Una-al-mes: Silicon Valley - Episodio#2 - Hispasec

Enunciado CTF:

Dinesh ha perdido la clave VERDADERA que usaba para abrir su zip secreto pero gracias a DIOS tiene un archivo .raw donde puede recuperarla y necesita que le echemos una mano.

A Dinesh le encantan los mensajes con doble sentido, debéis tenerlo en cuenta...

Archivo .raw (escoged el que mejor os venga):
https://www.mediafire.com/file/piv4t8514bp5dpg/pied_piper_bak.zip/file

https://mega.nz/#!iAUDnKwA!Y2g23qnZ9rwZvzZA3Bg8cbENe_ZtASOi1NFgrgfL8sg

Info: Las pistas os servirán a partir de que tengáis la contraseña del zip adjunto (Secretos_Dinesh.zip). Recordad que flag.txt tiene dos cifrados (leed bien README).

Info: La flag tiene el formato UAM{md5}

Resolución:

Nos bajamos el archivo pied_piper_bak.raw del enunciado, un dump de memoria.

Para analizarlo hacemos uso de la herramienta **Volatility**:

Usamos **imageinfo** para ver el profile que tenemos que utilizar para su análisis:

```
$ ~/Documentos/CTF/SiliconValley2$ python ~/code/volatility/vol.py -f
pied_piper_bak.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64,
Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000,
Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
```

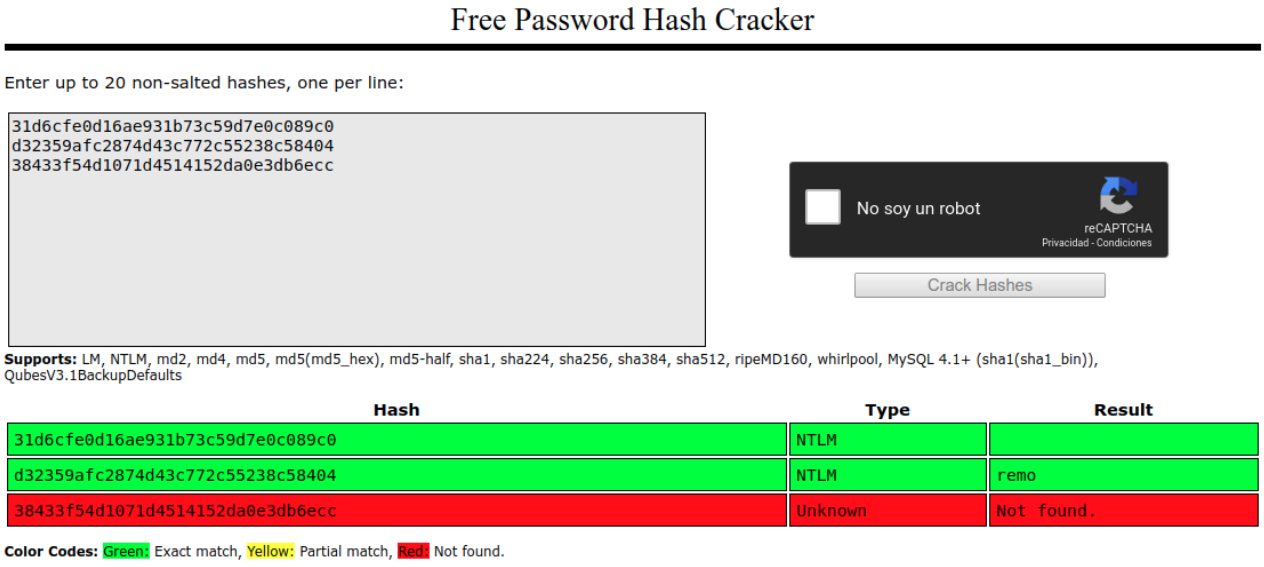
Usando un profile de la lista procedemos a su análisis, por ejemplo Win7SP1x64.

