



Introducción

Pablo López e Iván García

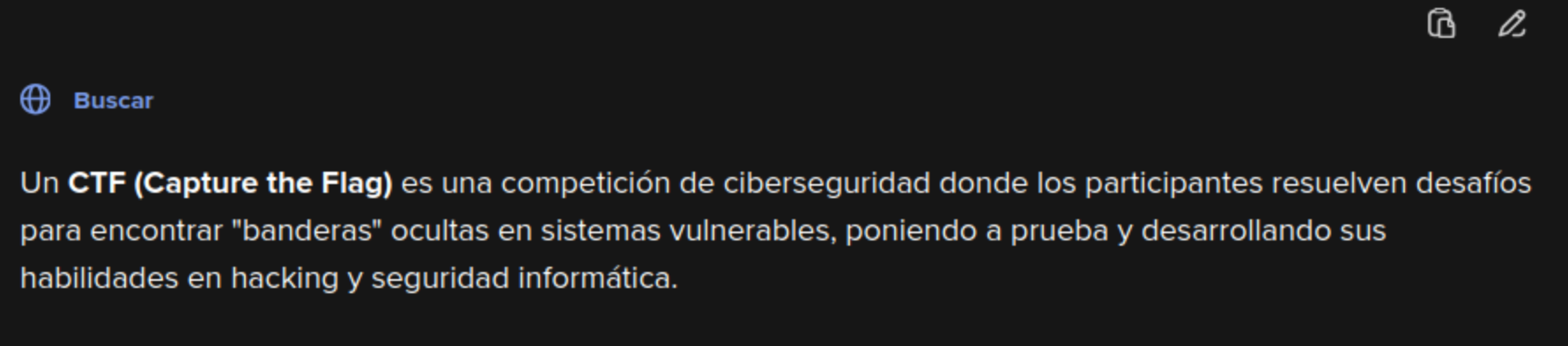
\$> whoarewe



- Qué hacemos?
 1. Curso de introducción a los CTF
 2. Cerves
 3. HackOn
 4. Cerves
 5. Resoluciones de máquinas de HTB
 6. Cerves
 7. Ciber en general
 8. Cerves

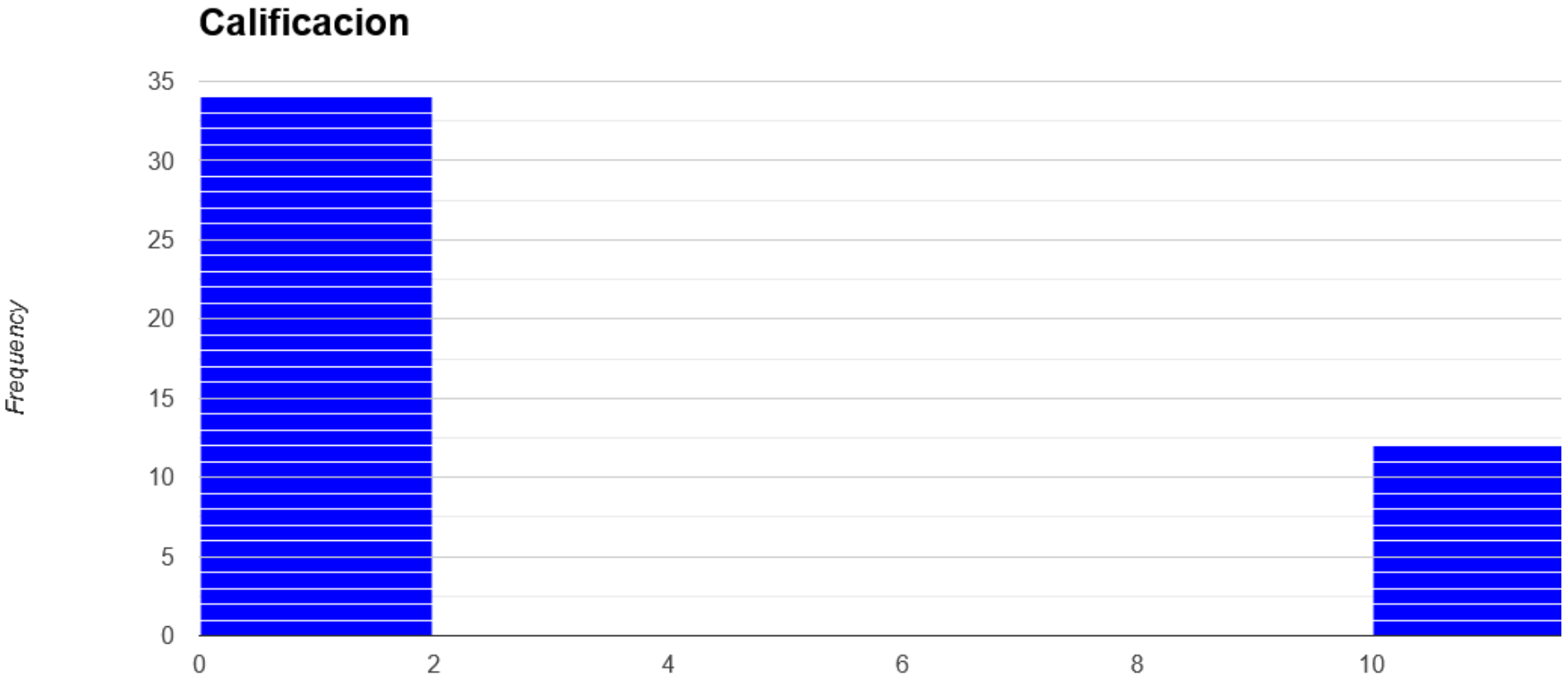
- Cómo me apunto?
 - [Formulario de inscripción](#)
- Qué necesito?
 1. Ganas
 2. Algo de tiempo

Curso de introducción a los CTF

- 
- CTF{
- **Calendario**
<https://urjc-ctf.github.io/web/>
- **Compromiso**
 - 11 Clases para conseguir los créditos
 - NO vengáis por los RAC

Curso de introducción a los CTF

Uso de ChatGPT y derivados





Criptografía básica

Pablo López e Iván García

Índice

1. Criptografía y codificaciones básicas

- Representación de los datos
- Codificaciones y cifrados
- Codificaciones (Binario, Hexadecimal, Base64...)
- XOR
- Hashes (MD5, SHA1, SHA256)

2. Retos básicos

Criptografía - Representación de los datos

Es esencial entender que **nos podemos encontrar los datos con diferentes formatos**. Sin embargo, **su significado será el mismo**. Las formas más comunes son:

ASCII

Relaciona caracteres con números. A cada carácter le corresponde un valor de la tabla ASCII.

Hexadecimal

Utiliza base 16 como representación de los datos. En caracteres toma como referencia el valor ASCII

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Criptografía - Representación de los datos

Es esencial entender que **nos podemos encontrar los datos con diferentes formatos**. Sin embargo, **su significado será el mismo**. Las formas más comunes son:

Binario

Es la representación más básica. Tan solo utiliza dos valores: 1 y 0.

$$\begin{array}{cccccc} 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 32 & +0 & +8 & +4 & +0 & +1 = 45 \end{array}$$

$$\begin{array}{cccccc} 28 & 2 & & & & \\ 0 & 14 & 2 & & & \\ & 0 & 7 & 2 & & \\ & & 1 & 3 & 2 & \\ & & & 1 & 1 & \end{array}$$

$$28 = 11100_2$$

Palabra

CTF{Bienvenidos}

ASCII

067 084 070 123
066 105 101 110
118 101 110 105
100 111 115 125

Binario

01000011 01010100 01000110
01111011 01000010 01101001
01100101 01101110 01110110
01100101 01101110 01101001
01100100 01101111 01110011
01111101

Hexadecimal

43 54 46 7b 42 69 65 6e
76 65 6e 69 64 6f 73 7d e2
80 8b



CyberChef: <https://gchq.github.io/CyberChef/>
Dcode.fr: <https://www.dcode.fr/>



Criptografía - Codificaciones y cifrados

Algunas de las maneras más comunes de ocultar información son mediante **codificaciones y cifrados**. Esto consiste en utilizar una única clave para cifrar y descifrar la información. Por lo tanto, siendo el cifrado **reversible**.

Codificaciones

- Representan la misma información de diferentes maneras.
- Es reversible
- Algunos ejemplos son Base64, Base32 o ASCII

Cifrados

- Ocultan la información mediante claves, normalmente secretas, y un conjunto de operaciones.
- Es reversible
- Algunos ejemplos son ROT-N/César o Vigenère

Operations

Search...

Favourites

Data format

To Hexdump

From Hexdump

To Hex

From Hex

To Charcode

From Charcode

To Decimal

From Decimal

To Binary

From Binary

To Octal

Recipe

From Binary

Delimiter
Space

Byte Length
8

From Base32

Alphabet
A-Z2-7=

☒ Remove non-alphabet chars

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

From Hex

Delimiter
Auto

Input

01001001 01010100 01001100 01001000 01001100 01001010 01001100
01001010 01000001 01010111 01001111 01010100 01010100 01001011
01010011 01000110 01001101 00110101 01001000 01001000 01010101
01000111 01010111 01011010 00110010 01001111 01010000 01001010
01001001 01010110 01010100 01010101 00110100 00110011 01001011
01010010 01001100 01001000 01001010 01010110 01000100 01010101
01000011 01010111 01001111 01010100 01010011 01001000 01001011
01011010 01001101 00110101 01000111 01010111 01010101 01010001
01010011 01011010 00110010 01001111 01001110 01001010 01010110
01001110 01010100 01010101 00110100 00110011 01001011 01011010
01001100 01001000 01001010 01010110 01000100 01010101 01001011
01010111 01001111 01010100 01010011 01010101 01001010 01010110
01001101 00110101 01000111 01010111 01010101 01010001 01001100
01011010 00110010 01001111 01001110 01001010 01000011 01010111
01010100 01010101 00110100 00110010 01010100 01001100 01001101
01001000 01001100 01001010 01001011 01000101 01001011 01011010
01001111 01010100 01001100 01001011 01001001 01010110 01010100
00110101 01001000 01000110 01001001 01010110 01001100 01001000
00110010 01001111 01010000 01001010 01001011 01010111 01001111
01010101 00110100 00110110 01010011 01010110 01001101 00110101

Output

¿Estáis listos chicos?
¡Sí capitán!
¡No oigo!
¡Sí capitán!
Uuuuuh

Criptografía - Codificaciones y cifrados

BASE64

Es un sistema de **numeración posicional** que usa 64 caracteres como base. Sirve para representar cualquier información en binario como texto. **Se suele identificar rápidamente** por su estructura (en general, suelen acabar en ==)

Texto original

CTF{Esto es un texto en Base64. También existen otras como Base32, Base58 o Base85, por ejemplo}

Texto en Base64

QIRGe0VzdG8gZXMgdW4
gdGV4dG8gZW4gYXNINj
QulFRhbWJp6W4gZXhpc3
RlbiBvdHJhcyBjb2IvIEJhc2U
zMiwgQmFzZTU4IG8gQm
FzZTgILCBwb3lgZWplbXB
sb30=

Criptografía - Codificaciones y cifrados

ROT-N

Es un tipo particular de cifrado en el que los caracteres se desplazan N posiciones. Por ello, N será nuestra clave secreta que ayudará a cifrar y descifrar el texto. Además de conocer la clave, deberemos conocer el diccionario que se usa.

Texto original

CTF{El rot solo va a
modificar las letras, pero no
las llaves}

abcdefghijklmnopqrstuvwxyz

Texto en ROT 13

PGS{Ry ebg fbyb in n
zbqvsvpne ynf yrgenf, creb ab
ynf yynirf}

nopqrstuvwxyzabcdefghijklm

Cifrado de
César



EJ I: VVJKQ3tNdXkgYmllbiwgdmVvlHFI ZSBzYWJlcyBpZGVudGlmaWNhciBI biBiYXNINjR9

Ej 2: GXJ{Rs xshsw psw VSX wsr l3}

Ej 3: $-\dots - \dots / \dots - \dots / \dots / -\dots -\dots \dots -\dots -\dots / -\dots -\dots \dots /$

[illegible]

Ejercicios propuestos

Criptografía - Otros cifrados

XOR

Consiste en cifrar siguiendo unas **reglas matemáticas** y una **clave secreta**. Como la **longitud** de la **clave** suele ser **menor al texto**, se repetirá **cíclicamente**. Todos los caracteres se pasarán a binario y se operará con ellos. Reglas:

- 1. Conmutativa:** $A \text{ xor } B = B \text{ xor } A$
- 2. Asociativa:** $(A \text{ xor } B) \text{ xor } C = A \text{ xor } (B \text{ xor } C)$
- 3. Autoinversa:** $(A \text{ xor } B) \text{ xor } B = A$

<i>A</i>	<i>B</i>	XOR
0	0	0
0	1	1
1	0	1
1	1	0

XOR

EJEMPLO I

Esta vez nos dan directamente la flag, pero parece que está cifrada:

Flag: 13 3e 2b 24 3d 3d 14 02 66 1f 04 47 34 09 11 0e 32 0d 41 0b 27 4c 02 0b 27 1a 04 47 28
03 41 02 35 4c 0c 12 3f 4c 12 02 21 19 13 08 3b

Formato de la flag: URJC{}

Challenge Time



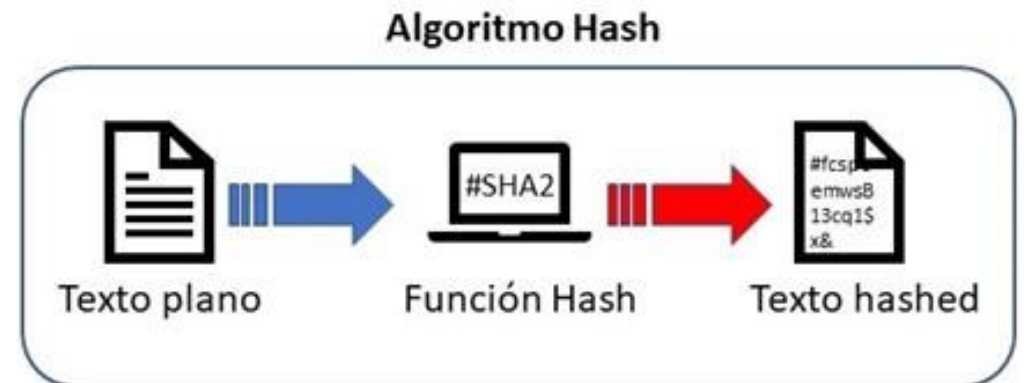
Práctica adicional



[Introduction to CryptoHack](#)

¿Qué es un hash?

- Es una **función matemática o criptográfica**, resume la información
- Da como **resultado** una cadena de caracteres de longitud fija (**digest**), **independientemente** de la longitud entrada
- Es **irreversible (One Way)**. Una vez aplicada **no se puede obtener el valor inicial**.



Criptografía - Otras codificaciones

¿Qué es un hash?

Your String	Hola Mundo	
MD5 Hash	d501194c987486789bb01b50dc1a0adb	Copy
SHA1 Hash	48124d6dc3b2e693a207667c32ac672414913994	Copy

Your String	En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo...	
MD5 Hash	8ad3504f03861ad37fd575ebef0ebe9f	Copy
SHA1 Hash	125bf1068008747a2095af763b940d374174592d	Copy

Propiedades

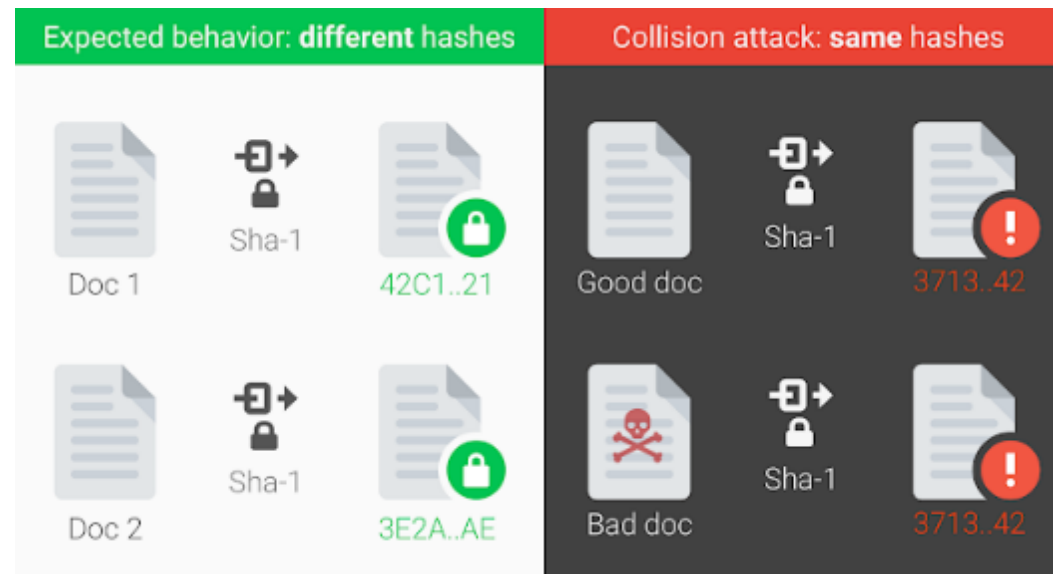
- **Collision Resistance (CR)**
 - Dado un hash H , encontrar dos mensajes m y m' tal que $H(m) = H(m')$
- **Target-collision resistance (TCR)**
 - Dado un hash H y un mensaje m , encontrar m' tal que $H(m) = H(m')$
- **Preimage resistance(PR)**
 - Dado un hash H y una imagen I , encontrar m tal que $H(m) = I$

