CracktheHash

En este reto nos embarcaremos en la magia de identificar los hash y crackearlos

Nivel 1:

1. 48bb6e862e54f2a795ffc4e541caed4d

para esto lo primero será identificar , la naturaleza del mismo , utilizaremos una herramienta llamada hash identifier:

Posible MD5 ;D
para crackear MD5 se utiliza hashcat

como es parte de un ctf y las preguntas dicen que se debe ocupar rockyou utilizaremos este metodo hashcat -m 0 Valor del hash /usr/share/wordlists/rockyou.txt

```
┌──(viernez13㉿kali)-[~]
└─$ hashcat -m 0 48bb6e862e54f2a795ffc4e541caed4d /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.7, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==========================================================================================================================================
* Device #1: cpu-sandybridge-AMD A10-7860K Radeon R7, 12 Compute Cores 4C+8G, 2251/4566 MB (1024 MB allocatable), 2MCU

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

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Initializing backend runtime for device #1. Please be patient ...
```

la hemos crackeado!

```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

48bb6e862e54f2a795ffc4e541caed4d:easy

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 48bb6e862e54f2a795ffc4e541caed4d
Time.Started.....: Mon Feb  5 00:21:03 2024 (0 secs)
Time.Estimated...: Mon Feb  5 00:21:03 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   868.3 kH/s (0.32ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 173056/14344385 (1.21%)
Rejected.........: 0/173056 (0.00%)
Restore.Point....: 172032/14344385 (1.20%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: florida69 → convict1
Hardware.Mon.#1..: Util: 52%

Started: Mon Feb  5 00:20:22 2024
Stopped: Mon Feb  5 00:21:05 2024
```

es easy!

Segundo nivel!

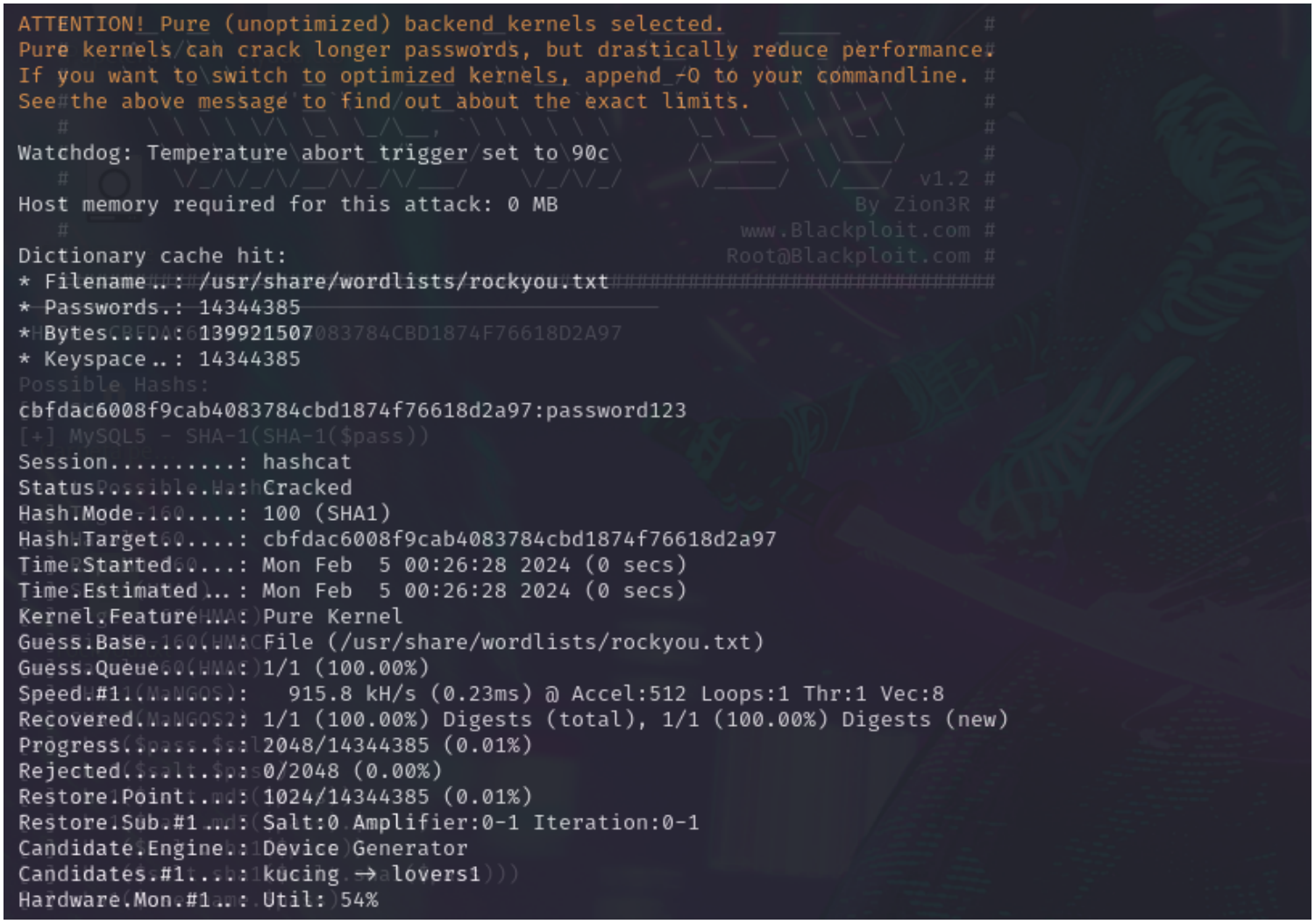
2. CBFDAC6008F9CAB4083784CBD1874F76618D2A97
D: identificamos

de seguro en esta parte se podrían preguntar como se puede saber el modulo a ejecutar , se puede adquirir desde acá : https://hashcat.net/wiki/doku.php?id=example_hashes

en el caso de Sha1 ,es este:

100	SHA1	b89eaac7e61417341b710b727768294d0e6a277b

probamos el comando :
hashcat -m 100 CBFDAC6008F9CAB4083784CBD1874F76618D2A97 /usr/share/wordlists/rockyou.txt



password123

Nivel 3!

