Password Guessing y la Seguridad detrás de las Contraseñas

Silvio Orozco Vizquerra
Administración y Seguridad de Sistemas
Universidad de Granada

01 Historia de Usable Privacy and Security Research

02 Historia de la Contraseña

03 ¿Cómo funciona una contraseña?

04 Cracking de una contraseña

05 Escoger contraseñas Seguras

06 Caso Práctico Ataque de Diccionario

INTRODUCCIÓN

Las contraseñas han sido el método predominante para autenticar usuarios y acceder a recursos restringidos. Evaluar su historia y conocer los posibles ataques nos permiten ser conscientes de su importancia y de las posibles medidas de protección que debemos tener.

01 HISTORIA DE UPS

You could enter a subtitle here if you need it

LINEA DE TIEMPO

1979

Morris y Thompson un caso de estudio de las contraseñas

1975

Saltzer y Schroeders

Sistema usable debe ser seguro

1995

Zurko y Simon User Centered Security

Resolver problemas de seguridad en el diseño con usuarios

HISTORIA USABLE PRIVACY AND SECURITY RESEARCH

1. 1975: Saltzer y Schroeders fueron los primeros en observar que para que un sistema informático sea usable debe ser seguro. Uno de los 8 principios era la aceptabilidad psicológica encajando la imagen mental del usuario.
2. 1979: Robert Morris y Ken Thompson publican paper "Password Security: A case History". Las metas de las contraseñas debe ser la mínima inconveniencia posible. Salting y hashing para no guardar contraseñas en texto plano.
3. 3. 1995 Zurko y Simon desarrollan el User Centered Security.

02

HISTORIA DE LA CONTRASEÑA

LÍNEA DE TIEMPO

1960

IBM Y MIT simultáneamente inventan la contraseña.

1969

Hoffman ve los problemas de un systematic attack.

1979

Morris y Thompson

Estudian las contraseñas y el salting

1985

Guidelines en el Green Book de US

2013

Cheswick

Rethinking Passwords

HISTORIA CONTRASEÑA

1. 1960: IBM trabaja en contraseñas para el sistema Sabre de reservaciones y MIT trabaja en el proyecto Compatible Time Sharing System CTTS.
2. 1969: Hoffman observa por primera vez que era fácil comprometer a los sistemas por medio de ataques sistemáticos.
3. 1979: Morris y Thompson descubrían 1ue 2831 de 3289 contraseñas podían ser descubiertas a través de un offline dictionary attack.

HISTORIA CONTRASEÑA

4. 1985: El departamento de Defensa de Estados Unidos establece en su green book el Password Management Guideline. Con los requerimientos y reglas que siguieron las siguientes 3 décadas. Algunas de las pautas importantes a mencionar son:

- Dar un tiempo de vida a las contraseñas (un año para la mayoría de los casos y un mes para sistemas de alta protección)
- Permitir un máximo de 3 intentos de contraseña antes de que una alarma sea generada
- Proponer tamaños de entre 6 y 8 caracteres.

5. Rethinking passwords

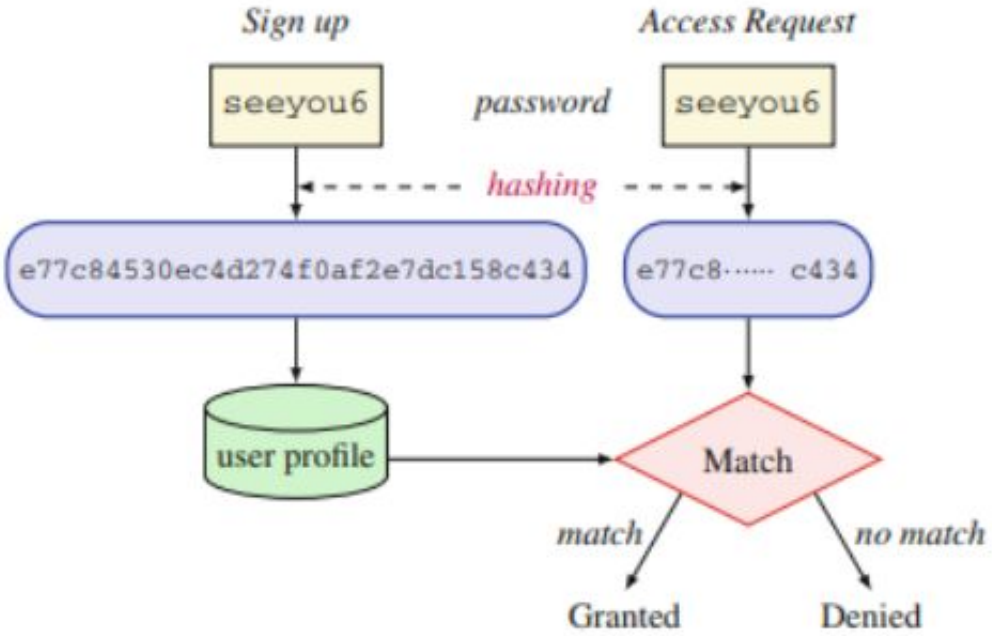
- Ahora se han agregado otras reglas tales como requerir un mix de letras minúsculas y mayúsculas, símbolos y contraseñas más largas.

03
¿CÓMO FUNCIONA LA CONTRASEÑA?

DEFINICIÓN CONTRASEÑA

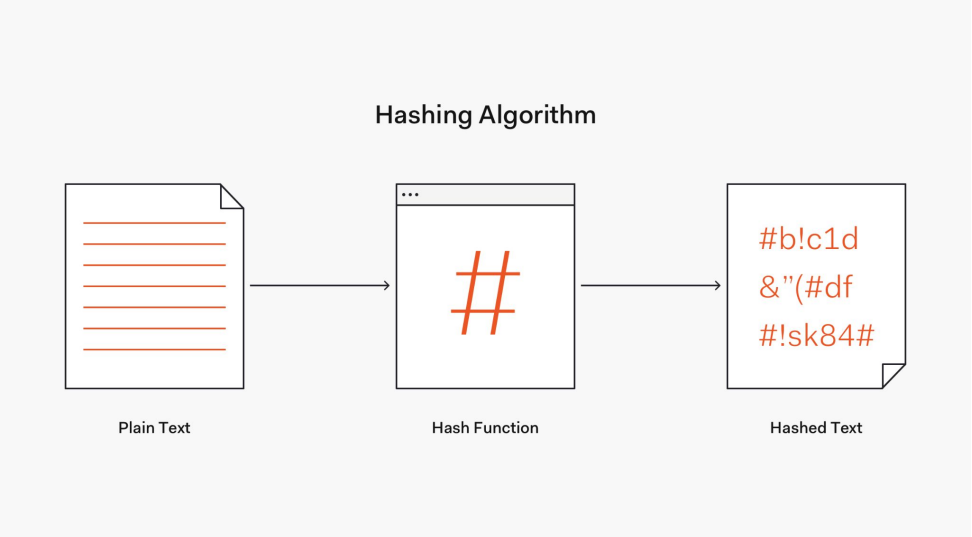
1. El concepto de contraseña en sistemas de seguridad y computación lo podemos definir como un string de carácteres secretos que solo el usuario conoce, de forma que al ser pasado por un algoritmo de encriptación hash y generando un hash code pueda ser guardado en en el servidor para proveer acceso a data. De forma que se utiliza tanto una contraseña como otra forma de identificación del usuario tales como nombre de usuario o un correo electrónico (Hu, 2017).

Mecanismo de contraseñas para acceder a un recurso



HASH FUNCTION

La función es hash es una función irreversible que mapea una contraseña p a un código cifrado h de la forma $f(p) = h$ y que f-1 es inexistente, por lo que la única –manera de que un hacker acceda a la contraseña es intentando con diferentes strings viendo si alguno hace match con el código hash de la contraseña actual.



04

CRACKING UNA CONTRASEÑA

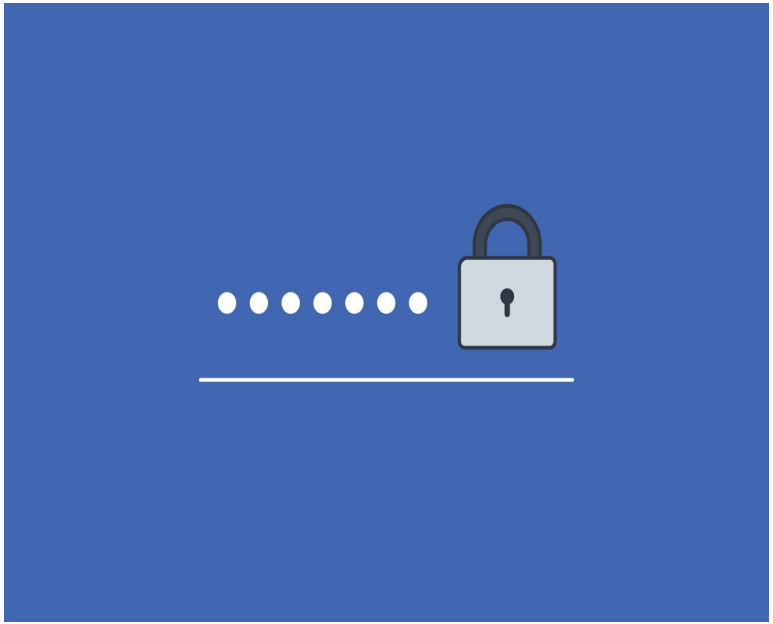
DEFINICIÓN CRACKING/DESCIFRADO

Cracking a password es la técnica de infringir la seguridad de un sistema para encontrar la contraseña de un usuario y asumir su identidad para ingresar al sistema (Cazier y Medlin, 2006). Se define el riesgo del ataque sobre lo siguiente:

1. El tiempo que toma al atacante computar y probar una contraseña
2. La cantidad de recursos que tiene el atacante
3. La probabilidad de éxito del método del atacante

FORMAS DE ATAQUE

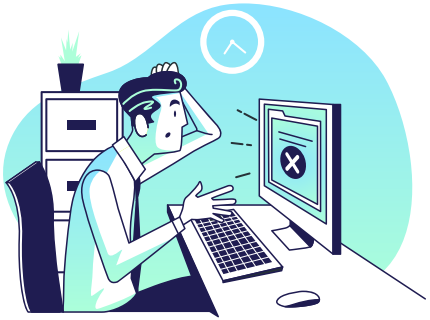
1. ONLINE ATTACKS: Ataques realizados en vivo. Mecanismo como Captcha y restringir máxima cantidad de intentos es suficiente.
2. OFFLINE ATTACKS: Los atacantes tienen acceso a la base de datos de hashes de las contraseñas y luego intentan el crack sin alertar al target host.



04

ATAQUES OFFLINE COMUNES

Problema



Problema

Los humanos tendemos a escoger contraseñas fáciles de recordar y que tengan algo familiar para nosotros.

Consecuencia

Los hackers pueden realizar ataques a nuestras contraseñas si no se tiene la seguridad suficiente.

Brute Force Attack

Los ataques de fuerza bruta se caracterizan por realizar una búsqueda exhaustiva calculando el hash de todas las posibles combinaciones de strings para un conjunto de caracteres y de una longitud m. En teoría, tiene un éxito del 100% aunque es un problema no computable por la cantidad de tiempo y cómo crece exponencialmente a medida que la contraseña es más larga. Finalmente, es ineficiente porque se prueba grandes cantidades de combinaciones poco probables como contraseña (Hu,2017).

Dictionary Attack

Los ataques de diccionario son útiles al pensar que los usuarios utilizan contraseñas humanamente memorizables por lo que existen palabras y variaciones comunes que un hacker puede intentar en lugar de usar combinaciones random como en el ataque de fuerza bruta. Los problemas con esta técnica es que el diccionario debe contener una vocabulario “enorme o completo” y que los usuarios que conocen este ataque realizan pequeños cambios a la palabra lo cual lo hace más difícil para este ataque.

Hybrid Dictionary Attack

Estos ataques combinan los ataques por diccionarios más un poco de fuerza bruta al adjuntas pequeños strings comunes al diccionario tales como caracteres numéricos y símbolos. Este ataque muchas veces sigue siendo inviable por la cantidad de tiempo que tomaría y depende en gran medida de la cantidad de caracteres que se adjunta (Yannis, 2013).

Rule Based Dictionary Attack

Este ataque se caracteriza por ser igual que el hybrid dictionary attack pero agrega reglas sobre adjuntar carácteres o palabras en el diccionario dependiendo de contraseñas comunes. Por lo tanto, el éxito del ataque depende en gran medida de que tan buenas pueden llegar a ser las reglas que tenemos que podrían reducir los tiempos para descifrar una contraseña.

Rule Based Dictionary Attack

Este ataque se caracteriza por ser igual que el hybrid dictionary attack pero agrega reglas sobre adjuntar carácteres o palabras en el diccionario dependiendo de contraseñas comunes. Por lo tanto, el éxito del ataque depende en gran medida de que tan buenas pueden llegar a ser las reglas que tenemos que podrían reducir los tiempos para descifrar una contraseña.

Free Public Hash Databases and Table Lookup Attack

Para este ataque puede ser simple buscar bases de datos de hashes públicos y comunes que es parecido al ataque de fuerza bruta solo que simplemente sin el tiempo de computar el hash sumado a que solo se realizan las combinaciones muy sencillas. Este método es efectivo si el usuario tiene una contraseña muy sencilla de una palabra común pero no garantiza el éxito (Yannis, 2013). Al no realizar el cálculo del hash es mucho más rápido que un ataque de diccionario. .

Rainbow Table Attack

Rainbow Table Attack es una variación del Table Lookup Attack. Se basa en la técnica de Time-Memory Trade-Off (TMTO) que busca guardar una cantidad mucho menor de hash codes que aún representando una cantidad enorme de contraseñas. La idea es crear una cadena de password-hash de longitud k que cubre k potenciales contraseñas y sus hash codes (Hu, 2017). Para optimizar los hashes se guardan únicamente el primera y última en cada cadena a memoria. Utilizando el mismo espacio de memoria, cubre k más palabras utilizando el mismo diccionario pero k veces más lento que un ataque de diccionario. El mayor problema de este método es las colisiones cuando dos diferentes hash son reducidos a la misma contraseña y que la función reducción se comporte correctamente. El salting es otro problema. Algunos ejemplos de software que utilizan esta técnica son Crackstation, Hashkiller y Hashcat.

Markov Model Attack

Al reconocer que los usuarios prefieren contraseñas fáciles de recordar, se puede pensar en diccionarios inteligentes con contraseñas que son probables que los usuarios puedan generar. Narayanan y Shmatikov introdujeron este método utilizando el modelo de Markov para generar este diccionario inteligente. Se crea el diccionario basado en la probabilidad de un carácter en una secuencia dado el lenguaje nativo del usuario. Estos modelos reducen drásticamente el espacio de búsqueda y se crean diccionarios útiles y probables para descifrar contraseñas con efectividades mayores al 60% (Hu, 2017).

Attack with Probabilistic Context Free Grammar

Estos ataques se basan en estimar la probabilidad de la contraseña de un usuario a partir de un training set de contraseñas reales filtradas y crear un context-free grammar que pueda estimar la formación de un string (Hu, 2017).

John the Ripper Attack

John the Ripper es un software abierto para descifrar contraseñas que combina ataques de diccionarios, rainbow tables y ataques de fuerza bruta con una variedad de reglas para la composición de palabras (Hu, 2017).

Artificial Intelligence Attack

Este acercamiento es el más novedoso, dado que se ha probado que con inteligencia artificial y un set de contraseñas filtradas de más de 43 millones de perfiles de LinkedIn se ha logrado descifrar más de un cuarto de las contraseñas (Hutson, 2017). PassGan usa Generative Adversarial Network GAN (deep learning) para que de forma autónoma aprende la distribución real de contraseñas de password leaks y generar adivinanzas de contraseñas de alta calidad logrando mejorar en gran medida los métodos originales de descifrado de contraseña (Hitaj, Gatsi, Ateniese y Perez-Cruz, 2019). Este proyecto se basa en utilizar deep learning de forma que se puedan generar muestras de la misma distribución del training set.

05
ESCOGER CONTRASEÑAS SEGURAS

Medida Entropia

Entropía: Introducida por Claude Shannon y definida como el número promedio de dígitos binarios requeridos por cada letra del lenguaje original. El National Institute of Standards and Technology (NIST) proveyó un valor para esta métrica expresando la incertidumbre de una contraseña expresada en número de bits indicando que una mayor incertidumbre se tiene una contraseña más difícil de adivinar. Para una contraseña w de tamaño m de un conjunto de caracteres de tamaño N, la entropía se define como:

$$H(w) = log_2(N^m) = m * log_2(N)$$

Entropía National Institute of Standards and Technology

.

- +4 Por el primer carácter
- +2 Por cada carácter entre el 2ndo y el 8vo
- +1.5 Por cada carácter entre el 9no y el 20vo
- +6 Por usar mayúsculas y caracteres no alfabéticos
- +6 No en diccionarios

Medida Password Quality Indicator

El PQI se basa en la idea anterior, consistiendo en una contraseña w con un par D (distancia Levenshtein la métrica que se utiliza es la distancia Levenshtein que es el mínimo número de operaciones para transferir una palabra a otra.) y L que la longitud efectiva de p. Se define con la siguiente fórmula dónde m es el tamaño de la contraseña y N el tamaño del conjunto de caracteres posibles.

$$L = m * log_{10}(N)$$

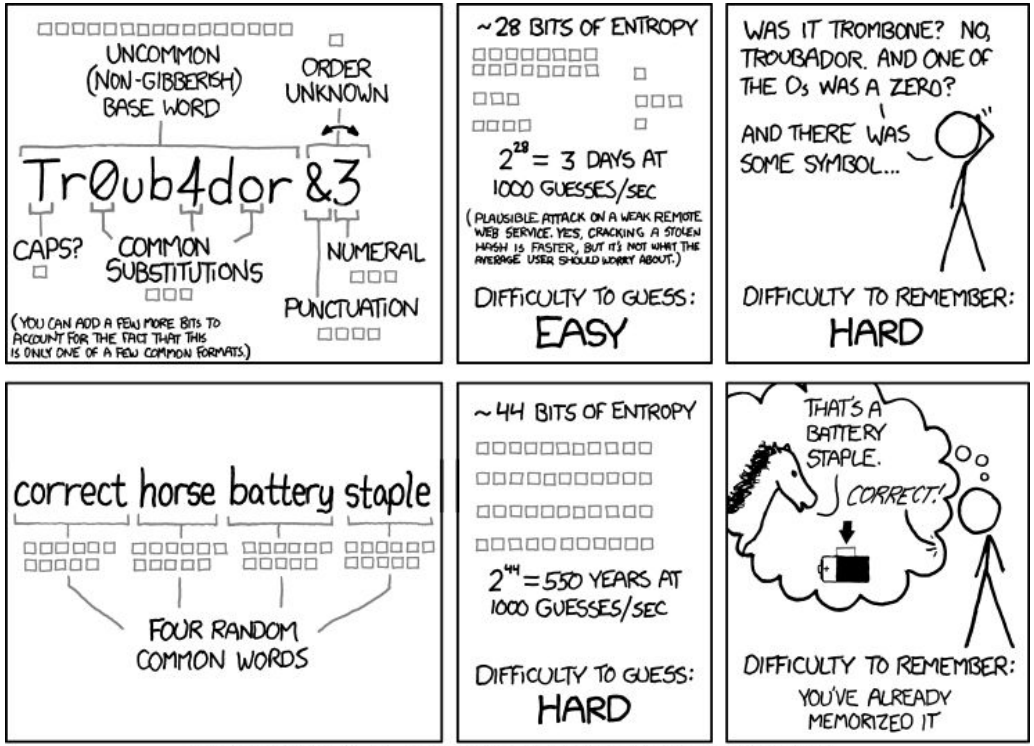
Se considera una contraseña segura si D es mayor o igual a 3 y si L es mayor o igual a 14.