Analizamos los posibles hashes de usuarios del sistema con **hashdump**:

```
$ ~/Documentos/CTF/SiliconValley2$ python ~/code/volatility/vol.py -f
pied_piper_bak.raw --profile=Win7SP1x64 hashdump
Volatility Foundation Volatility Framework 2.6
Volatility Foundation Volatility Framework 2.6
Administrador:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c08
9c0:::
Invitado:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
:
Richard:1001:aad3b435b51404eeaad3b435b51404ee:d32359afc2874d43c772c55238c58404::
:
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:38433f54d1071d4514152da0e3d
b6ecc:::
```

Haciendo uso de **crackstation** (https://crackstation.net/) buscamos las claves para los usuarios encontrados.

Si intentamos usarlas como password para el zip, no tenemos un resultado positivo.



Free Password Hash Cracker

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

```
31d6cfe0d16ae931b73c59d7e0c089c0
d32359afc2874d43c772c55238c58404
38433f54d1071d4514152da0e3db6ecc
```

No soy un robot — reCAPTCHA Privacidad - Condiciones

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	Result
31d6cfe0d16ae931b73c59d7e0c089c0	NTLM	
d32359afc2874d43c772c55238c58404	NTLM	remo
38433f54d1071d4514152da0e3db6ecc	Unknown	Not found.

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

Vemos los procesos que hay en ejecución con **pslist**.

```
$ ~/Documentos/CTF/SiliconValley2$ python ~/code/volatility/vol.py -f
pied_piper_bak.raw --profile=Win7SP1x64 pslist
0xfffffa80012d9b30 DB Browser for         1836   1464      9      345      1
0 2018-10-15 10:26:16 UTC+0000
0xfffffa8001355060 notepad.exe            2616   1464      1       62      1
0 2018-10-15 10:29:08 UTC+0000
0xfffffa8002b7a1b0 cmd.exe                2312   1464      1       21      1
0 2018-10-15 10:32:00 UTC+0000
0xfffffa8001273b30 conhost.exe            2200    412      2       51      1
0 2018-10-15 10:32:00 UTC+0000
0xfffffa80013286e0 notepad.exe            1520   1464      1       62      1
0 2018-10-15 10:39:42 UTC+0000
(salida truncada)
```

De la lista me llama la atención **notepad.exe** y **DB Browser for SQLite.**

Buscamos archivos de notepad con **filescan** y hacemos un **grep** con su extensión:

```
$ ~/Documentos/CTF/SiliconValley2$ python ~/code/volatility/vol.py -f
pied_piper_bak.raw --profile=Win7SP1x64 filescan | grep -e .txt
Volatility Foundation Volatility Framework 2.6
0x000000004146c660     16      0 R--rwd
\Device\HarddiskVolume2\Users\Richard\Desktop\piper.txt
0x0000000041515630      1      1 -W-rw-
\Device\HarddiskVolume2\Users\Richard\AppData\Local\Temp\FXSAPIDebugLogFile.txt
```

También encontramos este archivo que es una base de datos SQLite:

```
$ ~/Documentos/CTF/SiliconValley2$ python ~/code/volatility/vol.py -f
pied_piper_bak.raw --profile=Win7SP1x64 filescan | grep -e "piperdb.db"
Volatility Foundation Volatility Framework 2.6
0x0000000040501860     16      0 R--rw-
```

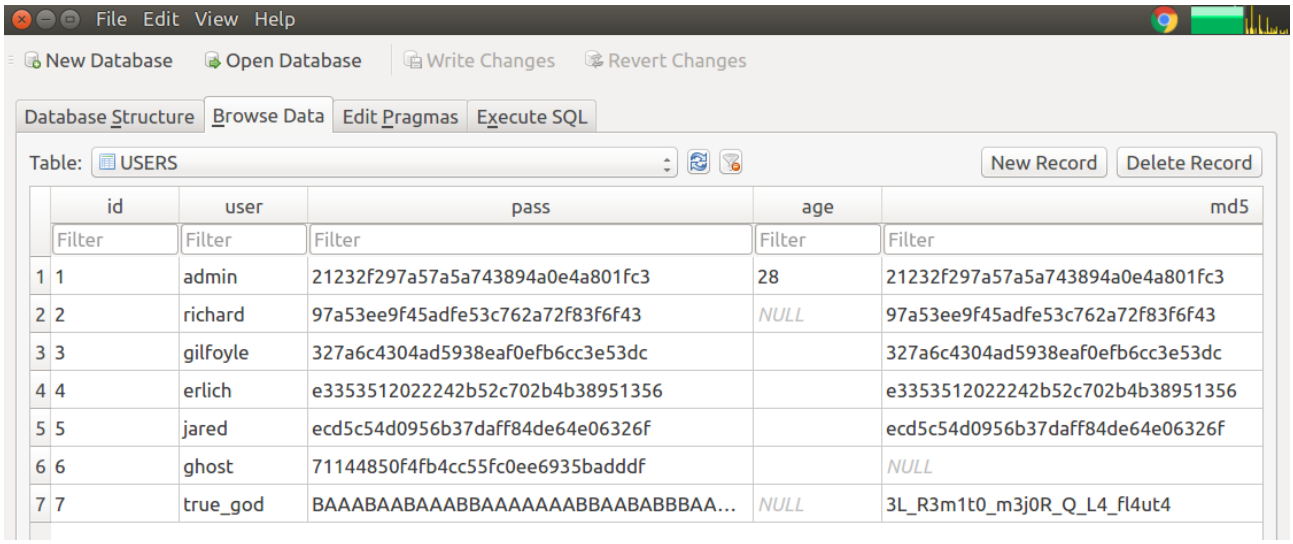
```
\Device\HarddiskVolume2\Users\Richard\Desktop\piperdb.db
0x000000004123a8d0      1       1 RW-rw-
\Device\HarddiskVolume2\Users\Richard\Desktop\piperdb.db
```

Y procedemos a hacer el dump de piperdb.db con **dumpfiles**:

```
$ ~/Documentos/CTF/SiliconValley2$ python ~/code/volatility/vol.py -f
pied_piper_bak.raw --profile=Win7SP1x64 dumpfiles -Q 0x0000000040501860 -D
/dumps/ -n
```

Los archivos .txt no nos aportan nada, pero la base de datos tiene información interesante que procedemos a analizar.

Abrimos la base de datos con la aplicación "**DB Browser for SQLite**" y encontramos la tabla usuarios:



	id	user	pass	age	md5
	Filter	Filter	Filter	Filter	Filter
1	1	admin	21232f297a57a5a743894a0e4a801fc3	28	21232f297a57a5a743894a0e4a801fc3
2	2	richard	97a53ee9f45adfe53c762a72f83f6f43	NULL	97a53ee9f45adfe53c762a72f83f6f43
3	3	gilfoyle	327a6c4304ad5938eaf0efb6cc3e53dc		327a6c4304ad5938eaf0efb6cc3e53dc
4	4	erlich	e3353512022242b52c702b4b38951356		e3353512022242b52c702b4b38951356
5	5	jared	ecd5c54d0956b37daff84de64e06326f		ecd5c54d0956b37daff84de64e06326f
6	6	ghost	71144850f4fb4cc55fc0ee6935badddf		NULL
7	7	true_god	BAAABAABAAABBAAAAAAABBAABABBBAA...	NULL	3L_R3m1t0_m3j0R_Q_L4_fl4ut4

Probamos con todos los valores de md5 de la columna "pass" como password del archivo zip y su equivalente en texto plano, pero no son passwords válidas:

Id	User	Pass (Md5 Encrypted)	Md5 Decrypted
1	admin	21232f297a57a5a743894a0e4a801fc3	admin
2	richard	97a53ee9f45adfe53c762a72f83f6f43	remo
3	gilfoyle	327a6c4304ad5938eaf0efb6cc3e53dc	flag
4	erlich	e3353512022242b52c702b4b38951356	r3m0
5	jared	ecd5c54d0956b37daff84de64e06326f	(no encontrada)
6	ghost	71144850f4fb4cc55fc0ee6935badddf	ghost
7	true_god	**BAAABAABAAABBAAAAAAABBAABABBBAAAAAAAAAABAAAAAAAABAABBBAABBABABAAAAAAAAABBAB**	3L_R3m1t0_m3j0R_Q_L4_fl4ut4

El enunciado nos da la pista de "VERDADERA – DIOS", que equivale al usuario "true_god".

Su pass está codificada en Beacon, así que usamos esta web para decodificarlo:
https://www.dcode.fr/bacon-cipher
Obteniendo así el password para el archivo "Secretos_Dinesh.zip":

```
REMAZOABACONIAN
```

Dentro del archivo .zip encontramos un README con las pistas para el doble cifrado del flag:

```
1. "We are the DATE" https://www.youtube.com/watch?v=tYIYRRLj-n4

2. La clave final de todo está en el corazón de Telegram, en sus comienzos...
```

Y el flag cifrado:

```
2Dd!E2(^as/MoI>2)$U91G(::/MJn20JtF90J+t:/N#@:2)?
gA1+b1>/N#772)Hm=2D$U>/N#@:0OcUk1+b@B/MK+82)Hm=2D$U?/MJk12)
[$D1bCCA/N#=90KC^B1+b1:/MJn21hIk2@<3Q#Bk;05+F.B<FCcS7F_,)l+Dk\-Df[N
```

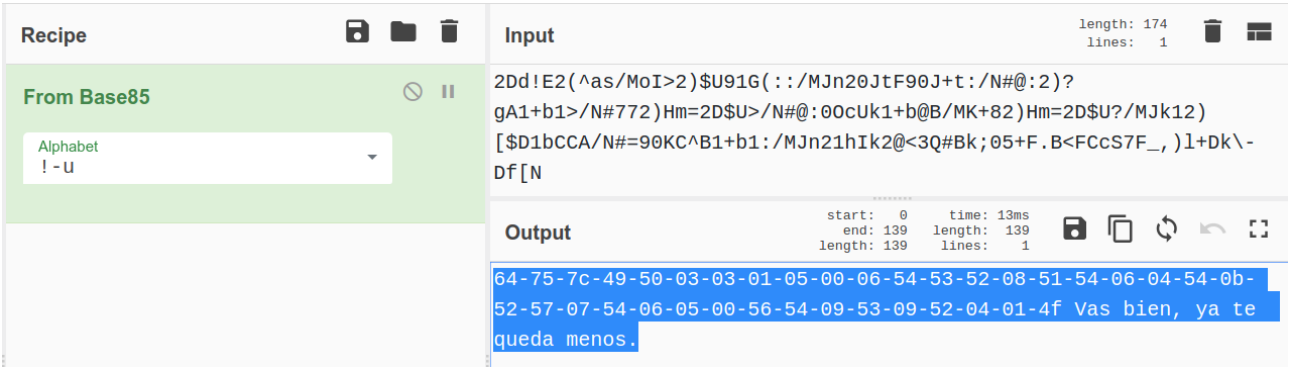
- **Primera parte:**

Accedemos al vídeo en Youtube teniendo en mente que DATE está en mayúsculas:

USA for Africa - We Are The World - 1985

La fecha de la canción es de 1985. Pensando en posibles codificaciones con esas cifras, tenemos el **Base85**.

https://gchq.github.io/CyberChef/



La primera parte del cifrado la tenemos resuelta:

```
64-75-7c-49-50-03-03-01-05-00-06-54-53-52-08-51-54-06-04-54-0b-52-57-07-54-06-
05-00-56-54-09-53-09-52-04-01-4f Vas bien, ya te queda menos.
```

- **Segunda parte:**

Tenemos un conjunto de 74 cifras hexadecimales. La pista nos habla de una clave, relacionada con el corazón de Telegram y sus comienzos.

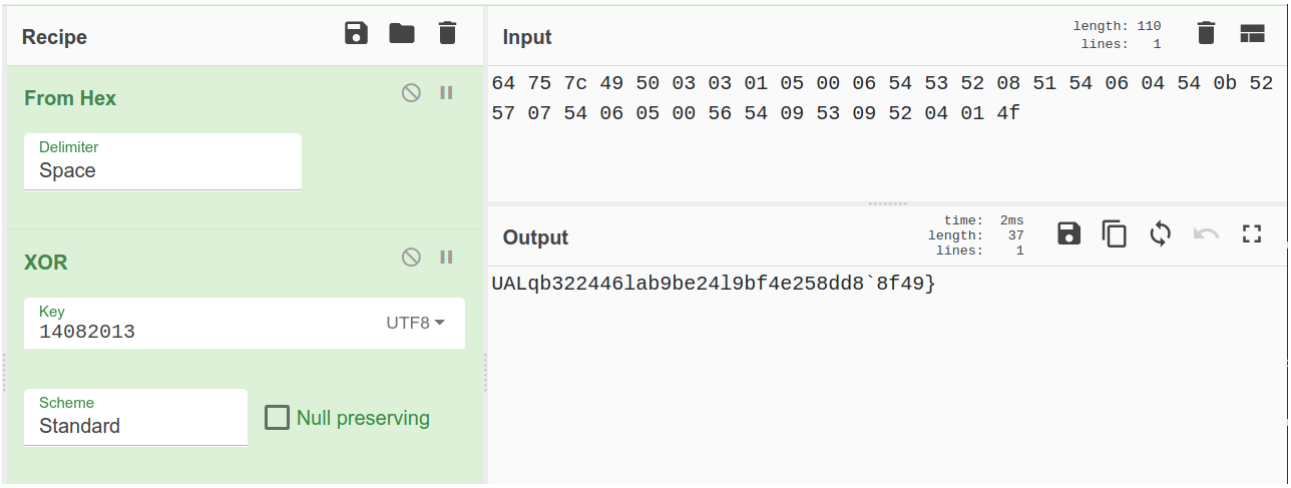
Si convertimos las cifras hexadecimales a texto, aparecen caracteres no imprimibles. Haciendo "rot" con diferentes valores, tampoco obtenemos todos los caracteres imprimibles.

[Omito el resto de pruebas que fui haciendo durante mi investigación con diferentes cifrados/codificaciones y me centro en la que me dio resultado.]

Mi hipótesis es que la clave se trata de una fecha. Busco en https://core.telegram.org/ y en el FAQ sobre la fecha en que nació Telegram:

Q: How old is Telegram?
*Telegram for iOS was launched on **August 14, 2013**. The alpha version of Telegram for Android officially launched on **October 20, 2013**. More and more Telegram clients appear, built by independent developers using Telegram's open platform.*

Al tener un conjunto de cifras hexadecimales, probamos con alguna operación a nivel de bit que produzca una transformación. En este caso probamos con un **XOR** del flag y la fecha de nacimiento de Telegram:



Vemos que nos da algo que se aproxima al flag que buscamos "UAM{md5}"

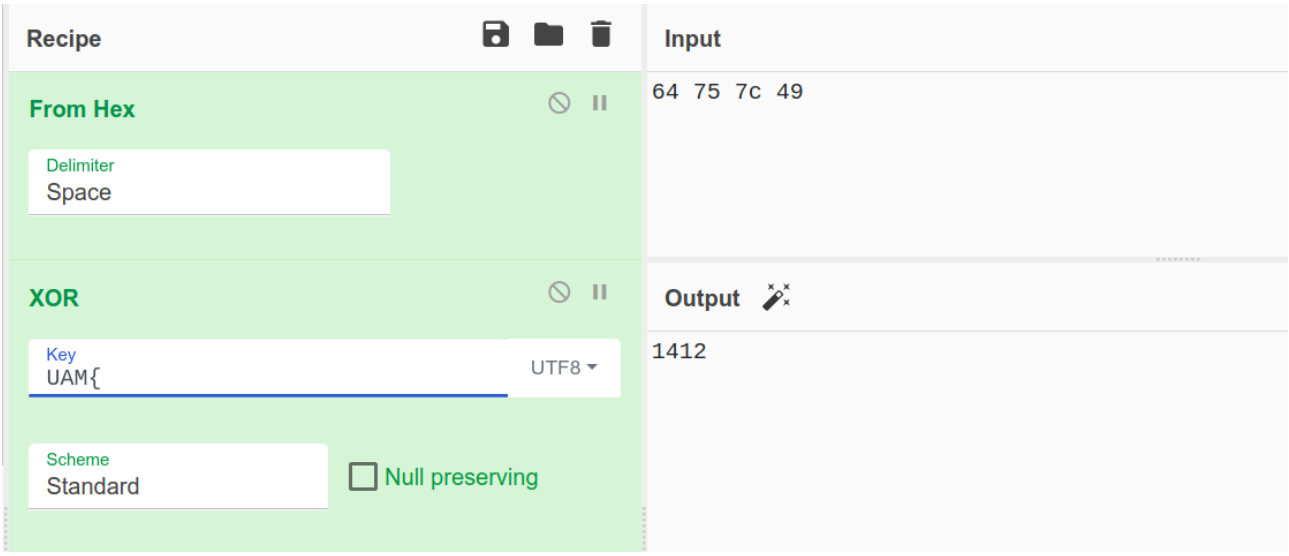
Del cifrado XOR sabemos que:

1. Texto plano XOR Clave = Texto cifrado
2. Texto cifrado XOR Clave = Texto plano
3. Texto plano XOR Texto cifrado = Clave

¿Qué conocemos hasta ahora?

- Texto cifrado: Lo tenemos completo, es nuestro conjunto de valores hexadecimales.
- Clave: La desconocemos. Suponemos que tiene que ser una fecha.
- Texto plano: Lo desconocemos, pero sabemos que empieza por "UAM{"

Por tanto, podemos sacar las cuatro primeras cifras de la clave con las cuatro primeras cifras hexadecimales (texto cifrado) y UAM{ (texto plano) [véase punto 3]:



Ya sabemos con certeza que la clave empieza por :

```
1412
```

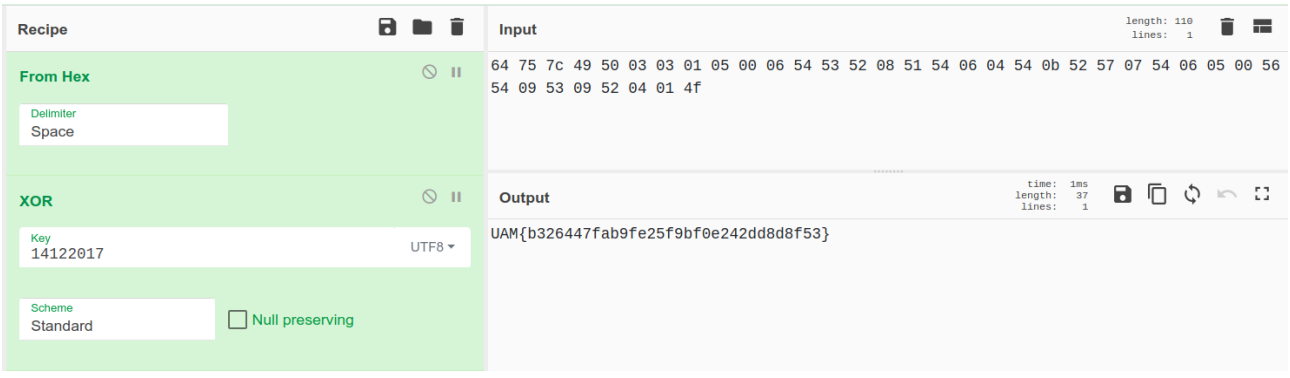
Lo que hacemos es ir probando el año para encontrar el flag correcto. En mi caso estos valores de clave me dieron un flag con el formato correcto:

```
14122002
UAM{b333447fab8ce25f9bg5e242dd9a8f53}

14122012
UAM{b323447fab9ce25f9bf5e242dd8a8f53}

14122014
UAM{b325447fab9ee25f9bf3e242dd8g8f53}

Fecha que dio el flag correcto:
14122017
UAM{b326447fab9fe25f9bf0e242dd8d8f53}
```



Por tanto, el flag es:

```
UAM{b326447fab9fe25f9bf0e242dd8d8f53}
```

Rafa Martos
@elbuenodefali