

Password Cracking

correct horse battery staple





Hello!

I am Javier Junquera

Investigador en ciberseguridad, y profesor
asociado en la Universidad de Alcalá

You can find me at  javier.junquera@uah.es
 [@junquera](https://twitter.com/@junquera)



Cybersecurity Group UAH



Hello!

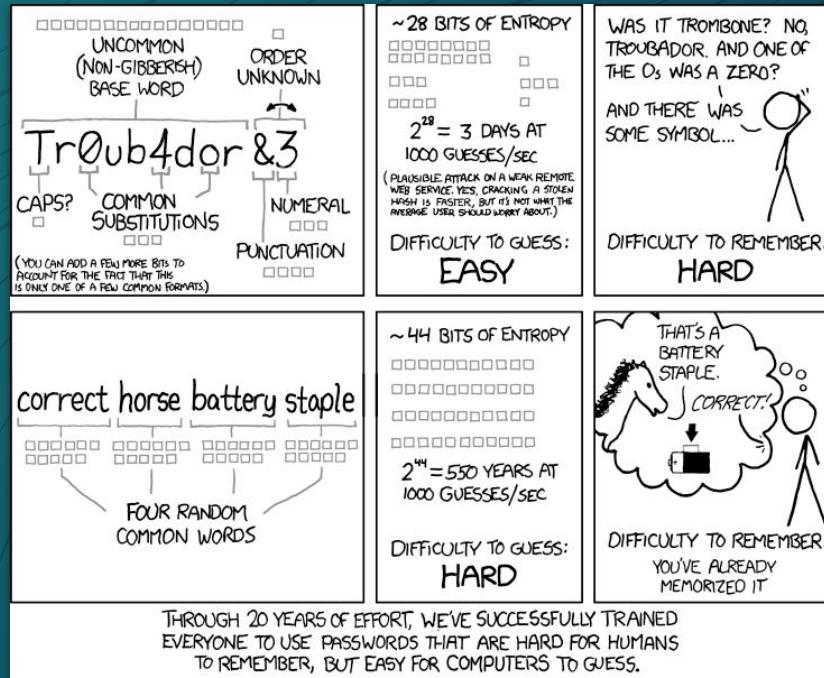
I am **Kevin Van Liebergen**

Becario de investigación en cátedra
ISDEFE

You can find me at  @KevinLiebergen



Contraseñas





Almacenamiento de contraseñas

- Codificar no es cifrar
 - ◆ Reversible sin dificultad
- Cifrar no es hashear
 - ◆ Reversible
- Hashear es hashear, pero no lo es todo



Codificaciones ¿Son seguras?

No existe clave

- ❖ Tipos

- Base64
- Base32
- ...

Texto	u						a						h									
ASCII	117						97						104									
Bits	0	1	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	0	0	0
Índice	29						22						5						40			
Base64 4	d						W						F						o			



Codificaciones. Usos

- Parámetro HTTP en los formularios
- Datos binarios en URL

```
$ printf "ciberseg" | base64
```

```
Y2liZXJzZWc=
```

```
$ printf "Y2liZXJzZWc=" | base64 -d  
ciberseg
```



Almacenamiento de contraseñas

- Las contraseñas se deben almacenar protegidas
 - ◆ En claro: No están protegidas
 - ◆ En base64: No están protegidas
 - ◆ En base32: Tampoco



Hashing: ¿Qué es un hash?

Es una función matemática que comprime la información de forma rápida para que sea identificable.

- ❖ Siempre del mismo tamaño
- ❖ No es reversible
- ❖ Los mismos datos dan siempre el mismo hash



Checksum

$$\begin{array}{r} 0000 \\ 0101 \\ 1111 \\ + \quad 0010 \\ \hline 10110 \end{array} \quad H(0000010111110010) = 0110$$




Checksum

$$\begin{array}{r} 0000 \\ 0101 \\ 1111 \\ + \quad 0010 \\ \hline 10110 \end{array}$$





Hashing: ¿Qué es un hash?

