Encrypted File

Categoría

Crypto/Forensic

Descripción del reto

¡Un ransomware ha convertido mi carpeta de archivos confidenciales en este archivo tan raro! He perdido la fotografía que contenía la flag...
¿Me ayudas a recuperarla?

Pistas

- ¿A qué algoritmo/cifrado le interesa tener bytes nulos sin encriptar?

Autor

Inés Martín - `@ineesdv`

- Twitter
- Linkedin
- Github

Writeup

Deducción del tipo de cifrado

Sobre el método de encriptación no nos dan ningún tipo de información, pero podemos ver que el fichero tiene null bytes.

Si un fichero encriptado contiene null bytes lo más probable es que estos no se hayan encriptado por algún motivo: la clave se filtraría.
Esto nos puede llevar a deducir que se ha utilizado el cifrado **XOR**, pues si aplicásemos un XOR sobre bytes nulos, veríamos la clave en texto plano.

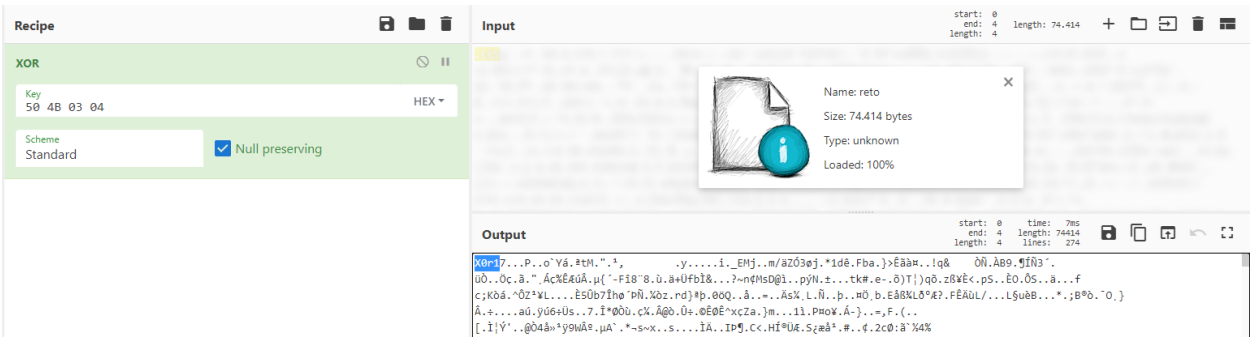
Obteniendo la primera mitad de la clave

Nos hablan de una **carpeta de archivos** encriptada, por lo que sabemos que la única manera de que una carpeta sea un archivo es que esta esté comprimida o *zipeada* (ficheros *.zip*, *.rar*, *.7z*, etc)

Gracias a la **propiedad autoinversa** del cifrado XOR, sabemos que si averiguamos parte del texto plano podremos conseguir la clave o parte de ella: `(A xor K) xor A = K`

Todos los ficheros empiezan por unos bytes de cabecera que determinan el tipo de archivo que es (los denominados ***Magic Bytes***). