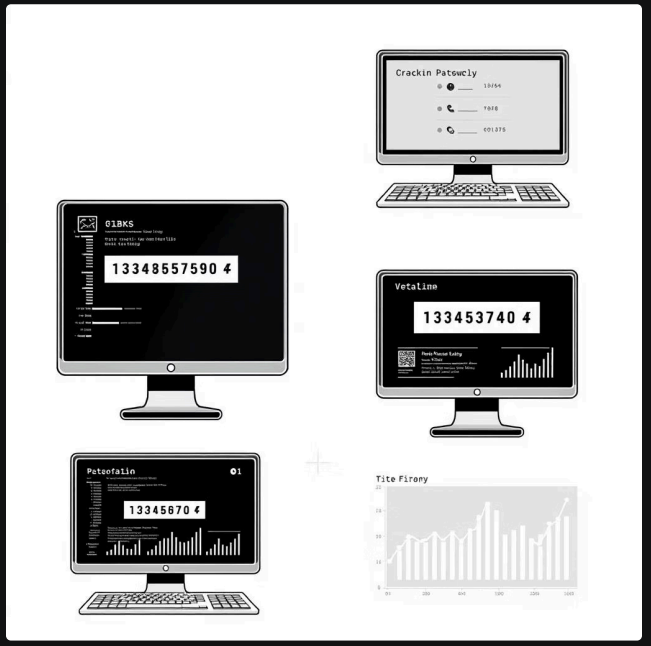
Cracking de Contraseñas: Fuerza Bruta por tiempo

Andrés Santiago Aguirre Macias Codigo:220293594
Eduardo Ramos Ochoa Codigo:304489918

Fuerza Bruta

¿Qué es el Cracking por Fuerza Bruta?

- Utilizada para descifrar contraseñas al probar sistemáticamente cada combinación posible de caracteres hasta dar con la correcta
- Su velocidad depende críticamente de la complejidad de la contraseña.



Los principales factores que afectan el tiempo de cracking son:

Longitud de la contraseña:

Cada caracter adicional aumenta exponencialmente el número de combinaciones.

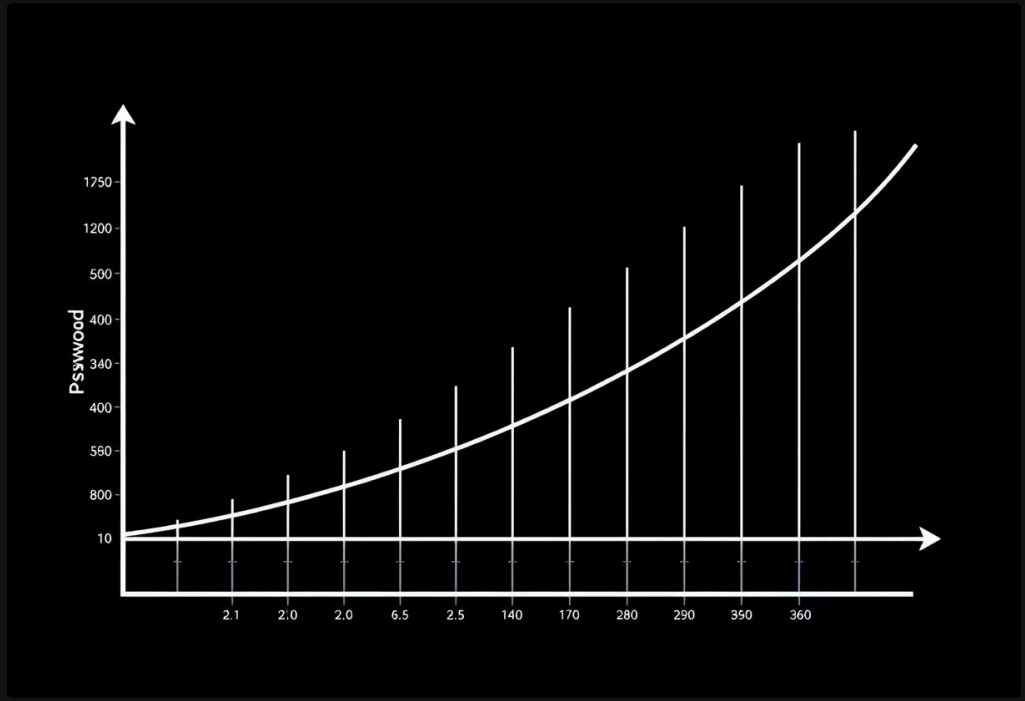
Conjunto de caracteres (keyspace):

El rango de caracteres posibles (minúsculas, mayúsculas, números, símbolos). Un conjunto más grande (por ejemplo, 70 caracteres vs. 26) incrementa drásticamente las combinaciones.

Ejemplos matemáticos de combinaciones:

Ejemplo:

- Para una contraseña de 4 caracteres, solo minúsculas (a-z, 26 opciones): $26^4 = 456,976$ combinaciones.
- Para una contraseña de 4 caracteres, con minúsculas, mayúsculas, números y símbolos (aproximadamente 70 opciones): $70^4 = 24,010,000$ combinaciones.



Ventajas

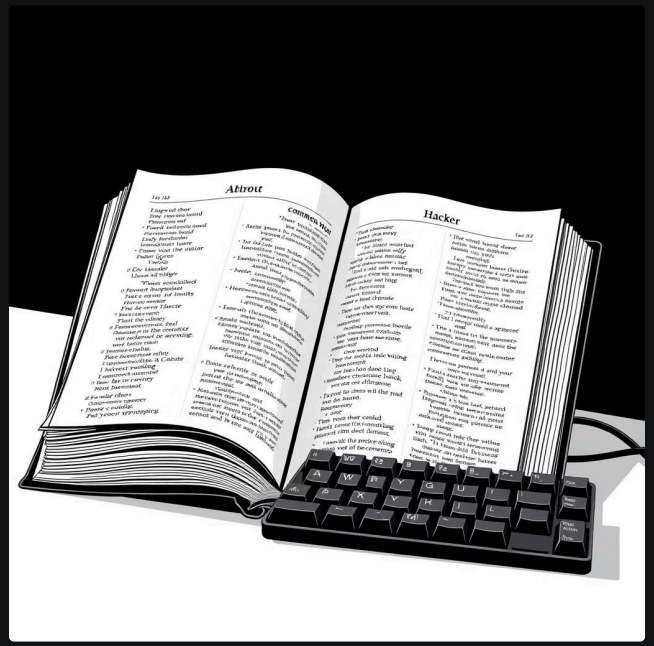
- 🔹 Altas probabilidades de éxito si el tiempo y los recursos son ilimitados.
- 🔹 No requiere conocimiento previo sobre la contraseña.

Desventajas

- 🔹 Puede ser extremadamente lento y costoso en términos de recursos computacionales.
- 🔹 Detección fácil por sistemas de seguridad (bloqueo de cuentas).

Fuerza Bruta usando Diccionarios

- En lugar de probar todas las combinaciones, el atacante utiliza una lista predefinida de palabras o frases comunes (un "diccionario").



Proceso del Ataque por Diccionario

Preparar Diccionario

Se crea un archivo que contiene una lista de contraseñas comunes y frases conocidas (ej. "rockyou.txt").

Probar Contraseñas

El programa compara cada contraseña del archivo de diccionario con el hash de la contraseña de la víctima.

Finalizar Cracking

Si se encuentra una coincidencia, el ataque por diccionario es exitoso y el proceso de cracking termina.

Ventajas:

- 🔹 Mucho más rápido que probar todas las combinaciones
- 🔹 Ideal cuando el usuario usa contraseñas comunes como "123456", "qwerty" o "password"

Desventajas:

- 🔹 No garantiza encontrar la contraseña si no está en el diccionario
- 🔹 Solo funciona si la contraseña está en el diccionario o es una variación simple

Aplicaciones en la Vida Real

- **Pruebas de seguridad (Pentesting) en empresas**

Equipos de seguridad usan ataques por diccionario y fuerza bruta controlados para evaluar la robustez de contraseñas de cuentas internas o sistemas de prueba.

- **Recuperación de contraseñas y acceso a datos cifrados (con permiso legal)**

Herramientas legítimas y administradores de sistemas usan ataques controlados para recuperar contraseñas de archivos o sistemas cuando el dueño olvida su clave (por ejemplo, discos cifrados, archivos ZIP, bases de datos locales).

¿Cómo Funcionan Estos Ataques?

01

Obtener Objetivo

El atacante identifica la contraseña en texto plano o su hash.

02

Generar Candidatos

Se crean posibles contraseñas: por diccionario (lista de palabras) o fuerza bruta (todas las combinaciones hasta una longitud L).

03

Procesar y Comparar

Cada candidata es (opcionalmente) hasheada y comparada con el objetivo.

04

Resultado

Si hay coincidencia, el ataque es exitoso; si no, el proceso continúa o se detiene si se alcanzó el límite.

Complejidad Computacional por fuerza bruta (Temporal y Espacial)

Complejidad Temporal

La complejidad temporal es $O(A^L)$ donde A es el tamaño del alfabeto y L es la longitud de la contraseña.