Es una función matemática que comprime la información de forma rápida para que sea identificable.

- ❖ Siempre del mismo tamaño
- ❖ No es reversible
- ❖ Los mismos datos dan siempre el mismo hash



¿Qué es un hash criptográfico?

Es una función matemática que comprime la información de forma rápida para que sea identificable.

- ❖ Siempre del mismo tamaño
- ❖ No es reversible
- ❖ Los mismos datos dan siempre el mismo hash
- ❖ **Dos datos distintos dan siempre dos hashes diferentes**



Hash criptográfico

1. $H(A) = H(A)$
2. $H(A) \neq H(B)$
3. $|H(A) - H(A + \epsilon)| > \delta$
4. $H^{-1}(H(A)) = \emptyset$



Hashes criptográficos

- ❖ MD5
- ❖ SHA1
- ❖ SHA256
- ❖ SHA-3 (Keccak)



Hashes criptográficos

```
$ printf 'jajaja' | openssl dgst -md5
```

```
(stdin)= 655faa8ba799a3a1ae309c2b40d142fc
```

```
$ printf 'jajaja' | openssl dgst -sha1
```

```
(stdin)= 4598cf1cfe70d29e5c9d77b51002f272790ba66
```

```
$ printf 'jajaja' | openssl dgst -sha256
```

```
(stdin)= 0a887c3cb3de97b3968be5c7fb81ed9e452928db71d66620b7766bf7c5413878
```

```
$ printf 'jajaja' | openssl dgst -sha3-512
```

```
(stdin)=
```

```
c88fbe79ca2e809915a6ab2e8fd8461c15180ac28cd7b4dab0fbca9f8812169089305890a6e468  
bcde9f08e4cc3eaa00775ab9c9ec28c8cfb54d4de9493676fd
```



Hashes

- Las contraseñas se deben almacenar protegidas
- ¿Están protegidas con MD5, SHA1, SHA256... ?
 - ◆ No lo suficiente



Hashes

Pongamos que tenemos la siguiente contraseña:

PasswOrd!

- a-z → 26 caracteres
- A-Z → 26 caracteres
- Caracteres especiales → 32 caracteres
- Total: 84 caracteres; Tamaño: 9;
- Coste medio: $(84^9)/2 \sim 104.107.874.265.464.832$

Hashes

- Coste medio: $(84^9)/2 \sim 104.107.874.265.464.832$
- Si tenemos 240GH/s
 - ◆ 240.000.000H/s
 - ◆ 13 años en obtenerlo

1. password → Mayus/Minus → $2^8=256$
 2. Password → 1337 → $2^4 \times 256 = 4096$
 3. PasswOrd → Append/Prepend →
 $2 \times 32 \times 4096 = 262144$
- 4. Password!**

$262.144 / 240.000.000 \sim \textbf{0,001 segundos}$

¡PERO!

Hashes

```
Tree: 6fae58fa9b - SecLists / Passwords / Default-Credentials / windows-betterdefaultpasslist.txt Find file Copy path  
govolution Update windows-betterdefaultpasslist.txt 2942b4d on 10 Oct 2019  
2 contributors  
27 lines (27 sloc) | 461 Bytes Raw Blame History  
1 Administrator:FELDTECH  
2 secure:SecurityMaster08  
3 admin:trinity  
4 administrator:Wyse#123  
5 user:Wyse#123  
6 admin:admin  
7 Administrator:Administrator  
11 nmt:1234  
12 admin:password  
13 IEUser:Passw0rd!  
14 openhabian:openhabian  
15 vagrant:vagrant  
16 admin:password  
17 IEUser:Passw0rd!  
18 openhabian:openhabian  
19 vagrant:vagrant  
20 Administrator:vagrant  
21 john:Password123!
```