Criptografía - Hashes

- Lo que sí puede hacerse es **pre-computar** cadenas típicas, dado que una función hash devolverá el mismo resultado para la misma cadena (es determinista)
- **Conociendo la función utilizada** podemos realizar ataques de **fuerza bruta** sobre los hashes, de forma que, si en nuestro **diccionario** se encuentra la palabra *hasheada*, **sabremos qué esconde el hash**
- Es importante destacar que esto **NO ES LO MISMO QUE REVERTIR EL CÁLCULO**
- **Intentar adivinar un hash** de una palabra de longitud mayor que 8 es **computacionalmente muy costoso**

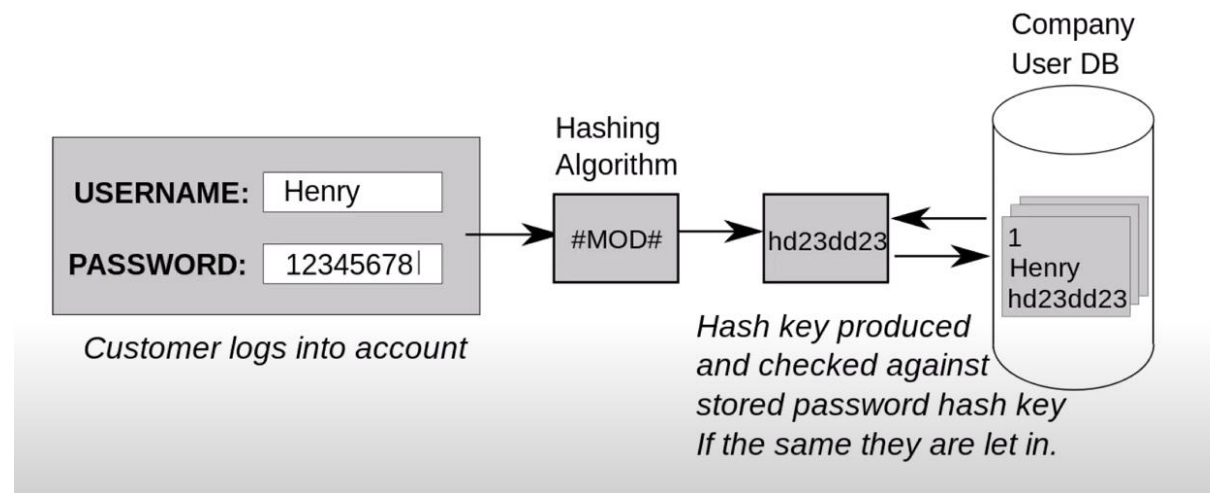
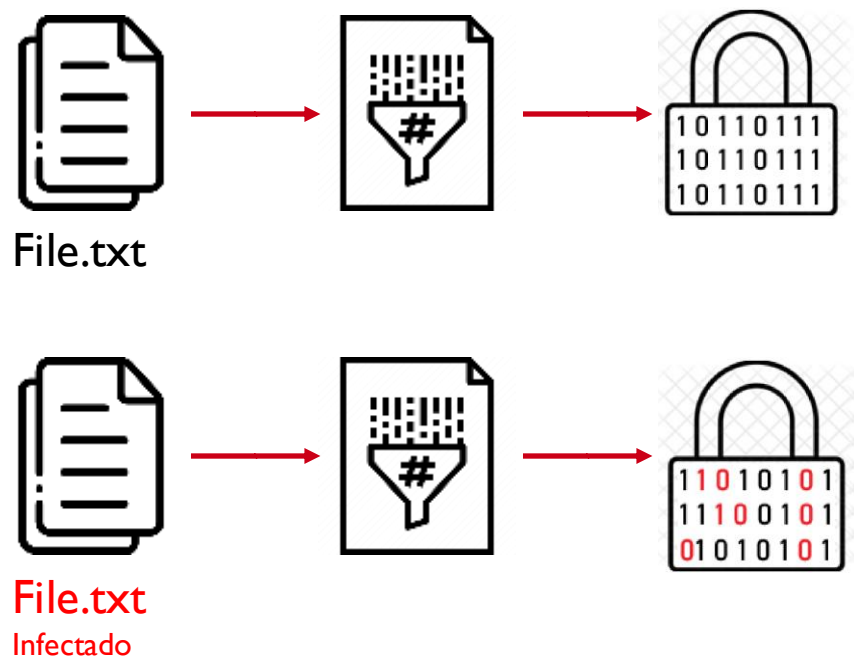
Criptografía - Hashes