Cazier y Medlin

Figura 2: Clasificación de escala basado en la contraseña del usuario

Level	Description	Criteria
1	Obvious name or number.	1. Password contains all or part of a username, email, name, etc. 2. Or is somehow related to known information, such as a spouse or a child's name. We may not have enough information to always assess this. 3. It is easily guessable by someone that knows a little about the individual, based on publicly available knowledge.
2	Common word or number	1. Password is a repeating number (i.e., 99999) 2. Or it is a common word recognizable as either a name or word likely to be found in a dictionary?
3	Words with number at start or end or non-repeating numbers	1. Example, John52 or 73horses, combining recognizable words or phrases with a number. 2. Or having a nonrepeating number, such as 345820, that does not appear to be a social security number, phone number, or street address. 3. Or adding a compound word, such as lovejohn or toomuchfun. 4. Or unrecognizable alpha characters that did not form a common word or name (i.e., mtlyfil).
4	Unrecognizable alpha and number combinations	1. Has a mixture of unrecognizable alpha characters with numbers thrown in (i.e.jds932). 2. May include compound words with numbers lovejohn99.
5	Special characters	1. Numbers and letters mixed with the inclusion of special characters like #, %, and, @ etc. 2. Spaces and dashes are included in this category.

Randall Munroe's XKCD 936



Recomendaciones

- **No utilizar la misma contraseña en múltiples servicios (Que una se filtre compromete todas tus demás plataformas) .**
- **No escribir tus contraseñas o guardarlas en archivos (Los ataques pueden venir de coempleados o personas cercanas o ya bien si tienen acceso a tus dispositivos podrían encontrar tus contraseñas). Password wallets pueden ser un problema si no son utilizadas correctamente.**
- **Utilizar multiple factor authentication**
- **Eye-of-newt password rules: One time passwords y "Don't be a Moron Rule" (No escoger contraseñas fáciles que un amigo o un familiar puedan adivinar con unos cuantos intentos)**
- **Utilizar como contraseñas una mezcla de palabras y varios caracteres diferentes como se muestra en la figura anterior.**

