Práctica 1: Estudio de la fortaleza de contraseñas

Ingeniería de la ciberseguridad

Octubre 2023

Grado en Ingeniería Informática

Pablo Hidalgo Delgado. NIA: 100451225
Marcos Caballero Cortés. NIA: 100451047

ÍNDICE

1. Introducción

El objetivo de esta práctica es profundizar en el estudio de la fortaleza de contraseñas y el uso de herramientas de rotura de las mismas. Para ello, generamos 25 datasets de contraseñas diferentes y, utilizando la herramienta John the Ripper, idearemos y analizaremos distintas estrategias para romperlas. Trataremos de romper el máximo número de contraseñas con el menor número de estrategias posible.

2. Generación de datasets

2.1 Datasets aleatorios con letras minúsculas.

Para los 5 datasets de letras minúsculas, hemos creado un script de python que utiliza la librería string. De esta manera y eligiendo una longitud de 3 a 7 para cada dataset, generamos los caracteres con “string.ascii_lowercase” y los juntamos dependiendo de la longitud especificada.
El script para una contraseña de longitud 3 es el siguiente:

```
Unset
import random
import string

def generar_caracter():
    longitud = 3
    caracter = string.ascii_lowercase
    return ''.join(random.choice(numeros) for _ in range(longitud))

# Generar la lista de strings
lista_de_caracteres = [generar_caracter() + "\n" for _ in range(100)]

# Unir la lista en una sola cadena
resultado = ''.join(lista_de_caracteres)

# Guardar el resultado en un archivo .txt
nombre_archivo = "min_3.txt"
with open(nombre_archivo, "w") as archivo:
    archivo.write(resultado)

print(f"Se ha guardado el resultado en {nombre_archivo}")
```

2.2 Datasets aleatorios con letras mayúsculas.

Para la generación de los 5 datasets de letras mayúsculas, utilizamos el mismo script que en el apartado anterior pero en vez de escoger caracteres de “string.ascii_lowercase”, los escogemos de “string.ascii_upercase”

2.3 Datasets aleatorios con números.

Para la generación de 5 datasets de números aleatorios, realizamos lo mismo que en los apartados anteriores pero utilizando “string.digits”.

2.4 Datasets alfanuméricos aleatorios.

Para crear estos 5 datasets de caracteres ascii, utilizamos el mismo script que en los apartados anteriores pero pudiendo elegirse caracteres de entre “string.ascii_letters”, “string.digits” y “string.punctuation”.

2.5 Datasets creados con palabras de diccionario.

Para crear estos 5 datasets hemos utilizado distintos scripts, los cuales se explicaran en los siguientes subapartados. Cada uno de los datasets ha sido pensado para abordar diferentes requisitos de seguridad en contraseñas.

2.5.1 Palabras en claro. → palabras1

Para la generación del primer conjunto de palabras de diccionario, empleamos la biblioteca NLTK de Python. Esta librería permite obtener una selección de 100 palabras en inglés, las cuales guardamos en un archivo de texto. Con este conjunto de contraseñas hemos querido simular las contraseñas que se utilizaban en los inicios, ya que se tratan de palabras simples y no excesivamente largas. Este dataset será la base de los siguientes datasets de diccionario.
El script es el siguiente:

```
Unset
import nltk
import random

# Descargamos los datos necesarios para generar palabras reales
nltk.download("words")

from nltk.corpus import words

# Obtenemos una lista de palabras reales disponibles en el diccionario de palabras
de nltk
word_list = words.words()

# Generamos  100 palabras aleatorias
random_words = random.sample(word_list, 100)

# Abrimos el archivo "palabras1.txt" en modo escritura y guardamos las palabras
with open("palabras_hashed_1.txt", "w") as file:
  for word in random_words:
    file.write(word.lower() + "\n")

print("Se han generado y guardado 100 palabras en el archivo 'palabras1.txt'.")
```

2.5.2 Palabras con números. → palabras2

Nuestro segundo dataset está basado en la aparición de contraseñas más seguras debido a la obligación de utilizar dígitos. Actualmente, todos sitios web y organizaciones obligan a que las contraseñas de sus usuarios contengan al menos 1 número.
De esta manera, generamos el segundo dataset de este apartado añadiendo después de cada palabra generada en el primer dataset de 1 a 4 dígitos. Hemos decidido escoger este rango basándonos en el hecho de que la gente suele añadir números de 1 o 2 cifras fáciles de recordar o números que tengan un significado para ellos como el año de nacimiento.

2.5.3 Palabras con números y al menos una mayúscula.→palabras3

Por otro lado, otras organizaciones deciden aumentar el nivel de seguridad, requiriendo que las contraseñas de sus usuarios contengan al menos un número y, además, al menos una letra mayúscula. Basándonos en esta práctica, generamos el tercer conjunto de palabras de diccionario, compuesto por contraseñas que contienen de 1 a 4 dígitos y al menos una letra mayúscula en cualquier posición. Para ello, cogemos el dataset anterior, y cambiamos una letra aleatoria de cada palabra de minúscula a mayúscula. Esta decisión se basa en la observación de que los usuarios, al crear una contraseña que no cumple los requerimientos de la organización, deciden completarlo de manera sencilla, por lo que ponen una letra cualquiera en mayúscula.