- Existen determinadas funciones hash cuyo uso **no se recomienda**
 - **MD5**
 - **SHA1**
- Aunque la probabilidad es muy baja, podrían existir **colisiones**



¿Para qué sirven los hashes?

Criptografía - Hashes



Criptografía - Hashes

- Cada **fichero** se puede resumir con un **valor hash**
- Existen herramientas que, dada una lista de **hashes**, nos automatizan el proceso de obtener un valor que genere dicho hash.
- **Esto permite obtener la contraseña de ficheros cifrados**



Criptografía - Hashes



Ataque por fuerza bruta (3)

```
(kali㉿kali)-[~/Documents]
$ hashcat -a 3 -m 0 hash.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEP)
* Device #1: cpu-sandybridge-11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz, 2919/5902 MB (100%)

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Candidates.#1..... 34y / xqx
Hardware.Mon.#1..: Util: 66%

4d186321c1a7f0f354b297e8914ab240:hola

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
```

Ataque por diccionario/wordlist (0)

```
(kali㉿kali)-[~/Documents]
$ hashcat -a 0 -m 0 hash.txt rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEP)
* Device #1: cpu-sandybridge-11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz, 2919/5902 MB (100%)

4d186321c1a7f0f354b297e8914ab240:hola

(kali㉿kali)-[~/Documents]
```

Ataque por fuerza bruta con mascara (3)


```
(kali㉿kali)-[~/Documents]
$ hashcat -a 3 -m 0 hash.txt ?uabc?d?s
hashcat (v6.2.6) starting


OpenCL API (OpenCL 3.0 PoCL 5.0+debian Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEP)
* Device #1: cpu-sandybridge-11th Gen Intel(R) Core(TM) i7-11700K @ 3.60GHz, 2919/5902 MB (100%)


Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
```

Criptografía - Hashes

https://hashes.com/en/tools/hash_identifier

 **Proceeded!**
1 hashes were checked: 1 possibly identified 0 no identification

 **Pay professionals to decrypt your remaining lists**
<https://hashes.com/en/escrow/view>

 **Possible identifications:** [Decrypt Hashes](#)

d41d8cd98f00b204e9800998ecf8427e - Possible algorithms: MD5

SEARCH AGAIN

Criptografía - Hashes

https://hashcat.net/wiki/doku.php?id=example_hashes

Generic hash types

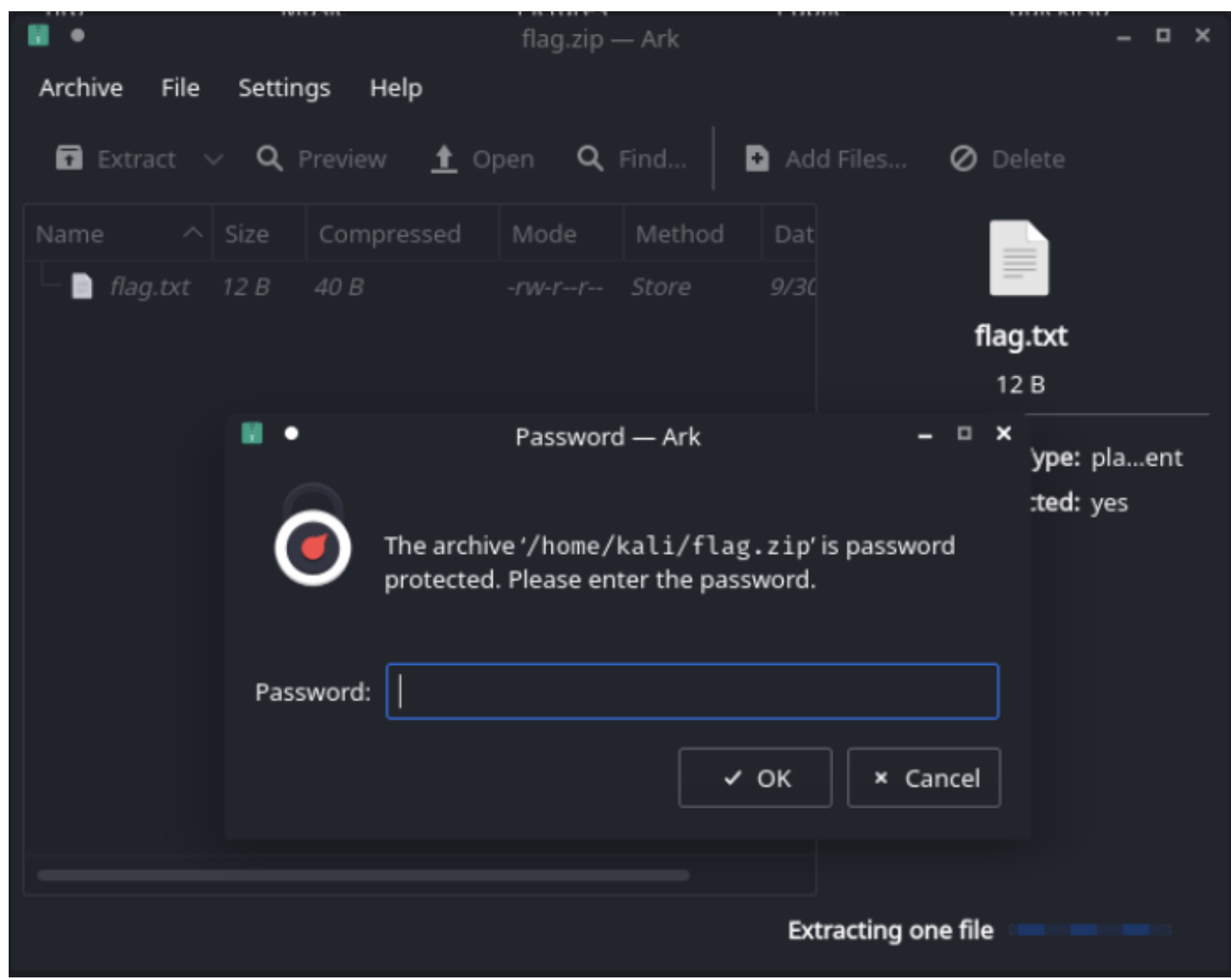
Hash-Mode	Hash-Name	Example
0	MD5	8743b52063cd84097a65d1633f5c74f5
10	md5(\$pass.\$salt)	01dfae6e5d4d90d9892622325959afbe:70
20	md5(\$salt.\$pass)	f0fda58630310a6dd91a7d8f0a4ceda2:422
30	md5(utf16le(\$pass).\$salt)	b31d032cfdcf47a399990a71e43c5d2a:14
40	md5(\$salt.utf16le(\$pass))	d63d0e21fdc05f618d55ef306c54af82:132
50	HMAC-MD5 (key = \$pass)	fc741db0a2968c39d9c2a5cc75b05370:12
60	HMAC-MD5 (key = \$salt)	bfd280436f45fa38eaacac3b00518f29:123
70	md5(utf16le(\$pass))	2303b15bfa48c74a74758135a0df1201
100	SHA1	b89eaac7e61417341b710b727768294d0e

hashcat -m 0 -a 3 hashes.txt

-m: Código del tipo de hash a crackear

-a: Tipo de ataque (Fuerza bruta con o sin mascarar o diccionario)