3. 1C8BFE8F801D79745C4631D09FFF36C82AA37FC4CCE4FC946683D7B336B63032
mismo procedimiento :
acá hay un truquito , SHA-256 es SHA2-256

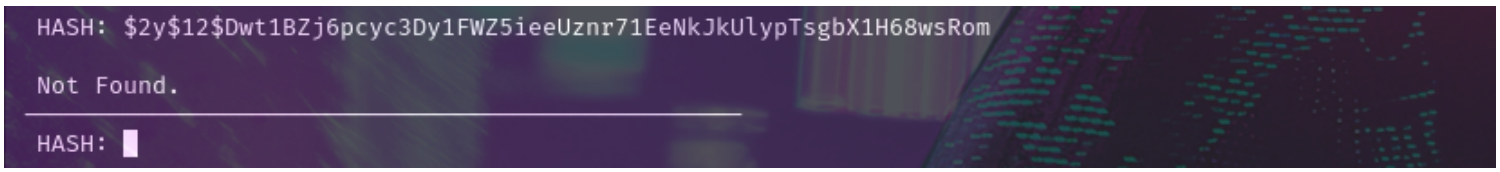
1400	SHA2-256	127e6fbfe24a750e72930c220a8e138275656b8e5d8f48a98c3c92df2caba935

password , letmein

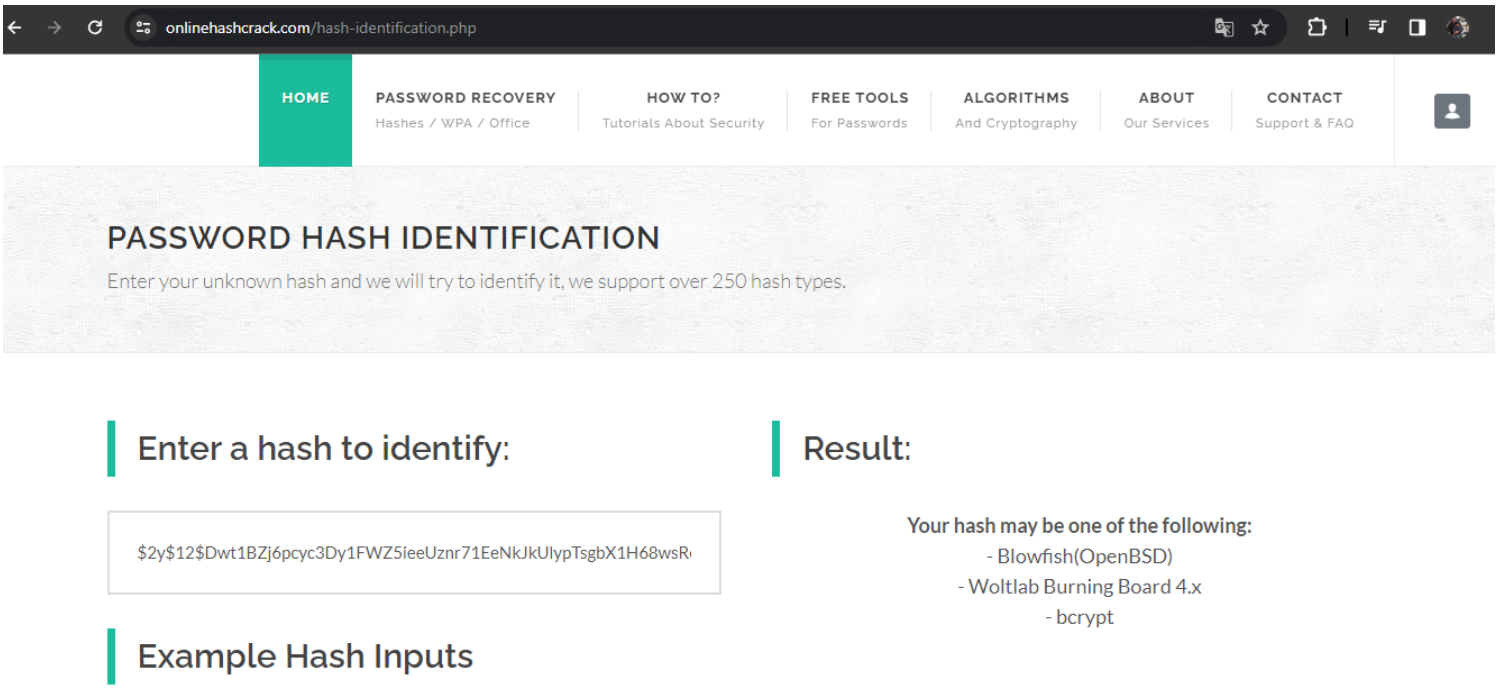
Avancemos al siguiente Nivel.

4. $2y$12$Dwt1BZj6pcyc3Dy1FWZ5ieeUznr71EeNkJkUlypTsgbX1H68wsRom

acá se pone más compleja la pista , debido a que hash-identifier , no detecta el hash!



buscando en internet llegué a esto buscando la naturaleza del resultado , llegue a una pagina donde muestran diferentes algoritmos :
https://www.onlinehashcrack.com/hash-acceptance.php