06

CASO PRÁCTICO

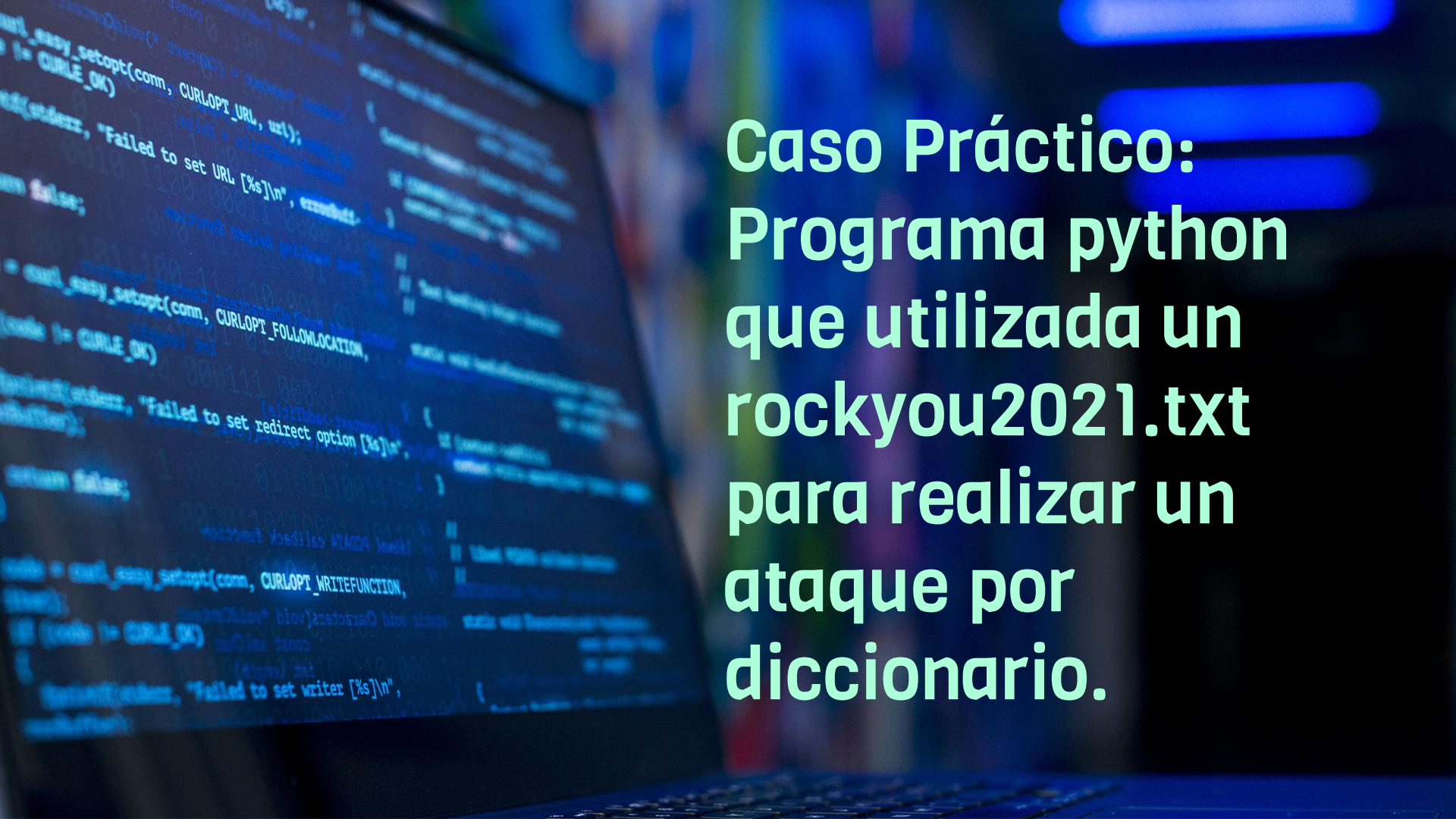
Rock You Filtraciones

En 2009, la red social RockYou sufrió una filtración de contraseñas que afectó a 32 millones de usuarios, especialmente porque RockYou guardaba todos los datos de los usuarios incluidas las contraseñas como texto plano exponiendo toda esta información a los hackers. Fue un total caos ya que no solo tenían políticas malas para el manejo de contraseñas sino también manejaron la situación terriblemente con sus usuarios.

En 2021, en un foro de hackers se filtró el archivo rockyou2021.txt haciendo referencia al incidente anterior. Este archivo de 100GB.