Criptografía – Hashes (Ejemplo)



Criptografía – Hashes (Ejemplo)



```
~: zsh — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Left/Right Split View Top/Bottom
(kali@kali)-[~]
$ zip2john flag.zip > hashZip
```

```
~: zsh — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Left/Right Split View Top/Bottom Load a new tab with layout 2x2 terminals
(kali@kali)-[~]
$ zip2john flag.zip | grep -E -o '(\$pkzip2\$.*\$/pkzip2\$) | (\$zip2\$.*\$/zip2\$)' > zipHash2hashcat
```


Criptografía – Hashes (Ejemplo)



```
~: zsh — Konsole
File Edit View Bookmarks Plugins Settings Help

New Tab Split View Left/Right Split View Top/Bottom

(kali@kali)-[~]
$ john hashZip --wordlist=wordlists.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 SSE2 4x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 1 candidate left, minimum 16 needed for performance.
hola1234 (flag.zip/flag.txt)
1g 0:00:00:00 DONE (2021-09-30 16:55) 100.0g/s 100.0p/s 100.0c/s 100.0C/s hola1234
Use the "--show" option to display all of the cracked passwords reliably
Session completed

(kali@kali)-[~]
$ john hashZip --show
flag.zip/flag.txt:hola1234:flag.txt:flag.zip:flag.zip

1 password hash cracked, 0 left

(kali@kali)-[~]
$
```

```
(kali@kali)-[~]
$ hashcat -m 13600 zipHash2hashcat ./wordlists.txt
hashcat (v6.1.1) starting... (press Ctrl-C to stop)
```

```
Session.....: hashcat
Status.....: Cracked
Hash.Name.....: WinZip
Hash.Target.....: $zip2$*0*3*0*f819c01513f1f5018f4e73128d711b52*8d6c* ... /zip2$
Time.Started.....: Thu Sep 30 16:59:35 2021 (0 secs)
Time.Estimated...: Thu Sep 30 16:59:35 2021 (0 secs)
Guess.Base.....: File (./wordlists.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 3 H/s (1.66ms) @ Accel:64 Loops:999 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 1/1 (100.00%)
Rejected.....: 0/1 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-999
Candidates.#1....: hola1234 → hola1234

Started: Thu Sep 30 16:58:55 2021
Stopped: Thu Sep 30 16:59:37 2021

(kali@kali)-[~]
$ hashcat -m 13600 zipHash2hashcat --show
$zip2$*0*3*0*f819c01513f1f5018f4e73128d711b52*8d6c*c*327662bd488eec34fe3ad3fa*4b36073395bdba927dda*$/zip2$:hola1234

(kali@kali)-[~]
$
```

Criptografía - Hashes

<https://crackstation.net/>

tion

y ⌵ Defuse Security ⌵

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

8743b52063cd84097a65d1633f5c74f5

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Res
8743b52063cd84097a65d1633f5c74f5	md5	hashcat

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

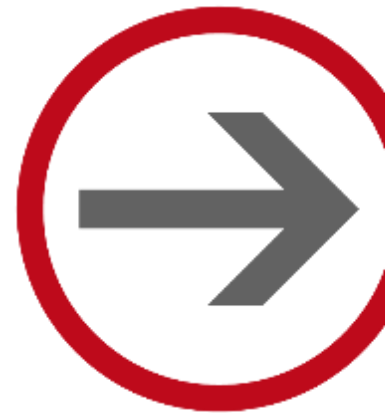
[Download CrackStation's Wordlist](#)

Para practicar lo aprendido

- Para **practicar lo que hemos visto hasta ahora**, podéis realizar los **primeros 7 retos** de la categoría ***Básica*** de la plataforma **Atenea**

<https://atenea.ccn-cert.cni.es/challenges>

- Estos retos resumen **lo visto hasta ahora**
- Plataforma **Cryptohack**: <https://cryptohack.org/courses/>



RETOS BÁSICOS
