



# Descifrado de Contraseñas (Password Cracking)

## Requisitos Previos

Puedes utilizar cualquier distribución de Linux, así como sistemas operativos Windows o macOS, e instalar las herramientas manualmente cuando sea posible. No obstante, distribuciones como Kali Linux o Parrot

OS ya vienen precargadas con la mayoría de las herramientas necesarias.

**Nota:** La instalación y configuración de estas distribuciones está **fuera del alcance de esta guía**, pero puedes referirte a sus documentos oficiales para más detalles:

- Kali Linux Documentation - [www.kali.org/docs](http://www.kali.org/docs)
- Parrot OS Documentation - [parrotsec.org/docs](http://parrotsec.org/docs)

## El Uso de Contraseñas y sus Vulnerabilidades

El método de seguridad más común y ampliamente utilizado sigue siendo el uso de contraseñas. En términos simples, una contraseña es una combinación de letras, números y símbolos diseñada para verificar el acceso a sistemas, archivos u otros recursos.

El 73% de las contraseñas más utilizadas pueden ser descifradas en menos de un segundo, y el 85% de las personas reutilizan las mismas contraseñas en múltiples servicios, lo que aumenta el riesgo de que, si una contraseña se ve comprometida, los demás accesos también lo estén.

Los hábitos comunes en la creación de contraseñas incluyen el uso de información personal (como nombres, fechas, mascotas o números de teléfono), contraseñas de 8 caracteres o menos, y palabras fácilmente encontradas en diccionarios, lo que las hace aún más vulnerables.

En caso de una violación de seguridad, cuando un atacante obtiene acceso no autorizado a una base de datos que contiene contraseñas cifradas, o intercepta hashes de autenticación de redes como WPA2, NTLM o Kerberos, o archivos protegidos por contraseñas como ZIP, PDF o Word, entre otros, el siguiente paso habitual es intentar descifrar esos hashes.

## Funcionamiento de una Contraseña en un Sistema Informático

### 1. Creación

El usuario elige una contraseña que puede incluir letras, números y símbolos. El sistema **puede** establecer políticas de seguridad, como una longitud mínima, el uso obligatorio de caracteres especiales, mayúsculas, minúsculas, o evitar el uso de contraseñas comunes y fáciles de adivinar.

### 2. Hashing

El sistema convierte la contraseña en un valor único y de longitud fija mediante un proceso criptográfico llamado **hashing**. Este proceso es unidireccional, lo que significa que no se puede revertir para obtener la contraseña original.

### 3. Salting

Para aumentar la seguridad, en algunos algoritmos se añade un valor aleatorio, conocido como **sal**, a la contraseña antes de aplicar el hash. Esto asegura que incluso si dos usuarios tienen la misma contraseña, los valores hash generados serán diferentes.

### 4. Almacenamiento

El sistema guarda el hash (y el sal si se utiliza) nunca la contraseña original.

## 5. Verificación de la Contraseña

El sistema compara el hash de la contraseña ingresada (y sal) con el almacenado. Si coinciden, el acceso es permitido; si no, se deniega.

## La Importancia del Poder Computacional

El poder computacional juega un papel crucial al intentar descifrar el hash de una contraseña.

## Longitud y Complejidad

A mayor longitud y complejidad, más combinaciones posibles. Por ejemplo, una contraseña de 8 caracteres con letras y números tiene 218 billones de combinaciones. A medida que aumentan la longitud y la complejidad, las combinaciones crecen exponencialmente.

## Algoritmo de Hashing

Algunos algoritmos, como MD5 y SHA-1, son rápidos y permiten realizar muchas operaciones por segundo, mientras que otros como bcrypt, scrypt o Argon2 son más lentos, lo que dificulta los ataques de fuerza bruta debido al coste computacional adicional.

## Recursos Computacionales

El poder de cómputo de ataque puede provenir de las CPUs o GPUs de un equipo. Las GPUs superan a las CPUs en velocidad para cálculos paralelos, procesando millones de hashes por segundo. Esto las hace ideales para descifrar contraseñas. Además, la distribución del cálculo en redes de computadoras o botnets reduce significativamente el tiempo necesario para descifrar contraseñas complejas.

En resumen, descifrar contraseñas depende de su longitud, complejidad, el algoritmo de hashing y el hardware utilizado para descifrarla.

## ATAQUES COMUNES PARA DESCIFRAR UNA CONTRASEÑA

### Fuerza Bruta

Es una técnica en la que se prueban **todas las combinaciones posibles** hasta encontrar la correcta. **Garantizan** el éxito si se tiene suficiente tiempo y poder computacional. Sin embargo, dependiendo de la complejidad de la contraseña, puede ser muy lento, por lo que se utilizan principalmente para contraseñas cortas o en situaciones específicas.

## Diccionario

Es uno de los métodos más comunes, junto con el ataque híbrido, ya que utiliza una lista de palabras comunes extraídas de un diccionario (archivo de palabras). Es más rápido que la fuerza bruta, ya que se enfoca en las combinaciones más probables.

## Híbridos / Mascara

Uno de mis preferidos ya que combinan diccionarios y modificaciones de esos términos (como agregar números o símbolos), lo que los hace muy efectivos para contraseñas algo complejas, pero todavía no extremadamente difíciles de descifrar.

## Rainbow Table (NTLM, MD5, SHA1, etc.)

Las Rainbow Tables se utilizan en ataques a hashes que no utilizan salting. Su efectividad depende de las contraseñas precomputadas en las tablas. Puedes descargar tablas desde RainbowCrack, generarlas con rtgen o usar servicios como CrackStation.

# HERRAMIENTAS PARA DESCIFRAR CONTRASEÑAS

## John The Ripper - <https://www.openwall.com/john/>

No podía iniciar esta guía sin hablar de nuestro querido “John The Ripper”. La última versión de la comunidad “Jumbo” de John the Ripper soporta cientos de tipos de hashes y cifrados, incluyendo contraseñas de Unix (Linux, macOS, Windows), aplicaciones web (WordPress), software de colaboración (Notes/Domino), servidores de bases de datos (SQL, LDAP), tráfico de red (WiFi WPA-PSK), claves privadas cifradas (SSH, criptomonedas), archivos (ZIP, RAR, PDF, Microsoft Office) y más.

**Modo Incremental (Fuerza Bruta) empleando todas sus configuraciones** El modo incremental predeterminado de John realiza una búsqueda exhaustiva de todas las combinaciones posibles, desde las más simples hasta las más complejas. Es potente, pero puede ser muy lento, especialmente con contraseñas largas o complejas.

```
john --incremental=all <hash o archivo>
```

```
echo -e "[Incremental:Alphanum]\nMinLen = 8\nMaxLen = 8\nCharset =
```

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ" >>
```

```
john.conf
```

```
john --incremental=alphanum <hash o archivo>
```

Modo incremental (Fuerza Bruta) empleando una configuración específica

Modo diccionario

```
john --wordlist=<diccionario> <hash o archivo>
```

```
john --wordlist=<diccionario> --format=<formato> <hash o archivo>
```

## Generación de diccionarios empleando reglas

También se pueden generar diccionarios personalizados mutando el archivo de diccionario original como por ejemplo agregarle una exclamación “!” al final de cada palabra o permutar la letra “a” por “@”.

**john --wordlist=<diccionario> --rules:[regla] > [archivo de destino]**

Si desean más información de cómo crear sus propios diccionarios con John pueden ver [este artículo](#) de Rapid7.

Además, incluye herramientas que permiten extraer hashes de diversas fuentes, como archivos PDF, claves SSH, archivos ZIP, entre otros, para luego proceder con su descifrado.

Herramienta	Descripción
ssh2john	Extrae hash de claves privadas SSH para John
zip2john	Extrae hash de archivos ZIP para John
rar2john	Extrae hash de archivos RAR para John
hccap2john	Extrae hash de capturas de handshake WPA/WPA2 para John
wpa2john	Extrae hash de handshakes WPA/WPA2 para John
office2john	Extrae hash de documentos MS Office para John
pdf2john	Extrae hash de documentos PDF para John
keepass2john	Extrae hash de bases de datos KeePass para John
mscash2john	Extrae hash de hashes MS Cash para John
truecrypt_volume2john	Extrae hash de volúmenes TrueCrypt para John
keychain2john	Extrae hash de archivos de llavero de OS X para John
pfx2john	Extrae hash de archivos PKCS#12 para John
vncpcap2john	Extrae hash de archivos PCAP de VNC para John
putty2john	Extrae hash de claves privadas de PuTTY para John

## JOHN THE RIPPER CHEATSHEET

### url

<https://www.openwall.com/john/>

### Installation

```
sudo apt-get install john
```

### The location of the password file

```
/root/.john/john.pot ~/john/john.pot
```

### Performance test

```
john --test
```

### **Create a session**

john hash.txt --session=session-name

### **Restoring the session**

john --restore=session-name

### **List of password formats**

john --list=formats

### **Rules list**

john --list=rules

### **Status display**

john --status

### **Unshadow**

unshadow passwd.txt shadow.txt > unshadowed.txt

### **Create a word list**

john --wordlist=list.txt --stdout --external:[filter] > output.txt

### **ZIP file conversion for John the Ripper**

zip2john file.zip > ziphash.txt

### **RAR file conversion for John the Ripper**

rar2john file.zip > rarhash.txt

### **Default attack**

john hash.txt

### **Incremental Attack**

john --incremental hash.txt

### **Mask Attack**

john format=hashtype hash.txt mask?a?a?a?a

### **Cracking MD5**

john --format=raw-md5 --wordlist=rockyou.txt hash1.txt

### **Breaking SHA256**

john --format=raw-sha256 --wordlist=rockyou.txt hash3.txt

## Breaking Whirlpool

```
john --format=whirlpool --wordlist=rockyou.txt hash4.txt
```

HASHCAT - <https://github.com/hashcat/hashcat>

Hashcat está diseñado desde el principio para aprovechar al máximo el poder de las GPU (Unidades de Procesamiento Gráfico), lo que lo convierte en una herramienta extremadamente rápida para el cracking de contraseñas, especialmente cuando se utilizan GPUs modernas. Soporta una gran variedad de algoritmos de hash (MD5, SHA-1, bcrypt, WPA, etc.), lo que lo hace muy versátil. La compatibilidad con más de 300 tipos de hashes lo convierte en una herramienta preferida en muchas situaciones.

Además, cuenta con soporte para modo distribuido, lo que significa que puedes distribuir el trabajo entre varias máquinas o en una red de computadoras.

Con un RIG “casero” con 4 GPUs NVIDIA RTX 4090 y un costo aproximado de 12,000€, puedes esperar el siguiente rendimiento aproximado:

- Hashes MD5: 4 GPUs x 155 GH/s = 620 GH/s
- Hashes NTLM: 4 GPUs x 285 GH/s = 1.140 GH/s

## ¡Mas de 1 billón de contraseñas por segundo en NTLM!

También existe la posibilidad de alquilar servidores en línea con GPU intensivo como puede ser **vast.ai** por un precio “razonable”.

## Detección Automática del Tipo de Hash

Hashcat no cuenta con una función automática para detectar tipos de hashes, por lo que es necesario especificarlo manualmente. Para identificar el tipo de hash, puedes utilizar herramientas como hashid.