- Collection#1 → 21.000.000
 - En GPU
 - ◆ 21.000.000/240.000.000
 - 0,0875 s
 - En una CPU
 - ◆ 21.000.000/200.000
 - 87,5 s
 - En cada entrada i del diccionario
- MD5(diccionario[i]) ==
MD5(PasswOrd!)**
- Hay diccionarios de hashes (RainbowTables)
- diccionario[i] ==
MD5(PasswOrd!)**

Rainbow Tables

Tilix: Predeterminado

```
1:junquera@opa:~$  
import time  
import hashlib  
  
k = "Passw0rd!"  
hk = hashlib.md5(k.encode()).hexdigest()  
hk2 = hashlib.md5(k.encode()).hexdigest()  
  
print("Pass", k)  
print("Hash", hk)  
  
def time_dict():  
    t0 = time.time()  
    hk_aux = hashlib.md5(k.encode()).hexdigest()  
  
    if hk_aux == hk:  
        return time.time() - t0  
  
def time_rt():  
    hk_aux = hk  
    t0 = time.time()  
  
    if hk_aux == hk2:  
        return time.time() - t0  
  
print("Dict", time_dict())  
print("RainbowT", time_rt())  
~
```

times.py 1,1 Todo

Tilix: Predeterminado

```
1:junquera@opa:~$ python3 times.py  
Pass Passw0rd!  
Hash 47b7fb65fa83ac9a71dcb0f6296bb6e  
Dict 1.6689300537109375e-06  
RainbowT 2.384185791015625e-07  
junquera@opa:~$
```



Hashes

- Primer paso para protegerlas más:
 - ◆ +SALT (no valen RainbowTables)
 - ◆ Passw0rd! + Número del 1 al 1024 →
 - **MD5(Passw0rd! + Número del 1 al 1024) + Número del 1 al 1024**
- **Coste con diccionario:** Tamaño del diccionario



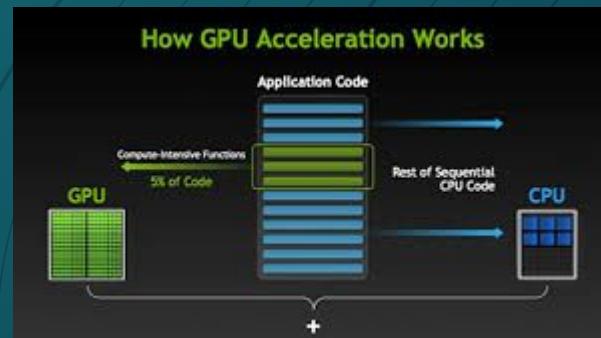
Hashes

- Primer paso para protegerlas más:
 - ◆ +PEPPER
 - ◆ Passw0rd! + Número del 1 al 1024 →
 - **MD5>Password! + Número del 1 al 1024)**
- **Coste con diccionario:** Tamaño del diccionario * 1024