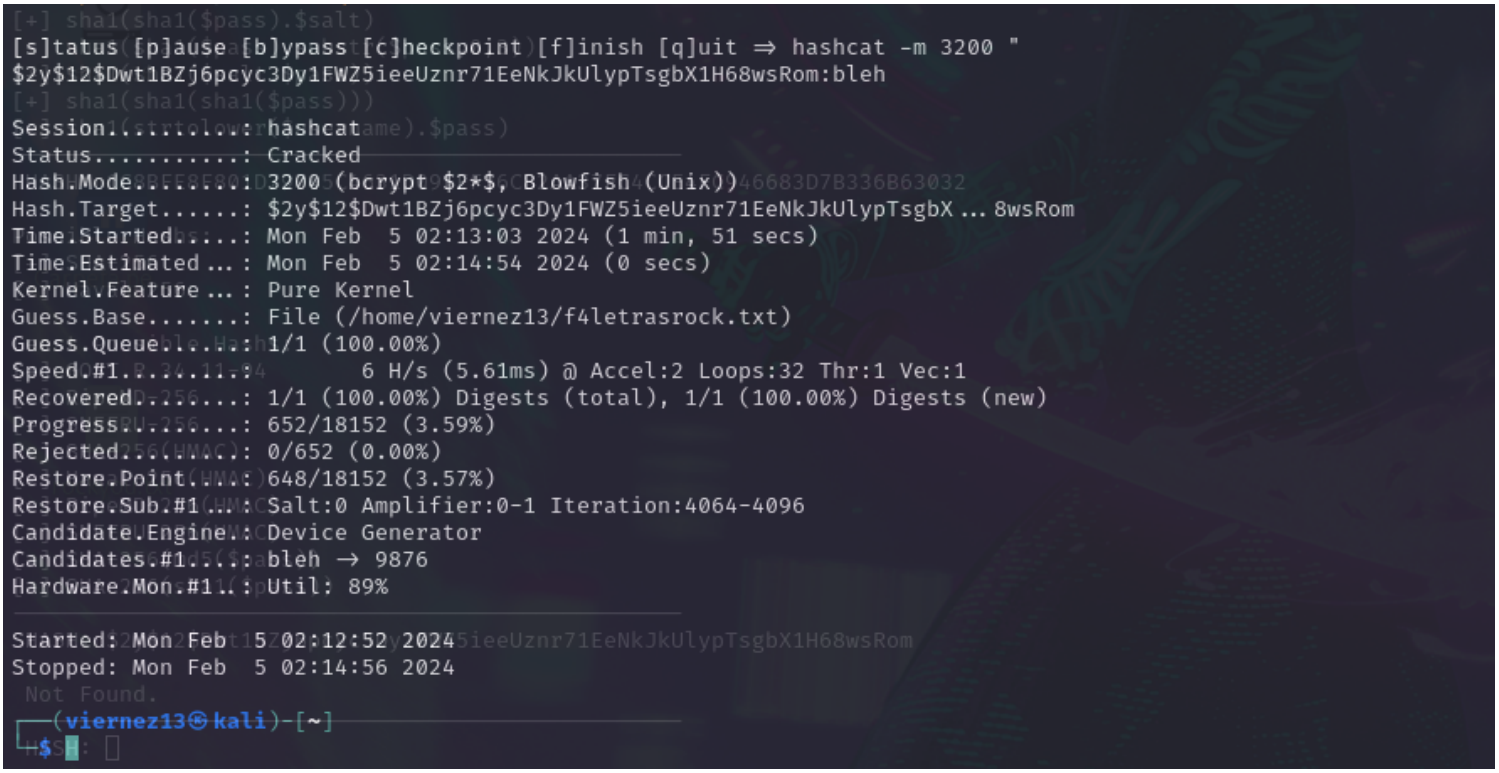
Blowfish(OpenBsd)

hashcat -m 3200 "\$2y\$12\$Dwt1BZj6pcyc3Dy1FWZ5ieeUznr71EeNkJkUlypTsgbX1H68wsRom" /usr/share/wordlists/rockyou.txt
acá habrá que separar la codificación lo que hice fue ponerle \ antes de cada $ para que sea reconocido.

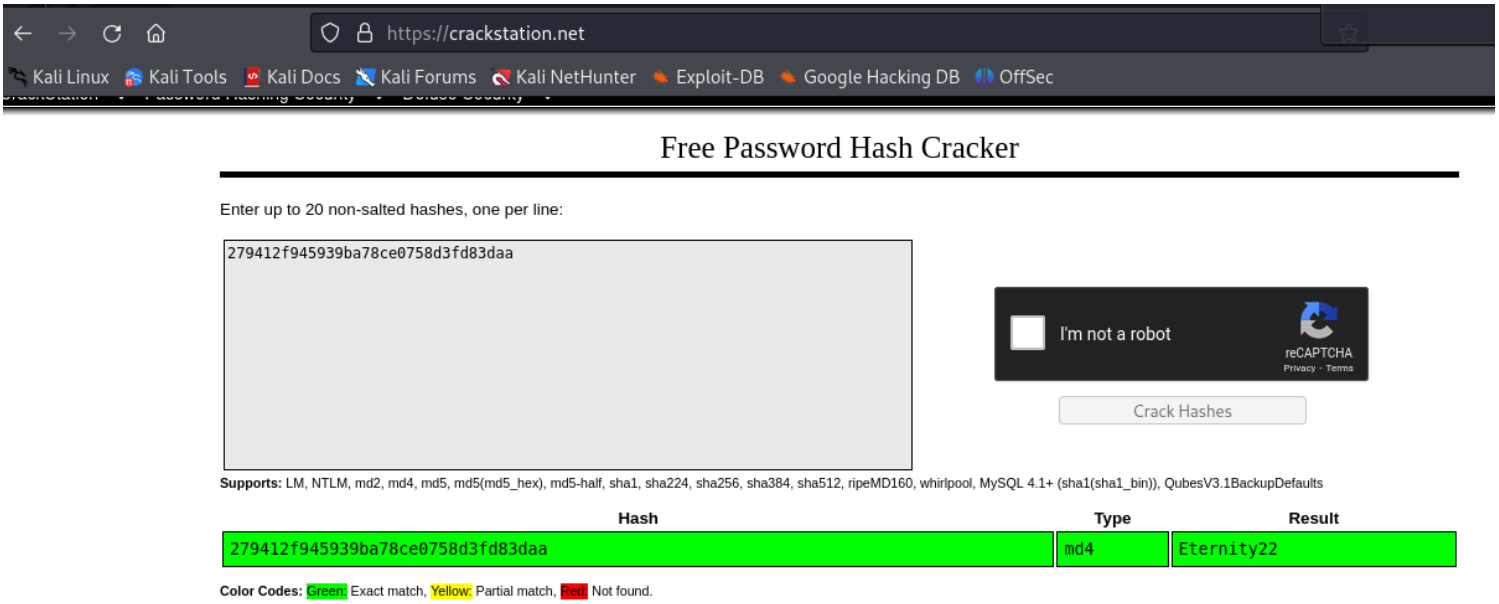
y haremos una trampa , debido a que si usamos rockyou completo tardará app 27 días en crackearla xD
como sabemos que la clave de la primera es de 4 digitos convertiremos rockyou en un diccionario de solo 4 caracteres alfanumericos.

para esto , primero creamos el dicc
awk 'length==4' /usr/share/wordlists/rockyou.txt > /home/viernez13/f4letrasrock.txt

luego lanzaremos el hashcat
hashcat -m 3200 "\$2y\$12\$Dwt1BZj6pcyc3Dy1FWZ5ieeUznr71EeNkJkUlypTsgbX1H68wsRom" /home/viernez13/f4letrasrock.txt

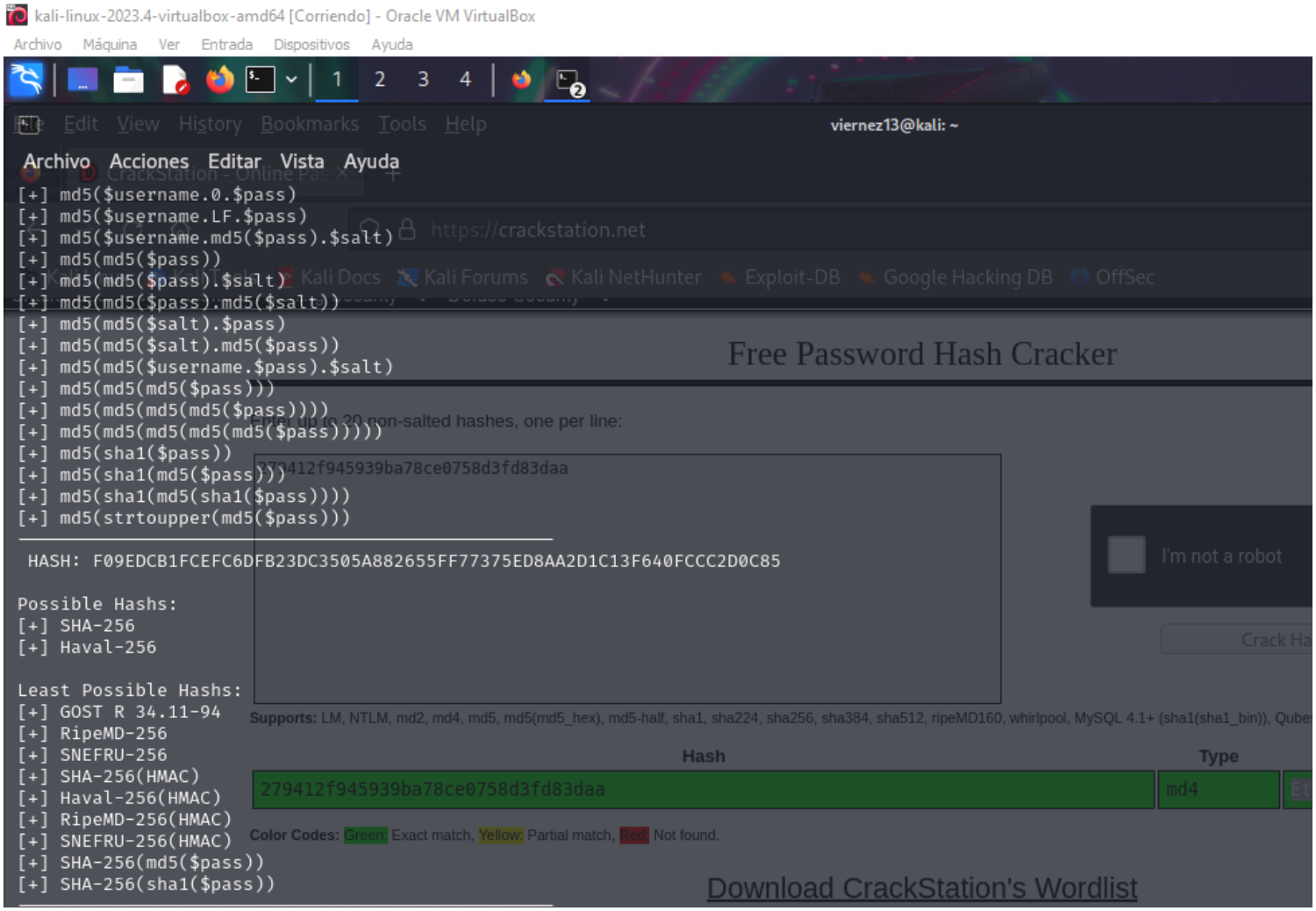


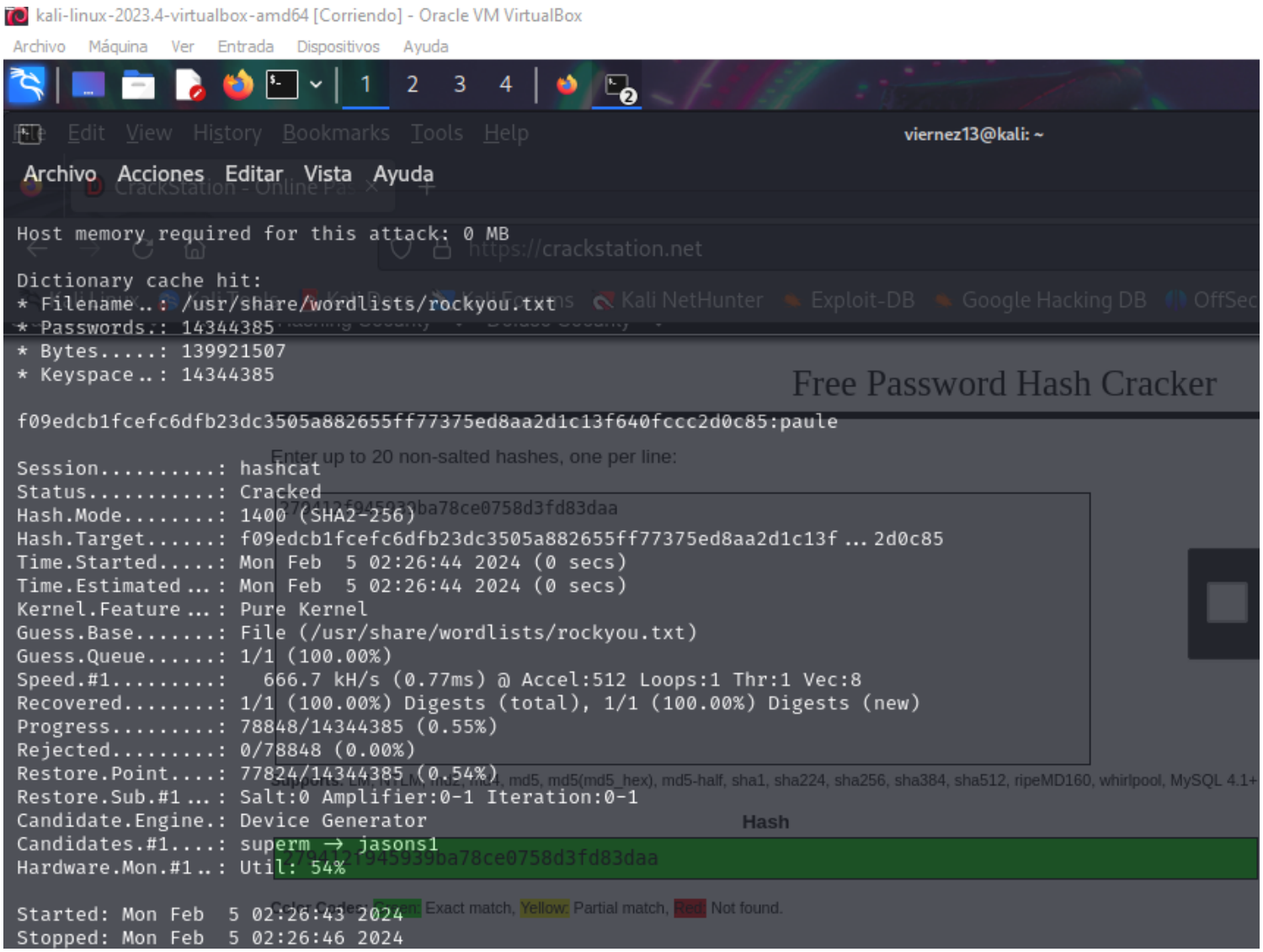
luego de unos minutos , hemos conseguido la clave!! ;D
nivel5
traté de identificar el hash resultando en un MD5 lo cual no era , por lo cual fui por la vía fácil , me fui a crackstation y fue correcto era un md4



2 Tarea , nivel 1
F09EDCB1FCEFC6DFB23DC3505A882655FF77375ED8AA2D1C13F640FCCC2D0C85

fue reconocido como sha-256
es el modulo 1400 , hacemos la magia

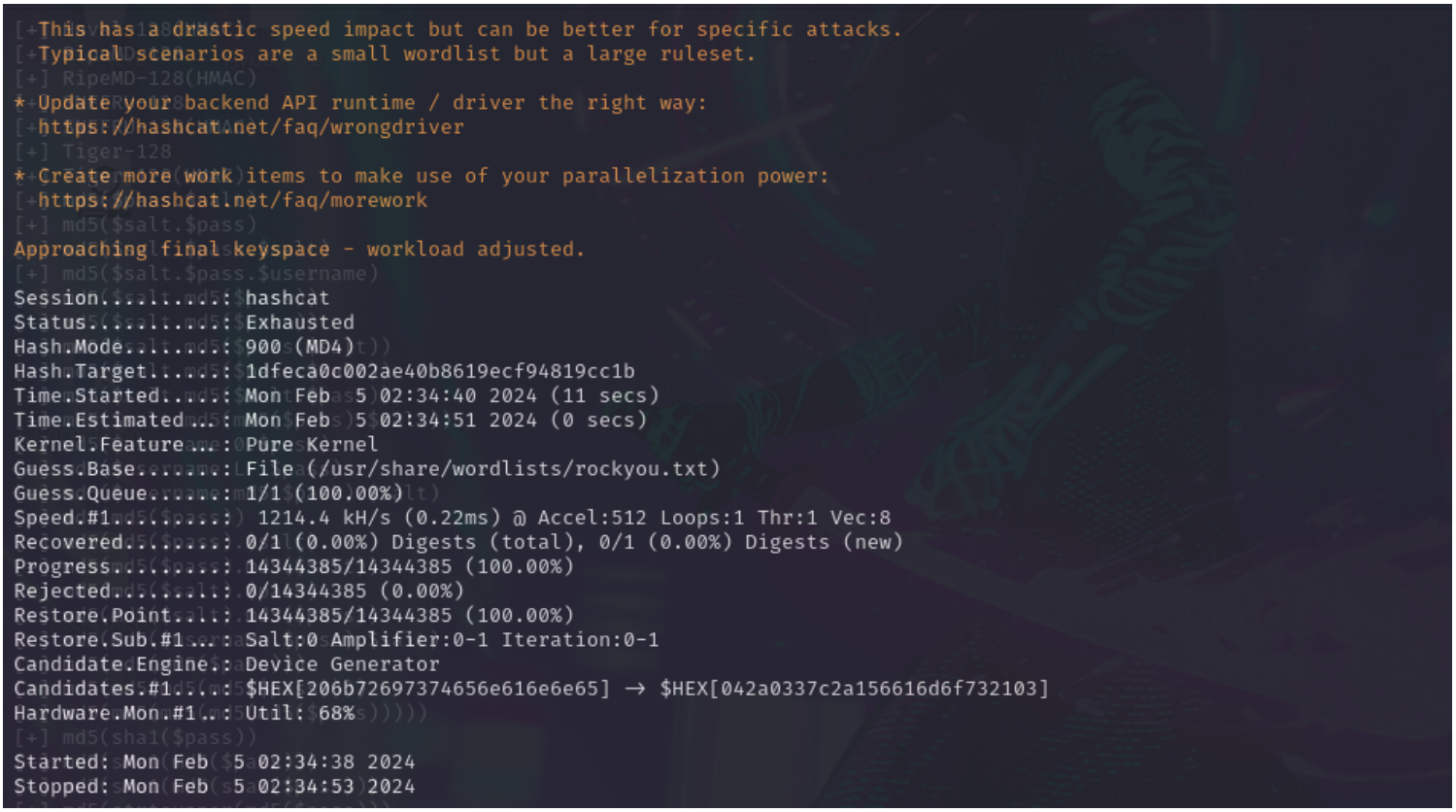

kali-linux-2023.4-virtualbox-amd64 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
viernez13@kali: ~
Archivo Acciones Editar Vista Ayuda
[+] md5($username.0.$pass)
[+] md5($username.LF.$pass)
[+] md5($username.md5($pass).$salt)
[+] md5(md5($pass))
[+] md5(md5($pass).$salt)
[+] md5(md5($pass).md5($salt))
[+] md5(md5($salt).$pass)
[+] md5(md5($salt).md5($pass))
[+] md5(md5($username.$pass).$salt)
[+] md5(md5(md5($pass)))
[+] md5(md5(md5(md5($pass))))
[+] md5(md5(md5(md5(md5($pass)))))
[+] md5(sha1($pass))
[+] md5(sha1(md5($pass)))
[+] md5(sha1(md5(sha1($pass))))
[+] md5(strtoupper(md5($pass)))
HASH: F09EDCB1FCEFC6DFB23DC3505A882655FF77375ED8AA2D1C13F640FCCC2D0C85
Possible Hashs:
[+] SHA-256
[+] Haval-256
Least Possible Hashs:
[+] GOST R 34.11-94
[+] RipeMD-256
[+] SNEFRU-256
[+] SHA-256(HMAC)
[+] Haval-256(HMAC)
[+] RipeMD-256(HMAC)
[+] SNEFRU-256(HMAC)
[+] SHA-256(md5($pass))
[+] SHA-256(sha1($pass))
https://crackstation.net
Free Password Hash Cracker
279412f945939ba78ce0758d3fd83daa
I'm not a robot
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), Qube
Hash
Type
279412f945939ba78ce0758d3fd83daa
md4
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.
Download CrackStation's Wordlist



Nivel 2
1DFECA0C002AE40B8619ECF94819CC1B

Probé MD5 sin resultado

Probé MD4 sin resultado

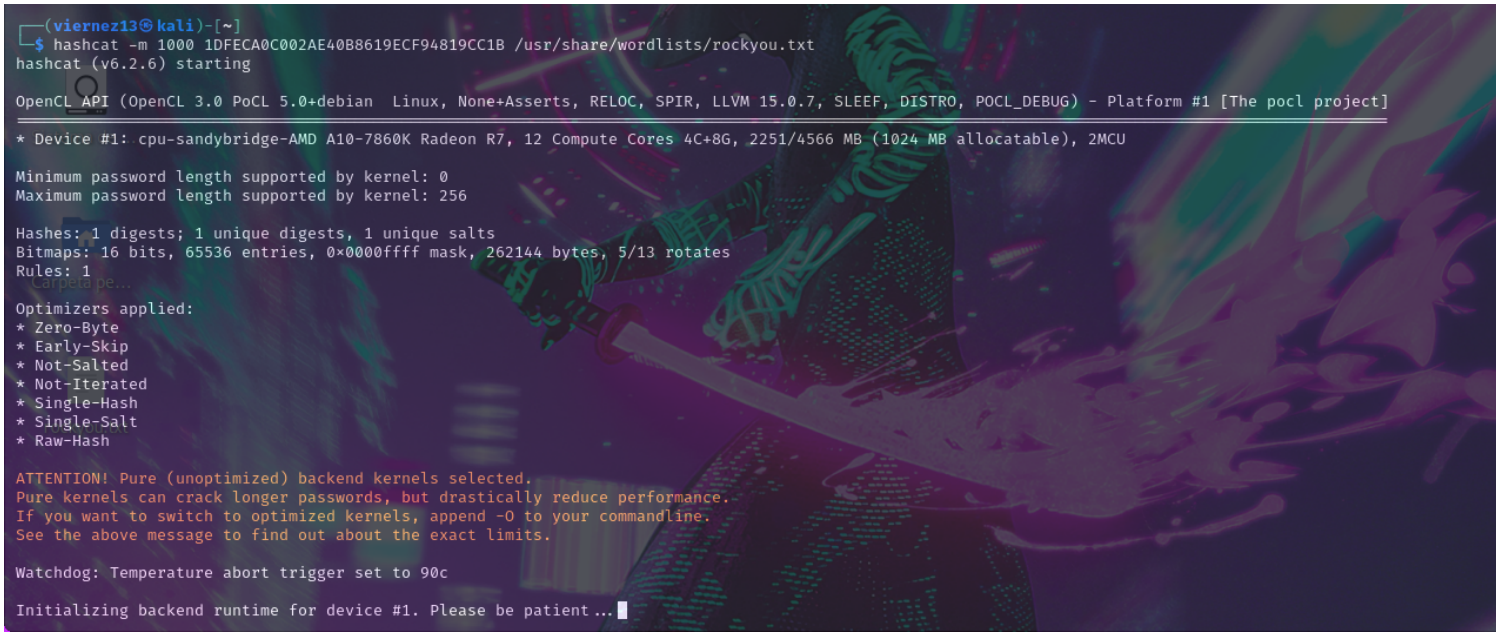


el reconocimiento que podrán ser posibles otros..