8.4 billones

RockYou2021


Caso Práctico:
Programa python que utilizada un rockyou2021.txt para realizar un ataque por diccionario.

```
#We do the dictionary attack
#We assume in this case we have the plain text password already and not the hashes
i=0
bar = progressbar.ProgressBar(max_value=progressbar.UnknownLength)
BUF_SIZE = 1024
TOTAL_SIZE = 0
tmp_lines = filePasswords.readlines(BUF_SIZE)
TOTAL_SIZE = TOTAL_SIZE + BUF_SIZE
# while TOTAL_SIZE<1000:
while tmp_lines:
    try:
        for line in filePasswords.read(1024):
            passw = line.rstrip()
            if(passw==password):
                found_password_in_dict = found_password_in_dict + 1
                password_found = passw
            time.sleep(0.1)
            i=i+1
            bar.update(i)
        TOTAL_SIZE = TOTAL_SIZE + BUF_SIZE
        tmp_lines = filePasswords.readlines(BUF_SIZE)
    except KeyboardInterrupt:
        break

print("\n")
if(found_password_in_dict>0):
    print("Password has ben found in leaked dictionary. {}".format(found_password_in_dict))
    print("Password Found was {} in {} entries".format(password_found,i))
else:
    print("Password was not found in dict and it will most likely pass a dictionary attack in a total of {} entries checked.".format(i))

end = time.time()
time_in_s = end - start
print("Total time in seconds to check {}".format(time_in_s))
```

```
# m size of possible caracters 94
N = 94
# m size of password
m = len(password)

# Entropy
entropy_of_password = math.log(N,2)*m
print("Entropy of password {}".format(entropy_of_password))
# Nist Guidelines entropy
entropy_nist = 0
entropy_nist = entropy_nist + 6 if m>0 else entropy_nist
entropy_nist= entropy_nist + (m-1)*2 if m>1 and m<=8 else (entropy_nist + 7*2 if m>8 else entropy_nist)
entropy_nist= entropy_nist + (m-8)*1.5 if m>8 else entropy_nist
ntropy_nist= entropy_nist + 3 if any(character.isupper() for character in password) else entropy_nist
entropy_nist= entropy_nist + 3 if not password.isalnum() else entropy_nist
entropy_nist= entropy_nist + 6 if not found_password_in_dict else entropy_nist
print("Entropy Nist of Password {}".format(entropy_nist))

# Calculate Effective L of Password, L is Good if its more than 14
L = m * math.log(N,10)
print("L value of password (safe if more than 14) {}".format(L))
```

¡Gracias por su atención!

¿Dudas?

REFERENCIAS

Cazier, J. A., & Medlin, B. D. (2006). *Password Security: An Empirical Investigation into E-Commerce Passwords and Their Crack Times.* Information Systems Security, 15(6), 45–55. doi:10.1080/10658980601051318

Cheswick, William (2013). *Rethinking passwords.* Communications of the ACM, 56(2), 40–. doi:10.1145/2408776.2408790

Garfinkel, S., & Lipford, H. R. (2014). *Usable Security: History, Themes, and Challenges. Synthesis Lectures on Information Security, Privacy, and Trust*, 5(2), 1–124. doi:10.2200/s00594ed1v01y201408spt011

Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F. (2019). *PassGAN: A Deep Learning Approach for Password Guessing.* In: Deng, R., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds) Applied Cryptography and Network Security. ACNS 2019. Lecture Notes in Computer Science(), vol 11464. Springer, Cham. https://doi.org/10.1007/978-3-030-21568-2_11

Hu, G. (2017). *On password strength: a survey and analysis.* In International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (pp. 165-186). Springer, Cham.

Morris, R., & Thompson, K. (1979). *Password Security: A Case History.* Commun. ACM, 22(11), 594–597. doi:10.1145/359168.359172

Lee, R. (2018). *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing.* doi:10.1007/978-3-030-26428-4

OhMyBahGosh. (2022). RockYou2021.txt. Github Repository. https://github.com/ohmybahgosh/RockYou2021.txt#:~:text=About-,RockYou2021.,compiled%20of%20various%20other%20wordlists.

Orozco, S. (2022). *PasswordCheck.* Github Repository. https://github.com/sorozcov/PASSWORDCHECK

WeLiveSecurity. (2017). *A short history of the computer password.* Recuperado de https://www.welivesecurity.com/2017/05/04/short-history-computer-password/

Yannis, C. (2013). *Modern Password Cracking: A hands-on approach to creating an optimized and versatile attack* (Tesis de Master). University of London.