- **Ejemplos reales:**
- $10^6 = 1,000,000$ combinaciones (contraseñas de 6 dígitos)
- $26^6 = 308,915,776$ combinaciones (solo minúsculas, L=6)
- $62^8 \approx 2.18 \times 10^{14}$ combinaciones (alfabeto alfanumérico, L=8)
- **Interpretación:** Si un atacante prueba 1 millón de candidatos por segundo, descifrar 62^8 contraseñas tomaría aproximadamente 2.18×10^8 segundos, que equivale a ≈ 6.9 años (lo que no es práctico).

Fuerza Bruta

Requiere $O(1)$ memoria adicional (se genera y prueba sobre la marcha).

Complejidad Computacional por diccionario(Temporal y Espacial)

Complejidad Temporal

Ataque por Diccionario

La complejidad temporal es $O(N)$ donde N es el tamaño del diccionario.

- Depende directamente de la calidad del diccionario.
- Mucho más rápido si la contraseña es común y se encuentra en la lista.

Complejidad Espacial (Memoria)

Ataque por Diccionario

Requiere $O(N)$ si el diccionario se carga completamente en memoria, o $O(1)$ si se procesa mediante *streaming*.

Importancia del Algoritmo (Eficiencia y Aplicabilidad)

Eficiencia:

La fuerza bruta no es eficiente comparada con otras técnicas: su tiempo crece exponencialmente con la longitud de la contraseña.

Sin embargo, es el único método que garantiza encontrar la solución si se tiene suficiente tiempo y recursos.

Es una línea base en seguridad informática: cualquier sistema debe ser resistente incluso a este ataque.

Aplicabilidad:

Útil en entornos controlados (clases, auditorías, pruebas de seguridad, CTFs) para mostrar los límites de la seguridad de contraseñas.

En la práctica se usa como parte de herramientas de recuperación de contraseñas (cuando no existen otras vías) o en pruebas de penetración autorizadas.

Para los defensores, entender la fuerza bruta permite diseñar políticas seguras (contraseñas largas, MFA, hashing robusto).

Rainbow tables:

- **Ventaja:** extremadamente rápidas para recuperar contraseñas **si** se tiene la tabla precomputada.
- **Desventaja:** requieren **mucho espacio** (terabytes) y **no funcionan con salt**.

Ataques por diccionario + reglas (hybrid / rule-based):

- Más eficaces en la práctica (los usuarios usan variaciones previsibles).
- Menos costosos que la fuerza bruta pura.

Técnicas avanzadas:

Markov models, PCFG, GPU/FPGAs para aceleración masiva.

- Permiten priorizar candidatos probables y reducir tiempo efectivo.

Conclusión comparativa: fuerza bruta = garantía teórica, pero **poco práctica** en contraseñas modernas; diccionario/heurísticas/rainbow (con permiso) = métodos más eficientes en la práctica.

Limitaciones y posibles mejoras

Limitaciones del enfoque presentado

- Exponencialmente lento para contraseñas largas y alfabetos grandes.
- Detectable por mecanismos de seguridad (rate-limiting, lockouts).
- Diccionarios dependen de la calidad de la lista; fallan contra contraseñas únicas.

Mejoras prácticas / alternativas

- Ataques híbridos: combinación de diccionario + reglas (mucho más eficaces).
- Uso de GPU/cluster para paralelizar y reducir tiempos (GPU acelera hashing masivo).
- Preprocesamiento inteligente: modelos probabilísticos (PCFG, Markov) para priorizar candidatos.
- Para defensores: forzar hashing lento (bcrypt/Argon2), usar salt, rate limiting, MFA, gestores de contraseñas.

Conclusión

Como se pudo notar a lo largo de la presente exposición, el algoritmo de ataque por fuerza bruta y por diccionario, se tornan en herramientas o métodos de cracking simples pero efectivos, si bien es verdad que, adolecen de ciertos puntos negativos como lo es la complejidad temporal que aumenta conforme al número de combinaciones posibles y/o elementos que contenga el diccionario que se le provea.

Por otro lado, no se debe pasar por alto, el hecho de que, la efectividad de esta herramienta, se puede ver afectada por factores como lo son; la limitación de intentos para introducir la contraseña correcta, así como el uso de contraseñas largas y bien diseñadas y, obviamente un correcto empleo de técnicas de seguridad cibernética.

Recuerda

La seguridad de tus cuentas depende de contraseñas robustas y prácticas de seguridad.

¡Sé proactivo en tu protección digital!