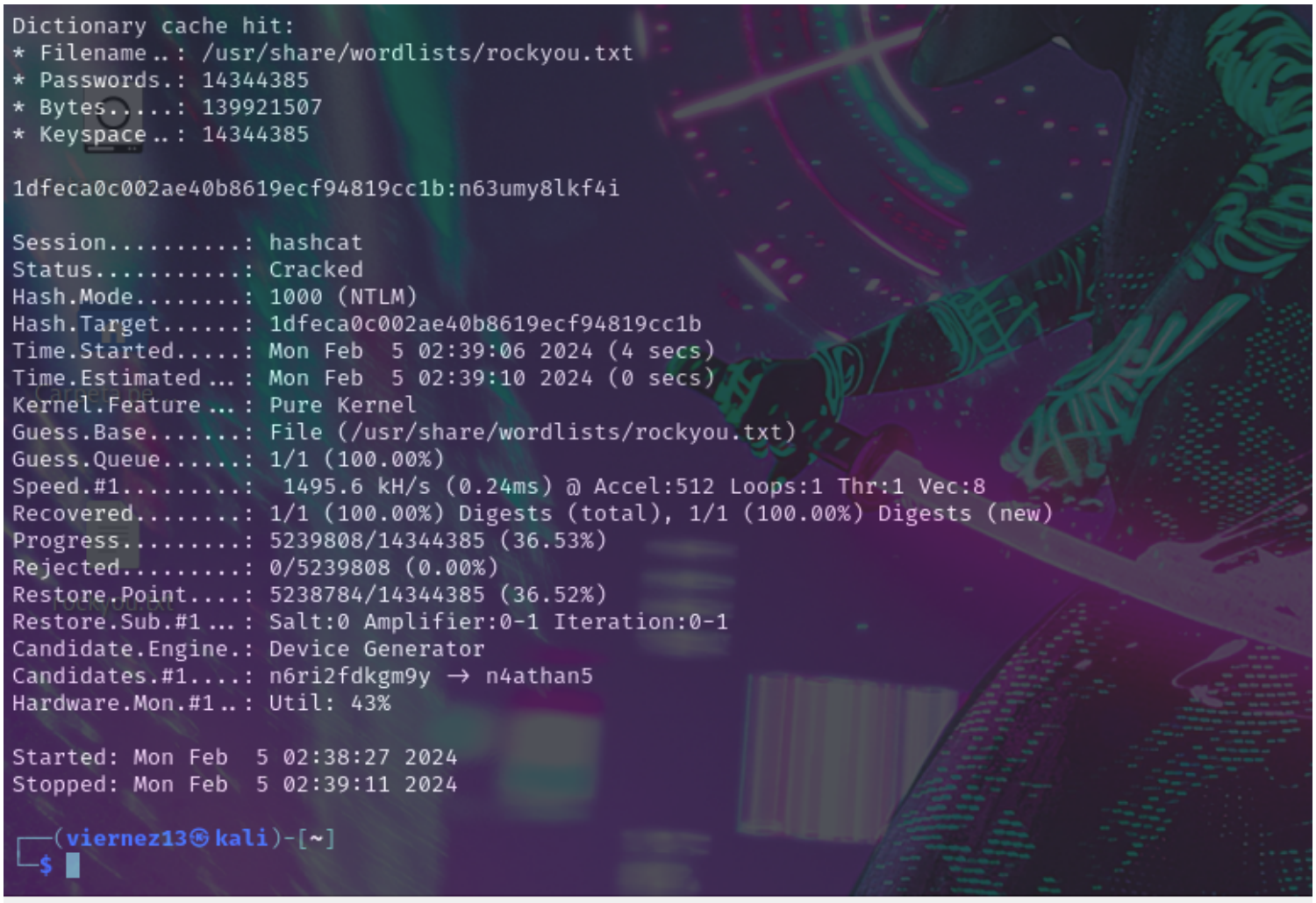
probaré con NTLM
modo de ntlm es 1000

1000	NTLM	b4b9b02e6f09a9bd760f388b67351e2b

hashcat -m 1000 1DFECA0C002AE40B8619ECF94819CC1B /usr/share/wordlists/rockyou.txt



era NTLM

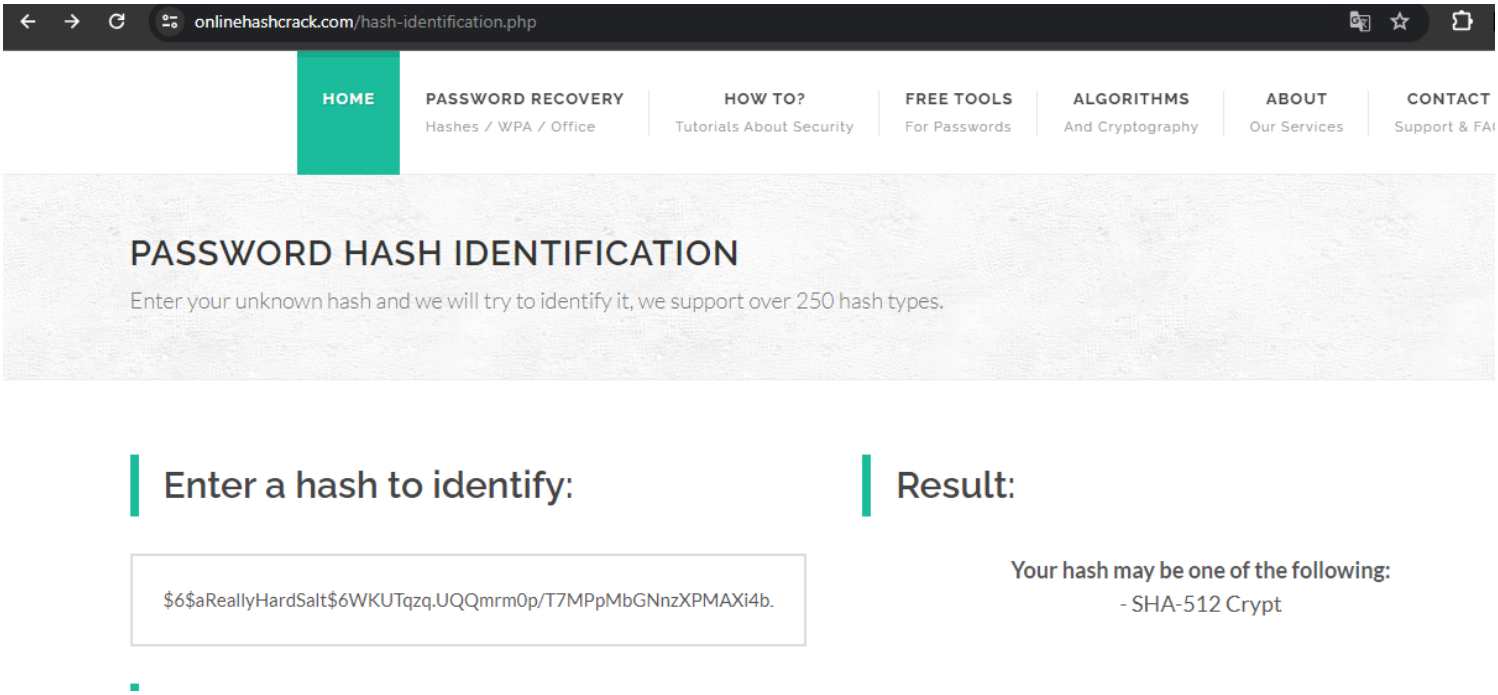


nivel3

Hash: 6aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02.
Salt: aReallyHardSalt
acá se complica un poco la cosa , ya comenzamos a tener salt...

pero ¿qué es un salt?
Salt es un número de dígitos aleatorios que se le agrega a la contraseña ya sea al principio o al final y que el usuario no conocerá. Con esto, la contraseña se hace de mayor longitud y por lo tanto más compleja, de esta manera será mucho más difícil encontrar el hash en una tabla y más aún obtener la contraseña, ya que está combinada con el salt.
De esta manera, el salt constituye un método de seguridad que los sistemas deben usar para la protección de las contraseñas.

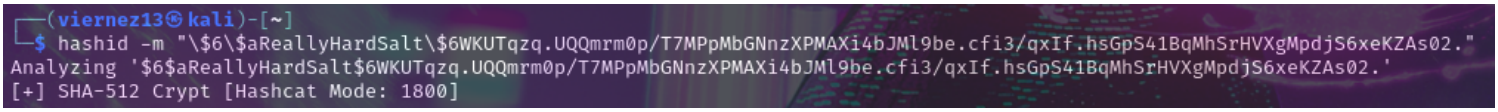
Identificamos el cifrado :



ahora buscaremos el módulo , podemos hacerlo con hashid

hashid -m "\$6\$aReallyHardSalt\$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02."
Analyzing '6aReallyHardSalt$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02.'