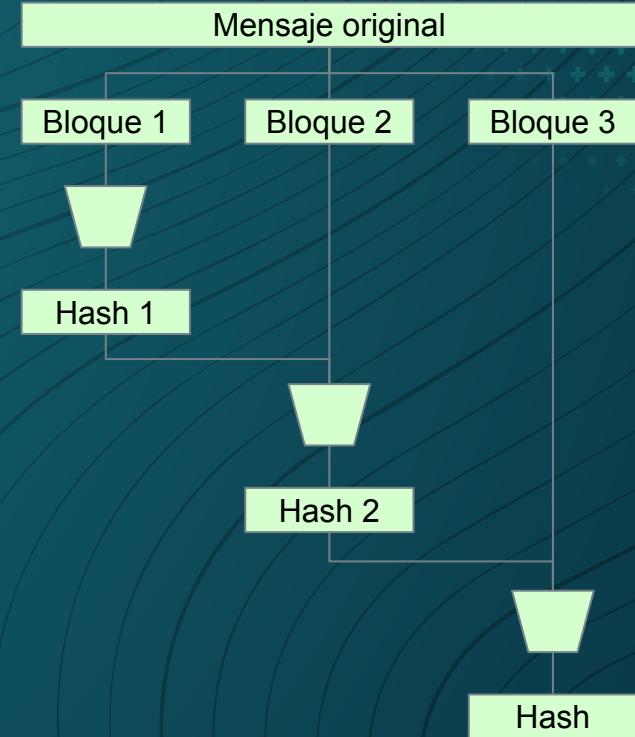
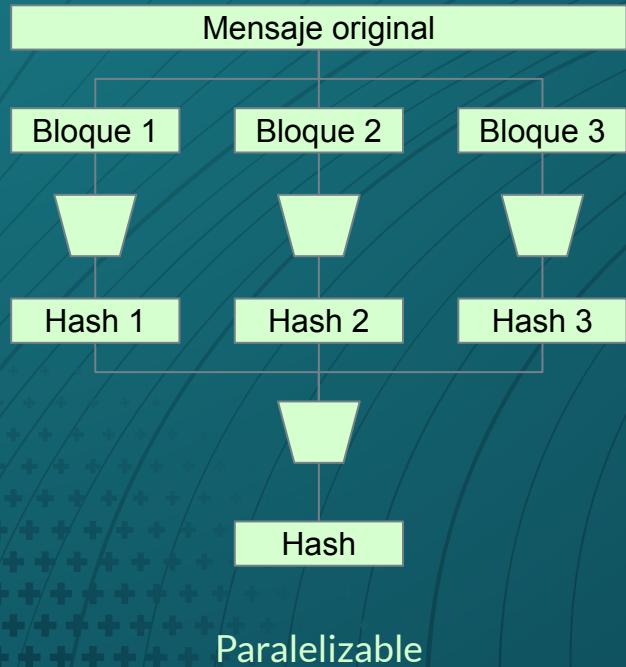


Hashes

- Las contraseñas se deben almacenar protegidas
- ¿Están protegidas con MD5, SHA1, SHA256... ? + (SALT | PEPPER)
- ◆ No lo suficiente → ¡Paralelizable!



Hashing



Paralelizable

Completamente paralelizable



Hashes

- Las contraseñas se deben almacenar protegidas
 - ◆ Hashes seguros
 - PBKDF2
 - BCRYPT
 - **ARGON2**



Hashes

- PBKDF2
 - ◆ Sistema de generación de claves
 - ◆ Incluye salt, y se indica factor de esfuerzo

Hashes

→ PBKDF2

Protegemos **Passw0rd!** con SALT **Oxdeadbeef**, y
esfuerzo 1.000:

- ◆ $K_0 = \text{HASH}(\text{Passw0rd!}\backslash xde\backslash xad\backslash xbe\backslash xef")$
- ◆ $K_1 = \text{HASH}(K_0 + "\backslash xde\backslash xad\backslash xbe\backslash xef")$
- ◆ ...
- ◆ $K_{1000} = \text{HASH}(K_{999} + "\backslash xde\backslash xad\backslash xbe\backslash xef")$