2.5.4 Palabras con números, mayúsculas y símbolos. → palabras4

Nuestro cuarto dataset está basado en que, en los últimos tiempos, las organizaciones más exigentes están requiriendo que las contraseñas contengan al menos un símbolo. Para abordar este requisito, creamos el cuarto conjunto de contraseñas. Este conjunto incluye palabras con hasta 4 dígitos, al menos una letra mayúscula y un símbolo al final de cada palabra. Esto refleja la práctica común de agregar un carácter especial al final de una contraseña existente.

2.5.5 Palabras repetidas y en mayúsculas. → palabras5

Para el último dataset, pensamos en hacer algo diferente a los últimos 3 datasets. Decidimos concatenar la palabra con ella misma y cambiar todas las letras a mayúsculas.

2.6 Generación de hashes.

Para obtener los hashes de las contraseñas de los datasets generados, utilizamos la librería hashlib de python. Aplicamos las funciones hash SHA-256 y MD5.
El script utilizado para hashear con el algoritmo SHA-256 es el siguiente:

```
Unset
import hashlib
```

```
# Función que hashea una contraseña con SHA-256
def hashear_contrasena(contrasena):
    sha256 = hashlib.sha256()
    sha256.update(contrasena.encode('utf-8'))
    # Obtenemos el hash en formato hexadecimal
    hashed_password = sha256.hexdigest()
    return hashed_password

# Archivo de entrada y salida
archivo_entrada = "palabras5.txt"
archivo_salida = "palabras_hashed_5.txt"

try:
    with open(archivo_entrada, "r") as entrada, open(archivo_salida, "w") as
salida:
        for linea in entrada:
            # Limpiamos la línea de cualquier carácter no deseado (por
ejemplo, saltos de línea)
            linea = linea.strip()
            # Hasheamos la contraseña con SHA-256
            hashed_password = hashear_contrasena(linea)
            # Escribimos la contraseña hasheada en el archivo de salida
            salida.write(hashed_password + '\n')
    print(f"Las contraseñas se han hasheado correctamente con SHA-256 y se
han guardado en '{archivo_salida}'.")
except FileNotFoundError:
    print(f"El archivo '{archivo_entrada}' no fue encontrado.")
except Exception as e:
    print(f"Ocurrió un error: {str(e)}")
```

3. Estrategias utilizadas

3.1 Estrategia 1: Ataque de fuerza bruta alfabeto ASCII longitudes 3, 4, 5, 6 y 7.

El objetivo de esta estrategia es abarcar el mayor conjunto de datasets posibles y realizar un ataque general. Se busca romper por completo los de menor longitud y observar los resultados de los de mayor longitud. Además, esta estrategia nos sirve para analizar y poder escoger, en función de los resultados, el resto de estrategias.

El comando de john para esta estrategia sobre hashes SHA-256 es el siguiente:

```
Unset
time timeout 15m john --format=raw-sha256 --incremental=ascii --min-length=3
--max-length=7 --fork=4 "$archivo"
```

Especificamos el parámetro --fork=4 para que se creen 4 hilos. También establecemos un timeout de 15 minutos. Estas 2 prácticas se repiten en el resto de estrategias.

3.2 Estrategia 2: Ataque de diccionario con regla creada.

Esta estrategia está enfocada en romper las contraseñas de los datasets generados con palabras de diccionario. Para minimizar el número de estrategias utilizadas, en vez de crear distintas reglas para cada dataset, decidimos crear una única regla de John que realice las transformaciones necesarias para romper las contraseñas de todos los conjuntos de datos de diccionario.

El comando de john para esta estrategia sobre hashes SHA-256 es el siguiente:

```
Unset
time timeout 15m john --rule=My_rule --fork=4 --format=raw-sha256
--wordlist=diccionario.txt "$archivo"
```

