha384         sha512     sha512-224   sha512-256
shake128       shake256    sm3

Cipher commands (see the 'enc' command for more details)
aes-128-cbc     aes-128-ecb aes-192-cbc   aes-192-ecb
aes-256-cbc     aes-256-ecb aria-128-cbc   aria-128-cfb
aria-128-cfb1   aria-128-cfb8 aria-128-ctr   aria-128-ecb
aria-128-ofb    aria-192-cbc  aria-192-cfb  aria-192-cfb1
aria-192-cfb8   aria-192-ctr  aria-192-ecb  aria-192-ofb
aria-256-cbc    aria-256-cfb aria-256-cfb1 aria-256-cfb8
aria-256-ctr    aria-256-ecb aria-256-ofb   base64
bf             bf-cbc       bf-cfb        bf-ecb
bf-ofb         camellia-128-cbc camellia-128-ecb camellia-192-cbc
camellia-192-ecb camellia-256-cbc camellia-256-ecb cast
cast-cbc       cast5-cbc    cast5-cfb     cast5-ecb
```

3. Creación de archivos

```
1 nano a1.txt
2 nano a2.txt
3 nano a3.txt
```

```
(eubgo KRAKEN)-[~]
$ nano a1.txt

(eubgo KRAKEN)-[~]
$ nano a2.txt

(eubgo KRAKEN)-[~]
$ nano a3.txt

(eubgo KRAKEN)-[~]
$ |
```

4. Cifrar archivos

```

1 openssl enc -des -in a1.txt -out enc_a1.enc -k 0000
2 openssl enc -des -in a2.txt -out enc_a2.enc -k crypto24up
3 openssl enc -aes-256-cbc -in a3.txt -out enc_a3.enc -k
  P4ssW0rD$

```

```

(eubgo@KRAKEN)~$ openssl enc -des -in a1.txt -out enc_a1.enc -k 0000
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

(eubgo@KRAKEN)~$ openssl enc -des -in a2.txt -out enc_a2.enc -k crypto24up
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

(eubgo@KRAKEN)~$ openssl enc -aes-256-cbc -in a3.txt -out enc_a3.enc -k P4ssW0rD"$
> ^C

(eubgo@KRAKEN)~$ openssl enc -aes-256-cbc -in a3.txt -out enc_a3.enc -k P4ssW0rD$
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

```

5. Obtener Contraseñas

```

1 zip --password 0000 archivo.zip a1.txt
2 zip2john archivo.zip > archivo_hash.txt
3 john archivo_hash.txt
4 john --show archivo_hash.txt

```

1. Se generó un ZIP con el archivo txt
2. Se obtuvo el hash del zip
3. Se atacó con John
4. Se visualizan las contraseñas

```

(eubgo@KRAKEN)~$ zip --password 0000 archivo.zip a1.txt
adding: a1.txt (stored 0%)

(eubgo@KRAKEN)~$ zip2john archivo.zip > archivo_hash.txt
Created directory: /home/eubgo/.john
ver 1.0 efb 8455 efb:7076 archivo.zip/a1.txt PKZIP Encr: 2b chh, Y5_chh, cplenc=24, decompLen=12, crc=856792B4 ts=0BCF cas=0bcb type=0

(eubgo@KRAKEN)~$ john archivo_hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [12/6a])
Will run 12 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.
Proceeding with wordlist:/usr/share/john/password.txt
0000 (archivo.zip/a1.txt)
1g 0:00:00.00 DONE 2/3 (020a-1b21 R1:45) 26.88g/s 12180p/s 12180c/s 123456..NATT
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(eubgo@KRAKEN)~$ john --show archivo_hash.txt
archivo.zip/a1.txt:0000:a1.txt:archivo.zip:archivo.zip
1 password hash cracked, 0 left

```

Para el segundo y tercero, íbamos a usar crunch, pero estaba tardando mucho, por lo que optamos por demostrar el funcionamiento de una wordlist, en la que escondimos la contraseña, y le pedimos a john que la encontrara

```

1 nano dictionary.txt
2 john --wordlist=dictionary.txt archivo2_hash.txt
3 john --wordlist=dictionary.txt archivo2_hash.txt
4 john --show archivo2_hash.txt

```