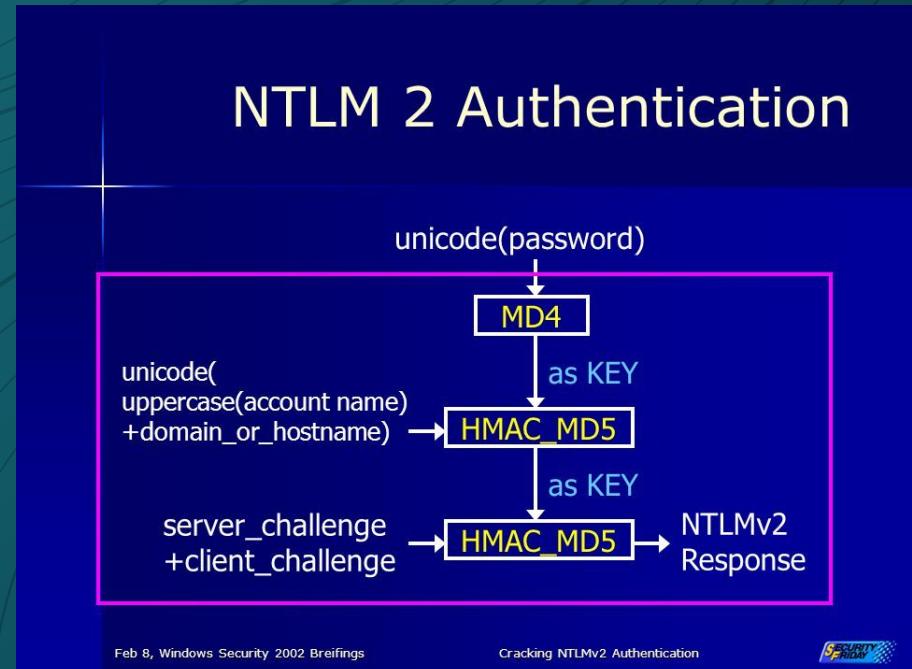
Hashes

- Argon2
 - ◆ Especial contra GPUs
 - ◆ Uso intensivo de memoria
 - ◆ Aleatorización de pasos dependiente de la password
- Modo i → Contra side-channels
- Modo d → Contra GPU
- Modo id → Mixto



PHC 2015

- Hashes
- Hay ciertos hashes especialmente críticos
 - ◆ NTLM





Hashes

→ Hay ciertos hashes especialmente críticos



JWT

The screenshot shows the JUUT JWT debugger interface. At the top, there's a navigation bar with links for Debugger, Libraries, Introduction, Ask, and Get a T-shirt!. To the right, it says "Crafted by Auth0". Below the navigation, there's a dropdown menu for "ALGORITHM" set to "HS256".

The main area is divided into two sections: "Encoded" and "Decoded".

Encoded: A text input field labeled "PASTE A TOKEN HERE" containing the following encoded JWT token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded: A section titled "HEADER: ALGORITHM & TOKEN TYPE" showing the JSON header:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Below that is the "PAYLOAD: DATA" section showing the JSON payload:

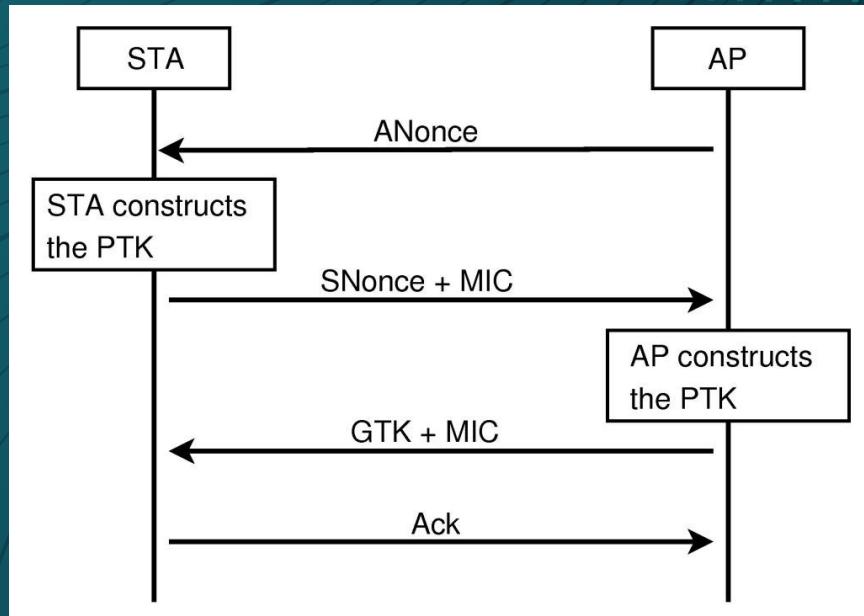
```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

At the bottom, there's a "VERIFY SIGNATURE" section with the HMACSHA256 formula and a placeholder for the secret key:

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

Hashes

- Hay ciertos hashes especialmente críticos
 - ◆ Handshake WPA2
 - HMAC (código MIC)
 - ~PBKDF2 (códigos PTK y PMK)





Enfoque

1. Ataque por diccionario
2. Ataque por patrones
3. Ataque por heurística
4. Ataque por fuerza bruta
5. Rubber hose cryptanalysis

Puesta a punto

```
$ git clone  
https://gitlab.com/ciberseg_uah/public  
/password-cracking.git  
$ cd password-cracking  
$ git submodule init  
$ git submodule update --recursive
```



Puesta a punto

```
$ sudo apt-get -y install \
git build-essential libssl-dev \
zlib1g-dev yasm pkg-config \
libgmp-dev libpcap-dev libbz2-dev \
ocl-icd-opencl-dev opencl-headers \
pocl-opencl-icd
```



Puesta a punto

```
$ cd JohnTheRipper/src/  
$ ./configure  
$ make -s clean && make -sj16
```



Diccionarios

→ **Públicos**

- ◆ SecLists (rockyou, openwall)

→ **“Ocultos”**

- ◆ Pwndb

→ **Reglas**

- ◆ Best64, d3ad0ne, T0X1C, dive → Jumbo

→ **Rainbow Tables**



Diccionarios

→ Diccionario público (ejemplo 1)

```
$ tar xvfvz \
SecLists/Passwords/Leaked-Databases/ \
rockyou.txt.tar.gz
$ JohnTheRipper/run/john \
--wordlist=rockyou.txt ARCHIVO
```



Diccionarios

→ Diccionario público (ejemplo 1)

```
$ JohnTheRipper/run/john \
--show ARCHIVO
```

Diccionarios

→ Diccionario + Reglas (ejemplo 2)

```
$ printf 'Passw0rd!' | md5sum | cut  
-d' ' -f1 > password.txt  
  
$ echo 'password' > dict1.txt  
  
$ JohnTheRipper/run/john  
--wordlist=dict1.txt --rules=best64  
--stdout > dict2.txt
```

Diccionarios

→ Diccionario + Reglas (ejemplo 2)

```
$ JohnTheRipper/run/john  
--wordlist=dict2.txt --rules=best64  
--stdout > dict3.txt  
  
$ JohnTheRipper/run/john  
--format=raw-md5 --wordlist=dict3.txt  
--rules=jumbo password.txt
```



Estadísticas

- Media 7-9 caracteres
- Mayúscula suele ser la primera. Número al final
- Mujeres más nombres personales
- Hombres más hobbies
- \$! #
- 1,2. Si son más de dos números suele ser progresión.



Patrones

- Ingeniería social
- Datos de evidencias forenses
- CUPP
- Combinaciones secuenciales (1234, qwerty) comunes



Patrones

→ Pwndb

- ◆ Visitar <http://pwndb2am4tzkvold.onion/>
 - ◆ Buscar un usuario con patrón
- En el repo tenéis una API para *pwndb*



Patrones

→ Diccionario de usuario

- ◆ Herramienta cupp:
<https://github.com/Mebus/cupp>
- ◆ También lo tenéis en el repo



Patrones

→ Crear máscara + Diccionario (ejemplo 3)

```
$ JohnTheRipper/run/john  
--wordlist=dict.txt --mask='1234?w'  
--stdout  
  
$ JohnTheRipper/run/john  
--wordlist=dict.txt  
--mask=' ?d?d?d?d?w' --stdout
```



Heurísticas

→ Modelos de Markov en John