```
hashid -m <hash a analizar>
```

## 1. Ataque de Fuerza Bruta (Brute-Force) con mascara

Tabla de Máscaras para Hashcat	
Símbolo	Significado
?l	Letra minúscula
?u	Letra mayúscula
?d	Dígito (número)
?s	Símbolo especial
?a	Cualquier carácter (alfanumérico + símbolos)
?b	Cualquier byte (carácter binario)
?h	Hexadecimal (0-9, a-f)
?H	Caracteres hexadecimales (mayúsculas)

**hashcat -m <tipo de hash> -a 3 <archivo hash> ?l?l?l?l?d?d?d?d**

La máscara ?l?l?l?l?d?d?d?d generará combinaciones de contraseñas de 8 caracteres, en la forma de 4 letras minúsculas seguidas de 4 dígitos numéricos (por ejemplo: abcd1234).

## 2. Ataque de Diccionario

**hashcat -m < tipo de hash > -a 0 < archivo hash > <diccionario>**

## 3. Ataque Combinado

En este tipo de ataque, Hashcat combina palabras de dos diccionarios diferentes. Cada palabra de un diccionario se combina con cada palabra del otro.

**hashcat -m < tipo de hash > -a 1 < archivo hash > <diccionario1><diccionario2>**

Esto probará todas las combinaciones posibles de las palabras de diccionario1 y diccionario2. Si diccionario1 tiene las palabras "apple" y "banana", y diccionario2 tiene "123" y "456", Hashcat probará combinaciones como:

- apple123
- apple456
- banana123
- banana456

## 4. Ataque Híbrido (Diccionario + Prefijos/Sufijos)

En este tipo de ataque, puedes combinar un diccionario con un prefijo o sufijo (como números, símbolos, etc.) que se añaden o anteponen a las palabras del diccionario.

**hashcat -m < tipo de hash > -a 6 < archivo hash > <diccionario> ?d?d?d hashcat -m < tipo de hash > -a 7 < archivo hash > ?d?d?d <diccionario>**

## 5. Ataque con el uso de Reglas

Las reglas se emplean para modificar las palabras del diccionario antes de hacer el hash. Las reglas pueden incluir convertir todas las letras a mayúsculas, agregar números al final, etc.

**hashcat -m < tipo de hash > -a 0 < archivo hash > <diccionario> -r  
<archivo\_regla>**

El archivo de reglas contiene un conjunto de instrucciones que dictan cómo modificar las palabras del diccionario antes de hacer el hash.

### Ejemplo de Reglas:

Regla	Función	Entrada	Salida
:	No hacer nada.	p@ssW0rd	p@ssW0rd
l	Convierte todas las letras a minúsculas	p@ssW0rd	p@ssw0rd
u	Convierte todas las letras a mayúsculas	p@ssW0rd	P@SSW0RD
c	Convierte la primera letra a mayúscula y las demás a minúsculas	p@ssW0rd	P@ssw0rd
ss\$	Reemplaza todas las instancias de s con \$	p@ssW0rd	p@\$sW0rd
\$!	Añade el carácter de exclamación al final de la palabra	p@ssW0rd	p@ssW0rd!

### Ejemplo de Archivo de Regla:

l # Convierte todas las letras a minúsculas u # Convierte todas las letras a mayúsculas \$!  
# Añade "!" al final de la palabra sa@ # Reemplaza el carácter "a" por "@"

## 6. Ataque de Rainbow Tables

Utiliza una tabla precalculada de hashes (Rainbow Tables) para encontrar la contraseña correspondiente a un hash específico. Las tablas almacenan los resultados de hashes previamente calculados, lo que hace que este tipo de ataque sea rápido. NTLM y MD5 son comunes para este tipo de ataque.

**hashcat -m < tipo de hash > -a 7 < archivo hash > <archivo rainbow>**

## 7. Ataque de Contraseñas de Archivos

Hashcat también es capaz de descifrar contraseñas de archivos cifrados (por ejemplo, archivos ZIP, RAR, PDF, etc.).

**hashcat -m 13600 -a 0 < archivo hash > <diccionario>**

-m 13600 para archivos PDF

Para mayor información puedes acceder a su Wiki:

<https://hashcat.net/wiki/>

## Generadores de Diccionarios

Herramienta	Descripción	Enlace
<b>Crunch</b>	Genera diccionarios personalizados con patrones específicos como combinaciones de caracteres y longitudes.	Crunch en GitHub <sup>[1]</sup>
<b>Cewl</b>	Extrae palabras de sitios web y las organiza en un diccionario. Ideal para personalización basada en contexto.	Cewl en GitHub <sup>[3]</sup>
<b>Username Anarchy</b>	Potente herramienta utilizada para generar diccionarios de nombres de usuario basados en patrones específicos	Anarchy en GitHub <sup>[5]</sup>
<b>Pydictor</b>	Generador de diccionarios avanzado basado en Python que admite múltiples patrones y configuraciones.	Pydictor en GitHub <sup>[7]</sup>
<b>CUPP (Common User Passwords Profiler)</b>	Genera listas basadas en información personal (nombre, cumpleaños, etc.) para ataques dirigidos.	CUPP en GitHub <sup>[9]</sup>

[1] <https://github.com/crunchsec/crunch>

[2] <https://github.com/crunchsec/crunch>

[3] <https://github.com/digininja/CeWL>

[4] <https://github.com/digininja/CeWL>

[5] <https://github.com/urbanadventurer/username-anarchy>

[6] <https://github.com/urbanadventurer/username-anarchy>

[7] <https://github.com/LandGrey/pydictor>

[8] <https://github.com/LandGrey/pydictor>

[9] <https://github.com/Mebus/cupp>

[10] <https://github.com/Mebus/cupp>

## Descarga de diccionarios

Nombre del Repositorio	Descripción	Enlace
<b>SecLists</b>	Repositorio con diccionarios de uso general y contraseñas filtradas.	SecLists en GitHub <sup>[1]</sup>
<b>BreachCompilation</b>	Colección de contraseñas filtradas de múltiples violaciones de datos.	BreachCompilation en <sup>[3]</sup> <sup>[4]</sup> GitHub <sup>[5]</sup>

[1] <https://github.com/danielmiessler/SecLists/tree/master/Passwords>

[2] <https://github.com/danielmiessler/SecLists/tree/master/Passwords>

[3] <https://gist.github.com/spacepatcher/8f94289236612a40ed93efc5645c0ccd>

[4] <https://gist.github.com/spacepatcher/8f94289236612a40ed93efc5645c0ccd>

[5] <https://gist.github.com/spacepatcher/8f94289236612a40ed93efc5645c0ccd>

[6] <https://gist.github.com/spacepatcher/8f94289236612a40ed93efc5645c0ccd>