La regla es la siguiente:

```
Unset
[List.Rules:My_rule]
# Password en claro --> palabras1
:
# Password en mayusculas y duplicada --> palabras5
u d
# Password con hasta 4 numeros al final --> palabras2
$[0-9]
$[0-9]$[0-9]
$[0-9]$[0-9]$[0-9]
$[0-9]$[0-9]$[0-9]$[0-9]
# Password con al menos 1 mayuscula, hasta 4 numeros al final
# y posibilidad de simbolo al final --> palabras3 y palabras4
T0
T0$[0-9]
T0$[0-9]$[0-9{~*#;\|`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T0$[0-9]$[0-9]$[0-9{~*#;\|`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T0$[0-9]$[0-9]$[0-9]$[0-9{~*#;\|`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T0$[0-9]$[0-9]$[0-9]$[0-9]$[{~*#;\|`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T1
T1$[0-9]
T1$[0-9]$[0-9{*#;\|`^!@#$%~^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T1$[0-9]$[0-9]$[0-9{*#;\|`^~!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
```

```
T1$[0-9]$[0-9]$[0-9]$[0-9{*#;\~|`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T1$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!@#$%^&*()_~+-='\"/.,<>?\[\]\{\}\\:}]
T2
T2$[0-9]
T2$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-='~\"/.,<>?\[\]\{\}\\:}]
T2$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^&*()~_+-='\"/.,<>?\[\]\{\}\\:}]
T2$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%~^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T2$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!@#~$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T3
T3$[0-9]
T3$[0-9]$[0-9{*#;\|~`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T3$[0-9]$[0-9]$[0-9{*#;~\|`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T3$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^~!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T3$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|~`^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T44$[0-9]
T4$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-=~'\"/.,<>?\[\]\{\}\\:}]
T4$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^&*(~)_+-='\"/.,<>?\[\]\{\}\\:}]
T4$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!@#$~%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T4$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!@~#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T5
T5$[0-9]
T5$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-='~\"/.,<>?\[\]\{\}\\:}]
T5$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^~&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T5$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!~@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T5$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`~^!@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T6
T6$[0-9]
T6$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-~='\"/.,<>?\[\]\{\}\\:}]
T6$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^&*~()_+-='\"/.,<>?\[\]\{\}\\:}]
T6$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!@#~$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T6$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!~@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T7
T7$[0-9]
T7$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-='~\"/.,<>?\[\]\{\}\\:}]
T7$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^&*~()_+-='\"/.,<>?\[\]\{\}\\:}]
T7$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!@#~$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T7$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!~@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T8
T8$[0-9]
T8$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-~='\"/.,<>?\[\]\{\}\\:}]
T8$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^&*~()_+-='\"/.,<>?\[\]\{\}\\:}]
T8$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!@#~$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T8$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!~@#$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
T9
T9$[0-9]
T9$[0-9]$[0-9{*#;\|`^!@#$%^&*()_+-='~\"/.,<>?\[\]\{\}\\:}]
T9$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%^&*()~_+-='\"/.,<>?\[\]\{\}\\:}]
T9$[0-9]$[0-9]$[0-9]$[0-9{*#;\|`^!@#$%~^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
```

```
T9$[0-9]$[0-9]$[0-9]$[0-9]$[{*#;\|`^!@#~$%^&*()_+-='\"/.,<>?\[\]\{\}\\:}]
```

Para que john reconozca esta regla, la añadimos al archivo /etc/john/john.conf. Esto debemos hacerlo como root ya que, en nuestro caso, el usuario predeterminado de la máquina virtual no tiene los suficientes permisos para hacerlo.

Es importante señalar que el orden de las transformaciones ha sido cuidadosamente planificado con el propósito de reducir al máximo el tiempo de ejecución de cada comando. Hemos dispuesto las reglas que tienen un espacio de búsqueda menor en las primeras líneas, de manera que se puedan encontrar primero las contraseñas de los datasets relacionados con estas reglas. Esto evita realizar pruebas innecesarias con las reglas que se encuentran más abajo.

También es relevante destacar que, al efectuar la transformación Toggle (Tn) (que consiste en cambiar una letra de minúscula a mayúscula en la posición n), se detectó un problema al aplicarla en la posición de letra 10 en la palabra "john". Esto se debía a que “john” no reconocía posiciones con dos dígitos. Como resultado, hemos detenido la transformación en T9. Por lo tanto, no será posible romper contraseñas que contengan una letra mayúscula en una posición igual o mayor a 10.

Además, al incorporar símbolos en las transformaciones, con el propósito de que "john" interprete los símbolos [,], {, } y \ como caracteres literales, se incluye el carácter de escape \ justo antes del carácter crítico.

3.3 Estrategia 3: Ataque de fuerza bruta alfabeto Alpha longitudes 6 y 7.

Esta estrategia se centra en el descifrado de contraseñas que constan únicamente de letras minúsculas y mayúsculas, con longitudes de 6 y 7 caracteres. Al reducir el conjunto de caracteres al alfabeto de letras mayúsculas y minúsculas, el espacio de búsqueda se reduce considerablemente, lo que mejora la eficiencia de la estrategia. También tratamos de especificar longitudes para que “john” no realice combinaciones con contraseñas de longitudes que no nos interesan.
El comando de john para esta estrategia sobre hashes SHA-256 es el siguiente:

```
Unset
time timeout 15m john --format=raw-sha256 --incremental=Alpha --min-length=6
--max-length=7 --fork=4 "$archivo"
```

3.4 Estrategia 4: Ataque de fuerza bruta alfabeto ASCII longitudes 3, 4 y 5.

El objetivo de esta estrategia es romper las contraseñas alfanuméricas y con símbolos de longitudes 3, 4 y 5. Al igual que en la estrategia anterior, se hace para reducir el espacio de búsqueda.
Teniendo en cuenta que el espacio de búsqueda del alfabeto ASCII es bastante grande, decidimos reducir la longitud de las contraseñas a probar. De esta manera, los resultados se obtendrán en un tiempo razonable y seremos capaces de romper un mayor número de contraseñas en los datasets cuya longitud es de 3, 4 o 5 caracteres.
El comando de john para esta estrategia sobre hashes SHA-256 es el siguiente:

```
Unset
time timeout 15m john --format=raw-sha256 --incremental=ascii --min-length=3
--max-length=5 --fork=4 "$archivo"
```

3.5 Estrategia 5: Ataque de fuerza bruta alfabeto mayúsculas longitud 7.

El objetivo de esta estrategia es romper las contraseñas compuestas únicamente por letras mayúsculas de longitud 7. De manera similar a estrategias previas, se ha diseñado con el fin de disminuir el espacio de búsqueda y agilizar el proceso. De esta manera, seremos capaces de romper un mayor número de contraseñas.
Optamos por centrarnos exclusivamente en este conjunto de datos debido a su considerable longitud, lo que podría resultar en que el proceso de descifrado con “john” sea extremadamente largo. Al llevar a cabo esta estrategia, hemos reducido al mínimo el tiempo requerido para buscar contraseñas en este conjunto de datos, ya que es la estrategia más eficaz para contraseñas generadas al azar con esta longitud y este alfabeto.
El comando de john que utilizamos sobre hashes SHA-256 es:

```
Unset
time timeout 15m john --format=raw-sha256 --incremental=upper --min-length=7
--max-length=7 --fork=4 "$archivo"
```

4. Resultados obtenidos

En cuanto al proceso de obtención de resultados, hemos diseñado un script de bash que automatiza la ejecución de cada estrategia con todos los datasets. La plantilla del script es la siguiente:

```
Unset
#!/bin/bash
```

```
archivos=("minus_hashed_3.txt" "minus_hashed_4.txt" "minus_hashed_5.txt"
 "minus_hashed_6.txt" "minus_hashed_7.txt" "mayus_hashed_3.txt"
 "mayus_hashed_4.txt" "mayus_hashed_5.txt" "mayus_hashed_6.txt"
 "mayus_hashed_7.txt" "num_hashed_3.txt" "num_hashed_4.txt"
 "num_hashed_5.txt" "num_hashed_6.txt" "num_hashed_7.txt"
 "alfanum_simb_hashed_3.txt" "alfanum_simb_hashed_4.txt"
 "alfanum_simb_hashed_5.txt" "alfanum_simb_hashed_6.txt"
 "alfanum_simb_hashed_7.txt" "palabras_hashed_1.txt" "palabras_hashed_2.txt"
 "palabras_hashed_3.txt" "palabras_hashed_4.txt" "palabras_hashed_5.txt")

# Loop a través de los archivos
for archivo in "${archivos[@]}"; do
  echo "ATAQUE NOMBRE DEL ATAQUE"
  echo "Ejecutando el comando para $archivo"
  rm ~/.john/john.pot
  rm ~/.john/john.log
  time timeout 15m comando de john
  john --format=raw-sha256 --show "$archivo"
  john --format=raw-sha256 --show "$archivo"
  python3.9 genera_estadisticas.py
  echo
```

De esta manera, ejecutamos el script 5 veces cada una con una estrategia distinta y obtenemos el número de contraseñas que se han conseguido romper, la media y la mediana de los tiempos en que se ha roto una contraseña.
Para calcular estas estadísticas (media y mediana), hemos implementado un script de python *genera_estadisticas.py* que contiene el siguiente código:

```
Unset
import re
```

```
import os
import statistics

# Funcion que converte  el formato de tiempo en segundos
def tiempo_a_segundos(tiempo):
    partes = tiempo.split(':')
    dias, horas, minutos, segundos = map(int, partes)
    return horas * 3600 + minutos * 60 + segundos

# Expandimos ~  a la ruta completa del directorio de inicio
ruta_archivo_log = os.path.expanduser("~/.john/john.log")

# Leemos el archivo .log
with open(ruta_archivo_log, "r") as archivo_log:
    lineas = archivo_log.readlines()

# Inicializamos una lista para almacenar los tiempos en segundos
tiempos_en_segundos = []

# Buscamos las líneas que contienen "+ Cracked" y extraemos los tiempos
for linea in lineas:
    if "+ Cracked" in linea:
        tiempo_match = re.search(r"(\d+:\d+:\d+:\d+)", linea)
        if tiempo_match:
            tiempo = tiempo_match.group(1)
            tiempos_en_segundos.append(tiempo_a_segundos(tiempo))

# Calculamos la media y la mediana de los tiempos en segundos
if tiempos_en_segundos:
    media = sum(tiempos_en_segundos) / len(tiempos_en_segundos)
    print(f"Media de los tiempos en segundos: {media:.2f}")
    mediana = statistics.median(tiempos_en_segundos)
    print(f"Mediana de los tiempos en segundos: {mediana:.2f}")
else:
    print("No se encontraron tiempos para calcular la media.")
```

Los resultados obtenidos se muestran en las siguientes tablas:

4.1 Resultados SHA-256

4.1.1 Estrategia 1: Ataque fuerza bruta código ASCII longitudes 3 a 7

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	100	7.44	2.00
Minúsculas	4	99	67.06	15.00
Minúsculas	5	99	32.16	15.00
Minúsculas	6	96	207.92	133.00
Minúsculas	7	21	327.43	258.00
Mayúsculas	3	91	103.36	89.00
Mayúsculas	4	61	430.00	422.00
Mayúsculas	5	67	190.45	92.00
Mayúsculas	6	43	339.51	270.00
Mayúsculas	7	0	-	-
Dígitos	3	100	2.34	2.00
Dígitos	4	100	1.20	1.00
Dígitos	5	100	10.48	10.00
Dígitos	6	100	12.21	8.00
Dígitos	7	100	329.32	314.00
Alfanuméricos	3	46	128.39	58.50
Alfanuméricos	4	14	331.43	237.50
Alfanuméricos	5	9	542.22	501.00
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	23	18.30	2.00
Palabras2	-	7	115.29	90.00
Palabras3	-	1	675.00	675.00
Palabras4	-	0	-	-
Palabras5	-	1	277.00	277.00

4.1.2 Estrategia 2: Ataque de diccionario

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	0	-	-
Minúsculas	4	0	-	-
Minúsculas	5	0	-	-
Minúsculas	6	0	-	-
Minúsculas	7	0	-	-
Mayúsculas	3	0	-	-
Mayúsculas	4	0	-	-
Mayúsculas	5	0	-	-
Mayúsculas	6	0	-	-
Mayúsculas	7	0	-	-
Dígitos	3	0	-	-
Dígitos	4	0	-	-
Dígitos	5	0	-	-
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	0	-	-
Alfanuméricos	4	0	-	-
Alfanuméricos	5	0	-	-
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	100	9.16	0.00
Palabras2	-	100	11.43	0.00
Palabras3	-	97	9.97	10.00
Palabras4	-	97	11.10	10.00
Palabras5	-	100	0.00	0.00

4.1.3 Estrategia 3: Ataque de fuerza bruta mayúsculas y minúsculas longitudes 6 y 7

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	0	-	-
Minúsculas	4	0	-	-
Minúsculas	5	0	-	-
Minúsculas	6	100	19.46	19.00
Minúsculas	7	95	346.60	247.00
Mayúsculas	3	0	-	-
Mayúsculas	4	0	-	-
Mayúsculas	5	0	-	-
Mayúsculas	6	67	219.33	75.00
Mayúsculas	7	15	567.07	527.00
Dígitos	3	0	-	-
Dígitos	4	0	-	-
Dígitos	5	0	-	-
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	0	-	-
Alfanuméricos	4	0	-	-
Alfanuméricos	5	0	-	-
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	18	15.33	1.50
Palabras2	-	7	115.29	90.00
Palabras3	-	0	-	-
Palabras4	-	0	-	-
Palabras5	-	1	26.00	26.00

4.1.4 Estrategia 4: Ataque de fuerza bruta ASCII longitudes 3, 4 y 5

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	100	0.67	0.00
Minúsculas	4	100	3.13	1.00
Minúsculas	5	100	3.26	2.00
Minúsculas	6	0	-	-
Minúsculas	7	0	-	-
Mayúsculas	3	100	24.14	6.00
Mayúsculas	4	100	20.21	17.00
Mayúsculas	5	100	21.32	10.00
Mayúsculas	6	0	-	-
Mayúsculas	7	0	-	-
Dígitos	3	100	0.11	0.00
Dígitos	4	100	0.00	0.00
Dígitos	5	100	1.04	1.00
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	100	87.38	137.00
Alfanuméricos	4	100	138.97	80.50
Alfanuméricos	5	100	135.22	133.00
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	6	0.33	0.00
Palabras2	-	0	-	-
Palabras3	-	0	-	-
Palabras4	-	0	-	-
Palabras5	-	0	-	-

4.1.5 Estrategia 5: Ataque de fuerza bruta mayúsculas longitud fija 7

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	0	-	-
Minúsculas	4	0	-	-
Minúsculas	5	0	-	-
Minúsculas	6	0	-	-
Minúsculas	7	0	-	-
Mayúsculas	3	0	-	-
Mayúsculas	4	0	-	-
Mayúsculas	5	0	-	-
Mayúsculas	6	0	-	-
Mayúsculas	7	100	184.15	175.00
Dígitos	3	0	-	-
Dígitos	4	0	-	-
Dígitos	5	0	-	-
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	0	-	-
Alfanuméricos	4	0	-	-
Alfanuméricos	5	0	-	-
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	0	-	-
Palabras2	-	0	-	-
Palabras3	-	0	-	-
Palabras4	-	0	-	-
Palabras5	-	0	-	-

4.2 Resultados MD5

4.2.1 Estrategia 1: Ataque fuerza bruta código ASCII longitudes 3 a 7

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	100	3.47	1.50
Minúsculas	4	100	28.32	5.00
Minúsculas	5	100	19.29	8.00
Minúsculas	6	100	97.53	60.00
Minúsculas	7	42	355.55	330.00
Mayúsculas	3	91	41.86	36.00
Mayúsculas	4	84	256.30	244.00
Mayúsculas	5	81	195.69	44.00
Mayúsculas	6	53	216.53	144.00
Mayúsculas	7	3	810.33	791.00
Dígitos	3	100	1.60	1.00
Dígitos	4	100	0.65	1.00
Dígitos	5	100	4.73	5.00
Dígitos	6	100	5.73	4.00
Dígitos	7	100	128.28	122.00
Alfanuméricos	3	46	52.15	24.00
Alfanuméricos	4	24	345.54	314.50
Alfanuméricos	5	13	316.15	265.00
Alfanuméricos	6	3	464.00	462.00
Alfanuméricos	7	0	-	-
Palabras1	-	24	24.92	3.00
Palabras2	-	5	84.00	13.00
Palabras3	-	2	534.50	534.50
Palabras4	-	0	-	-
Palabras5	-	0	-	-

4.2.2 Estrategia 2: Ataque de diccionario

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	0	-	-
Minúsculas	4	0	-	-
Minúsculas	5	0	-	-
Minúsculas	6	0	-	-
Minúsculas	7	0	-	-
Mayúsculas	3	0	-	-
Mayúsculas	4	0	-	-
Mayúsculas	5	0	-	-
Mayúsculas	6	0	-	-
Mayúsculas	7	0	-	-
Dígitos	3	0	-	-
Dígitos	4	0	-	-
Dígitos	5	0	-	-
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	0	-	-
Alfanuméricos	4	0	-	-
Alfanuméricos	5	0	-	-
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	100	5.41	0.00
Palabras2	-	100	7.36	1.00
Palabras3	-	97	6.07	6.00
Palabras4	-	97	7.09	6.00
Palabras5	-	100	0.00	0.00

4.2.3 Estrategia 3: Ataque de fuerza bruta mayúsculas y minúsculas longitudes 6 y 7

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	0	-	-
Minúsculas	4	0	-	-
Minúsculas	5	0	-	-
Minúsculas	6	100	8.62	8.00
Minúsculas	7	100	149.69	100.00
Mayúsculas	3	0	-	-
Mayúsculas	4	0	-	-
Mayúsculas	5	0	-	-
Mayúsculas	6	94	242.87	87.50
Mayúsculas	7	30	343.90	320.00
Dígitos	3	0	-	-
Dígitos	4	0	-	-
Dígitos	5	0	-	-
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	0	-	-
Alfanuméricos	4	0	-	-
Alfanuméricos	5	0	-	-
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-			
Palabras2	-	0	-	-
Palabras3	-	0	-	-
Palabras4	-	0	-	-
Palabras5	-	1	6.00	6.00

4.2.4 Estrategia 4: Ataque de fuerza bruta ASCII longitudes 3, 4 y 5

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	100	0.25	0.00
Minúsculas	4	100	1.45	1.00
Minúsculas	5	100	1.81	1.00
Minúsculas	6	0	-	-
Minúsculas	7	0	-	-
Mayúsculas	3	100	8.21	3.00
Mayúsculas	4	100	8.92	9.00
Mayúsculas	5	100	9.48	5.00
Mayúsculas	6	0	-	-
Mayúsculas	7	0	-	-
Dígitos	3	100	0.05	0.00
Dígitos	4	100	0.00	0.00
Dígitos	5	100	0.53	1.00
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	100	34.73	53.00
Alfanuméricos	4	100	55.39	33.50
Alfanuméricos	5	100	54.29	54.00
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	24	24.92	3.00
Palabras2	-	1	0.00	0.00
Palabras3	-	0	-	-
Palabras4	-	0	-	-
Palabras5	-	0	-	-

4.2.5 Estrategia 5: Ataque de fuerza bruta mayúsculas longitud fija 7

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	0	-	-
Minúsculas	4	0	-	-
Minúsculas	5	0	-	-
Minúsculas	6	0	-	-
Minúsculas	7	0	-	-
Mayúsculas	3	0	-	-
Mayúsculas	4	0	-	-
Mayúsculas	5	0	-	-
Mayúsculas	6	0	-	-
Mayúsculas	7	100	67.19	64.50
Dígitos	3	0	-	-
Dígitos	4	0	-	-
Dígitos	5	0	-	-
Dígitos	6	0	-	-
Dígitos	7	0	-	-
Alfanuméricos	3	0	-	-
Alfanuméricos	4	0	-	-
Alfanuméricos	5	0	-	-
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	0	-	-
Palabras2	-	0	-	-
Palabras3	-	0	-	-
Palabras4	-	0	-	-
Palabras5	-	0	-	-

5. Análisis de resultados

5.1 Estrategia 1: Ataque de fuerza bruta alfabeto ASCII longitudes 3, 4, 5, 6 y 7.

Prestando atención a los resultados de esta estrategia, podemos hacernos una idea general de tiempos y el número de contraseñas que “john” es capaz de romper para cada dataset.
En cuanto a la longitud de las contraseñas, se observa una clara tendencia: a medida que la longitud de las contraseñas aumenta, ya sean compuestas por letras minúsculas, mayúsculas o caracteres alfanuméricos, la cantidad de contraseñas vulneradas disminuye significativamente. Este declive se debe a que las contraseñas más largas son intrínsecamente más sólidas y, por lo tanto, más resistentes a los ataques de fuerza bruta.
La razón detrás de este fenómeno radica en la creciente complejidad que se añade a medida que se incrementa el número de caracteres y, por consiguiente, las combinaciones posibles. "John", en las estrategias de fuerza bruta suele probar todas las combinaciones en un orden ascendente de longitud, por lo que las contraseñas más cortas se prueban antes que las más largas.
No obstante, podemos observar que existen casos puntuales en que esto no se corresponde con nuestros resultados. Un ejemplo de esto es cuando las contraseñas de dígitos de longitud 7 se rompen antes que las contraseñas de letras mayúsculas de longitud 5. Esto se puede atribuir a la estructura de la estrategia --incremental=ascii en "John the Ripper". Los dígitos tienen un espacio de búsqueda relativamente más reducido en comparación con otros tipos de caracteres, como letras mayúsculas y minúsculas. Como resultado, "john" puede dar prioridad a probar combinaciones con dígitos, ya que se espera que recorra ese espacio de búsqueda más rápidamente.
Al comparar la composición de caracteres en los conjuntos de datos, es evidente que las contraseñas compuestas por caracteres alfanuméricos y símbolos se rompen con mucha menos frecuencia y son más difíciles de descifrar en comparación con las demás. Esto se debe a que el espacio de búsqueda es considerablemente más amplio, y "john" tiende a probar estas contraseñas en una etapa más avanzada de la búsqueda.

En lo que respecta a la media de los tiempos, esta cifra actúa como un indicador que nos permite estimar y valorar el esfuerzo y el tiempo necesario para romper contraseñas de diversas longitudes y conjuntos de caracteres. Un valor de tiempo bajo refleja que las contraseñas son relativamente fáciles de crackear, mientras que un valor alto indica que las contraseñas son más resistentes.
Al observar las contraseñas compuestas solo por dígitos, vemos que el 100% de ellas se ha quebrantado, independientemente de su longitud. Sin embargo, al analizar los tiempos, se destaca que las contraseñas de longitud 7 han requerido un tiempo significativamente mayor para ser vulneradas en comparación con las más cortas. Esto subraya que, aunque todas las contraseñas de dígitos se han descifrado, las de longitud 7 han resultado más complicadas de romper (mayor tiempo).
Por otra parte, las métricas de media y mediana en conjunto proporcionan información valiosa sobre la distribución de los tiempos en que las contraseñas son vulneradas para cada conjunto. Tomemos como ejemplo el dataset de contraseñas en minúsculas, donde la media es considerablemente mayor que la mediana. Esto sugiere que la mayoría de las contraseñas se han descifrado antes del tiempo señalado por el valor de la media. En este

caso, podemos deducir que el proceso de “john” ha experimentado dificultades prolongadas al intentar descifrar algunas contraseñas particularmente desafiantes.
En contraste, cuando tanto la media como la mediana tienen valores similares, esto indica que la distribución en el tiempo en que se rompen las contraseñas ha sido más uniforme. Es decir, desde el inicio del proceso hasta su conclusión, las contraseñas se han ido descifrando de manera uniforme a lo largo del tiempo.

5.2 Estrategia 2: Ataque de diccionario con regla creada

Esta estrategia se comporta según lo previsto, ya que logra romper la gran mayoría de las contraseñas generadas a partir de palabras de diccionario. Las tres contraseñas que no se consiguen romper en el tercer y cuarto dataset son aquellas que tienen una letra mayúscula en una posición que es la décima o posterior dentro de la palabra. Como mencionamos anteriormente, John the Ripper no permite realizar la operación toggle a las letras que ocupan posiciones de dos dígitos.
Por otra parte, era de esperar que esta estrategia no pudiera romper ninguna contraseña en los demás datasets, ya que han sido generadas aleatoriamente y no forman palabras.

5.3 Estrategia 3: Ataque de fuerza bruta alfabeto Alpha longitudes 6 y 7.

Observando los resultados, podemos notar una notable mejora en el porcentaje de contraseñas crackeadas de los datasets en minúsculas y mayúsculas con longitudes de 6 y 7 caracteres, en comparación con la primera estrategia. Esta mejora se debe a la reducción del espacio de búsqueda, ya que se limita la búsqueda a letras mayúsculas y minúsculas, así como a longitudes específicas de 6 y 7 caracteres.

5.4 Estrategia 4: Ataque de fuerza bruta alfabeto ASCII longitudes 3, 4 y 5.

Esta estrategia logra un 100% de éxito en la ruptura de contraseñas en los datasets para los que estaba enfocada. Además, destaca por sus tiempos medios de rotura, los cuales son notablemente bajos, especialmente en los conjuntos de datos que contienen contraseñas en minúsculas y mayúsculas. La razón detrás de este buen rendimiento radica en la longitud relativamente corta de las contraseñas que se prueban mediante fuerza bruta, ya que el espacio de búsqueda se ve considerablemente reducido.

Es importante resaltar la marcada diferencia en los resultados entre esta estrategia y la Estrategia 1, ambas basadas en ataques de fuerza bruta. El simple cambio en la longitud de las contraseñas que se prueban, orientándose más hacia conjuntos de datos específicos, tiene un impacto significativo en la cantidad de contraseñas vulneradas.

5.5 Estrategia 5: Ataque de fuerza bruta alfabeto mayúsculas longitud 7.

Esta estrategia logra romper todas las contraseñas para las que estaba orientada.

5.6 Comparación entre SHA-256 y MD5

A continuación, vamos a combinar las mayores cifras de contraseñas que se han conseguido romper para cada algoritmo de hashing en una única tabla. Si 2 estrategias han

roto la misma cantidad de contraseñas, escogemos la estrategia que menor media de tiempos tenga.
Cada color indica de qué estrategia se han escogido los datos.

- Amarillo: estrategia 1 → fuerza bruta ASCII longitudes 3,4,5,6 y 7.
- Azul: estrategia 2 → regla con palabras de diccionario.
- Verde: estratega 3 →fuerza bruta mayusculas y minusculas longitudes 5 y 6
- Rojo: estrategia 4 → fuerza bruta ascii longitudes 3,4,5
- Naranja: estrategia 5 → fuerza bruta mayúsculas de longitud 7.
- Blanco: ninguna estrategia