```
$ printf 'jajaja' | sha256sum | cut  
-d' ' -f1 > password.txt  
$ JohnTheRipper/run/john  
--format=raw-sha256 --markov  
password.txt
```



Fuerza bruta

```
$ printf 'jajaja' | sha1sum | cut -d' ' -f1 > password.txt
$ JohnTheRipper/run/john --list=inc-modes
digits upper lower lowerspace    upperspace
lowernum alpha alnum custom    alnumspace
lanman   lm_ascii  ascii   latin1  utf8
```



Fuerza bruta

```
$ JohnTheRipper/run/john --inc=alpha password.txt  
$ JohnTheRipper/run/john --inc=alnum password.txt  
$ JohnTheRipper/run/john --inc=utf8 password.txt
```



Aplicaciones

→ *2john

- ◆ Además de todas las herramientas de generación de diccionarios y ruptura de contraseñas, tenemos las herramientas de tratamiento de archivos



Aplicaciones

→ *2john

```
$ chmod +x JohnTheRipper/run/*2john*
$ JohnTheRipper/run/keepass2john
KEEPASS.database
database:$keepass$*2*60000*0*f8f8072b4
e5b2bb61498393294538d3ed33d7c1093c04cb
07540ece1eb3da4b8*fdcaa85383c...
```



Aplicaciones

1. Romper KeepassX
2. Romper ZIP
3. Romper PDF



Optimización por GPU





Optimización por GPU