```

(eubgo@KRAKEN)~$ nano dictionary.txt
(eubgo@KRAKEN)~$ john --wordlist=dictionary.txt archivo2_hash.txt
stat: archivo2_hash.txt: No such file or directory
(eubgo@KRAKEN)~$ john --wordlist=dictionary.txt archivo2_hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 12 OpenMP threads
Crash recovery file is locked: /home/eubgo/.john/john.rec
(eubgo@KRAKEN)~$ john --wordlist=dictionary.txt archivo2_hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 5 candidates left, minimum 12 needed for performance.
crypto24up (archivo2_des.zip/a2.txt)
ig 0:00:00:00 DONE (2024-10-21 02:00) 50.00g/s 250.0p/s 250.0c/s 250.0C/s P4ssw0rd...crypto24up
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
(eubgo@KRAKEN)~$ john --show archivo2_hash.txt
archivo2_des.zip/a2.txt:crypto24up:a2.txt:archivo2_des.zip:archivo2_des.zip
1 password hash cracked, 0 left
(eubgo@KRAKEN)~$

```

```

(eubgo@KRAKEN)~$ john --wordlist=mi_diccionario_personalizado.txt archivo_a3_hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates left, minimum 12 needed for performance.
P4ssw0rD$ (archivo_a3.zip/a3.txt)
ig 0:00:00:00 DONE (2024-10-21 02:12) 50.00g/s 150.0p/s 150.0c/s 150.0C/s P4ssw0rD$.P4ssw0rD$
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

6. Resultados y Desafíos Encontrados

Durante esta práctica, se realizaron diversas actividades de cifrado y descifrado utilizando herramientas como **OpenSSL**, **John the Ripper** y el uso de diccionarios de contraseñas. A continuación, se detallan los resultados obtenidos y los principales desafíos encontrados.

Para el **archivo a1.txt**, cifrado con el algoritmo **DES** y la contraseña de baja complejidad **0000**, se utilizó **John the Ripper** para descubrir la contraseña mediante un ataque directo. Debido a la sencillez de la contraseña, el proceso fue exitoso en pocos segundos. Este experimento destaca la importancia de evitar el uso de contraseñas triviales, ya que pueden ser descifradas fácilmente mediante ataques de fuerza bruta.

En el caso del **archivo a2.txt**, cifrado también con **DES**, pero con la contraseña de complejidad media **crypto24up**, se intentó realizar un ataque utilizando **Crunch** para generar combinaciones. Sin embargo, debido al alto número de combinaciones posibles, el proceso resultó ineficaz en términos

de tiempo y recursos. Como alternativa, se optó por utilizar una lista de contraseñas (**wordlist**) personalizada que contenía la contraseña escondida. Al utilizar esta wordlist con **John the Ripper**, la contraseña fue descubierta exitosamente.

Finalmente, para el **archivo a3.txt**, cifrado con **AES-256-CBC** y la contraseña **""P4ssW0rD\$""**, se realizó un ataque similar, utilizando también una wordlist. Aunque la contraseña era más compleja y requería más tiempo de procesamiento, **John the Ripper** pudo encontrar la contraseña con éxito, demostrando la eficiencia de los ataques basados en diccionarios en estos casos.

Un desafío importante fue el manejo de contraseñas más complejas y el ajuste del tamaño de las wordlists. En particular, para el cifrado con **AES-256-CBC**, fue crucial optimizar las listas de contraseñas y evitar el uso de ataques de fuerza bruta completos, que eran imprácticos debido al alto número de combinaciones.

7. Conclusión

A lo largo de esta práctica, se realizaron ejercicios de cifrado y descifrado con herramientas populares como **OpenSSL** y **John the Ripper**. Aunque se enfrentaron dificultades al realizar ataques de fuerza bruta en contraseñas más complejas, el uso de diccionarios personalizados permitió superar estos obstáculos y completar el proceso de descifrado exitosamente. Esta práctica subraya la importancia de utilizar contraseñas fuertes y complejas para proteger los datos y la necesidad de optimizar los ataques de descifrado para contraseñas de diferentes niveles de complejidad.