5.6.1 SHA-256

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	100	0.67	0.00
Minúsculas	4	100	3.13	1.00
Minúsculas	5	100	3.26	2.00
Minúsculas	6	100	19.46	19.00
Minúsculas	7	95	346.60	247.00
Mayúsculas	3	100	24.14	6.00
Mayúsculas	4	100	20.21	17.00
Mayúsculas	5	100	21.32	10.00
Mayúsculas	6	67	219.33	75.00
Mayúsculas	7	100	184.15	175.00
Dígitos	3	100	2.34	2.00
Dígitos	4	100	1.20	1.00
Dígitos	5	100	10.48	10.00
Dígitos	6	100	12.21	8.00
Dígitos	7	100	329.32	314.00
Alfanuméricos	3	100	87.38	137.00
Alfanuméricos	4	100	138.97	80.50
Alfanuméricos	5	100	135.22	133.00
Alfanuméricos	6	0	-	-
Alfanuméricos	7	0	-	-
Palabras1	-	100	9.16	1.00
Palabras2	-	100	14.32	1.00
Palabras3	-	97	14.62	16.00
Palabras4	-	97	18.33	17.00
Palabras5	-	100	1.00	1.00
TOTAL	-	2256	1616.83	1273.5

En esta tabla, llaman la atención los resultados relacionados con los datasets alfanuméricos de longitudes 6 y 7. En nuestro enfoque práctico, priorizamos romper la mayor cantidad posible de contraseñas, y por lo tanto, decidimos no centrarnos en estos datasets en particular. Esto se debe a que las contraseñas alfanuméricas de longitudes 6 y 7 tienen un

espacio de búsqueda extremadamente amplio, lo que haría que romperlas fuera un desafío considerable. En su lugar, nos hemos concentrado en maximizar el número de contraseñas rotas en el resto de datasets..

Por otra parte, en total, al combinar las cinco estrategias, hemos logrado romper un máximo de 2256 contraseñas, lo que equivale al 90.24% de la cantidad inicial de hashes que teníamos como objetivo crackear. Este porcentaje es bastante alto y sugiere que la selección de estrategias ha sido efectiva y exitosa en nuestro enfoque.

5.6.2 MD5

Dataset	Longitud	Porcentaje contraseñas rotas (%)	Media de los tiempos en que se ha roto una contraseña (s)	Mediana de los tiempos en que se ha roto una contraseña (s)
Minúsculas	3	100	0.25	0.00
Minúsculas	4	100	1.45	1.00
Minúsculas	5	100	1.81	1.00
Minúsculas	6	100	8.62	8.00
Minúsculas	7	100	149.69	100.00
Mayúsculas	3	100	8.21	3.00
Mayúsculas	4	100	8.92	9.00
Mayúsculas	5	100	9.48	5.00
Mayúsculas	6	94	242.87	87.50
Mayúsculas	7	100	67.19	64.50
Dígitos	3	100	0.05	0.00
Dígitos	4	100	0.00	0.00
Dígitos	5	100	0.53	1.00
Dígitos	6	100	5.73	4.00
Dígitos	7	100	128.28	122.00
Alfanuméricos	3	100	34.73	53.00
Alfanuméricos	4	100	55.39	33.50
Alfanuméricos	5	100	54.29	54.00
Alfanuméricos	6	3	464.00	462.00
Alfanuméricos	7	0	-	-
Palabras1	-	100	5.41	0.00
Palabras2	-	100	7.36	1.00
Palabras3	-	97	6.07	6.00
Palabras4	-	97	7.09	6.00
Palabras5	-	100	0.00	0.00
TOTAL		2291	1267.42	1021.5

Comparando las dos tablas con los mejores resultados para SHA-256 y MD5, se aprecia que el número total de contraseñas es ligeramente mayor en las generadas mediante MD5. Con SHA-256 se rompe el 90.24% de las palabras mientras que con MD5 se rompe el 91,64%. Sin embargo, la diferencia más notable se manifiesta en los tiempos. Tanto el

tiempo promedio como la mediana son consistentemente más bajos en los resultados de MD5. Esto podemos verlo reflejado claramente en el total.
En consecuencia, podemos concluir que John es capaz de romper contraseñas generadas con hashes MD5 más rápidamente que generadas con SHA-256.

6. Conclusiones

En resumen, la longitud y la complejidad de las contraseñas desempeñan un papel crucial en su resistencia a los ataques de fuerza bruta. Las contraseñas más cortas y menos complejas (menor espacio de búsqueda) son más fáciles de romper, mientras que las contraseñas alfanuméricas y con símbolos tienden a ser más seguras. Es importante que los usuarios seleccionen contraseñas fuertes y únicas para proteger sus cuentas.

Nuestra investigación también destacó la relevancia de las estrategias empleadas en la búsqueda de contraseñas. La elección de la estrategia puede llevar a resultados sobresalientes o deficientes según el conjunto de datos analizado.

Por otra parte, los datos revelaron que SHA-256 es más resistente a los ataques de fuerza bruta en comparación con MD5. Las contraseñas generadas con MD5 se rompen en menos tiempo, tanto en términos de tiempo medio como de mediana.

En cuanto a los ataques de diccionario, observamos que son mucho más rápidos en comparación con los ataques de fuerza bruta. Este tipo de ataque puede resultar extremadamente útil y eficiente gracias a que se pueden realizar distintas transformaciones a las palabras.

En un sentido más personal, hemos dedicado mucho tiempo y puesto un gran esfuerzo en esta práctica. Además de la planificación, información, desarrollo e implementación de las estrategias hemos llevamos a cabo pruebas exhaustivas que abarcaron más de 24 horas en total. Todo este trabajo nos ha permitido aprender sobre la fortaleza de las contraseñas, la creación de estrategias efectivas para romperlas y el manejo competente de la herramienta "John the Ripper".