```
JohnTheRipper/run/john \
--fork=8 --format=KeePass-opencl \
--wordlist=rockyou.txt \
keepass-hash.txt
```



Aplicaciones

1. Romper KeepassX
 - a. Rockyou
2. Romper ZIP
3. Romper PDF



Aplicaciones

1. Romper KeepassX
 - a. Rockyou
2. Romper ZIP
 - a. Rockyou + Patrón
3. Romper PDF



Aplicaciones

1. Romper KeepassX
 - a. Rockyou
2. Romper ZIP
 - a. Rockyou + Patrón
3. Romper PDF
 - a. alnum ([a-z][0-9]) < 10 caracteres



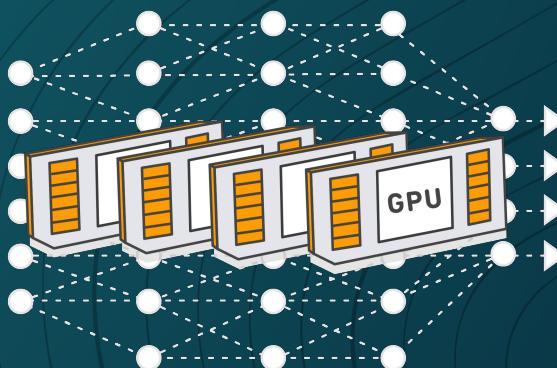
Soluciones en la nube

- Crackstation
- OnlineHashCrack
- AWS

The screenshot shows the CrackStation website's password cracking interface. At the top, there's a navigation bar with links for 'CrackStation', 'Password Hashing Security', and 'Defuse Security'. Below the navigation is a large title 'CrackStation' with a subtitle 'Free Password Hash Cracker'. A text input field is labeled 'Enter up to 20 non-salted hashes, one per line.' To the right of the input field is a reCAPTCHA checkbox labeled 'No soy un robot'. Below the input field is a button labeled 'Crack Hashes'. At the bottom of the interface, there's a note about supported hash types and a link to 'Download CrackStation's Wordlist'.

The screenshot shows the OnlineHashCrack website. The header reads 'OnlineHashCrack is a Cloud Password Recovery Service assisting cyber security experts since 2008'. Below the header are four icons: a cloud icon for 'Cloud-based. No software to install', a lightning bolt icon for 'Fast, accurate & inexpensive', a gear icon for 'Customizable recovery options', and a stack of documents icon for 'Support 78+ algorithms'. The main section has tabs for 'Password/Hashes' (selected), 'WPA / Office / iTunes / Archive / PDF', and 'Max size per file: 200 Mb. We support:'. A list of supported file formats follows:

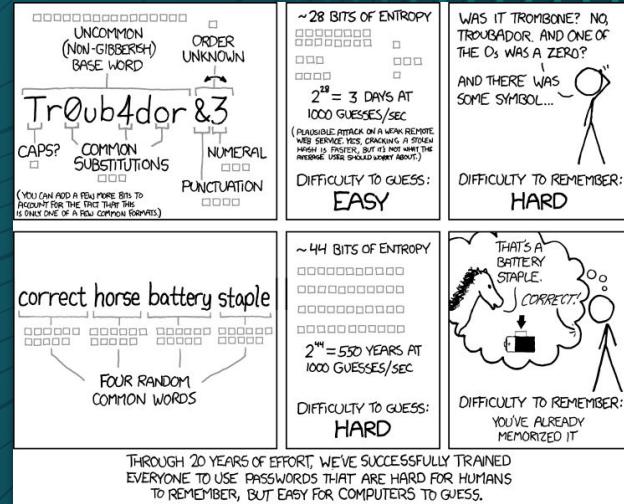
- ✓ WPA WPA(2): pcap & pcapng. Process all ESSIDs and PMKIDs
- ✓ MS Office: encrypted Word, Excel or Powerpoint, version 97 to 2019
- ✓ iTunes Backup: encrypted Apple iTunes Backup Manifest.plist
- ✓ Archives: encrypted ZIP / RAR / 7-zip archives





Recomendaciones

- Longitud: Mínimo 10 caracteres
- ¡No repetirlas!
- Juego de caracteres
 - ◆ ¿Necesitas recordarla?
 - Mejor frase larga
 - ◆ Si no, usa un gestor de contraseñas
- Segundo factor de autenticación





Recomendaciones

- Comprobar seguridad de contraseña
 - ◆ No está en un volcado o diccionario
- Sobre el cambio periódico de contraseñas... No :)
 - ◆ Subscripción a servicios de detección de leaks

The screenshot shows the Firefox Monitor homepage with a dark purple background. At the top, there are three navigation links: "Inicio" (underlined), "Filtraciones", and "Consejos de seguridad". Below the navigation is the Firefox logo and the text "Firefox Monitor". A large white headline reads: "Comprueba si formaste parte de una filtración de datos en línea." Underneath the headline, smaller text says: "Averigua qué saben de ti los hackeadores. Descubre cómo ir siempre un paso por delante." At the bottom, there is a search bar with the placeholder "Introduce una dirección de correo electrónico" and a blue button labeled "Busca filtraciones". A small note at the bottom right states: "Busca tu dirección de correo en filtraciones de datos públicos hasta 2007."



Acknowledgment

- ▷ ProTego is focused in provide a toolkit for health care organisations to better assess and reduce cybersecurity risk
 - ▶ <https://protego-project.eu/>
 - ▶ @protego_project



Bibliografía

- [The Ultimate Guide to Passwords in 2019: Length, Complexity & More!](#)
- [iphelix/pack: PACK \(Password Analysis and Cracking Kit\)](#)
- [Mebus/cupp: Common User Passwords Profiler \(CUPP\)](#)
- [Password Hashing: Scrypt, Bcrypt and ARGON2 - Michele Prezioso](#)
- [Rubber-Hose Cryptanalysis](#)
- [how does NTLM authentication work | What is Active Directory? Microsoft Active Directory Fundamentals with Video Tutorials](#)
- [I know your P4\\$\\$wOrd \(and if i don't, I will guess it\) \(J. Sánchez y P. Caro, Telefónica\)](#)



Thanks!

Any questions?

You can find us at

javier.junquera@uah.es

kevin.van@edu.uah.es