se les recuerda separar , siempre antes de un $ con un \

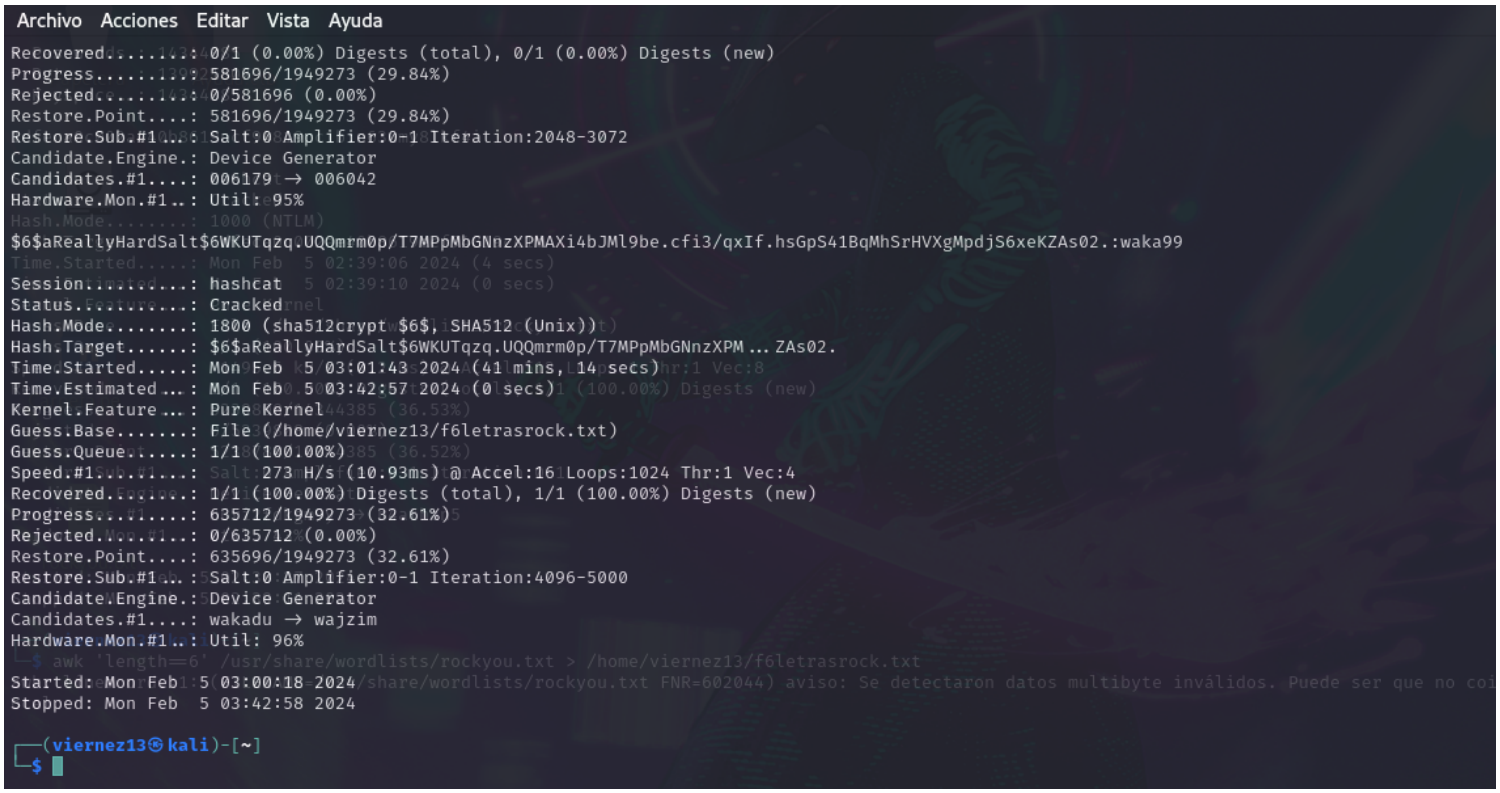


el crackeo de SHA-512 puede ser tardío por lo cual aplicaremos la misma trampa que hicimos hace unos niveles atrás

son 6 digitos

awk 'length==6' /usr/share/wordlists/rockyou.txt > /home/viernez13/f6letrasrock.txt

hashcat -m 1800 "\$6\$aReallyHardSalt\$6WKUTqzq.UQQmrm0p/T7MPpMbGNnzXPMAXi4bJMl9be.cfi3/qxIf.hsGpS41BqMhSrHVXgMpdjS6xeKZAs02." /home/viernez13/f6letrasrock.txt



luego de 40 minutos o un poco mas logramos la contraseña.

Ultimo nivel!

Hash: e5d8870e5bdd26602cab8dbe07a942c8669e56d6
Salt: tryhackme
Identificamos.-

```
Stopped: Mon Feb  5 03:42:58 2024
┌──(viernez13㉿kali)-[~]
└─$ hashid -m e5d8870e5bdd26602cab8dbe07a942c8669e56d6
Analyzing 'e5d8870e5bdd26602cab8dbe07a942c8669e56d6'
[+] SHA-1 [Hashcat Mode: 100]
[+] Double SHA-1 [Hashcat Mode: 4500]
[+] RIPEMD-160 [Hashcat Mode: 6000]
[+] Haval-160
[+] Tiger-160
[+] HAS-160
[+] LinkedIn [Hashcat Mode: 190]
[+] Skein-256(160)
[+] Skein-512(160)
```

investigando descubri que el modo HMAC-SHA1 pass+salt corresponde al 160
160

160	HMAC-SHA1 (key = $salt)	d89c92b4400b15c39e462a8caa939ab40c3aeeea:1234

hashcat -m 160 e5d8870e5bdd26602cab8dbe07a942c8669e56d6:tryhackme /usr/share/wordlists/rockyou.txt

```
* Typical scenarios are a small wordlist but a large ruleset.
* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

Hash.Mode.........: 1000 (NTLM)
e5d8870e5bdd26602cab8dbe07a942c8669e56d6:tryhackme:481616481616

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 160 (HMAC-SHA1 (key = $salt))
Hash.Target......: e5d8870e5bdd26602cab8dbe07a942c8669e56d6:tryhackme
Time.Started.....: Mon Feb  5 04:12:33 2024 (22 secs)
Time.Estimated...: Mon Feb  5 04:12:55 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 524.5 kH/s (1.43ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 12313600/14344385 (85.84%)
Rejected.........: 0/12313600 (0.00%)
Restore.Point....: 12312576/14344385 (85.84%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 48162450 → 4812475
Hardware.Mon.#1..: Util: 46%

Started: Mon Feb  5 04:12:01 2024
Stopped: Mon Feb  5 04:12:57 2024
```

y hemos resuelto la máquina :D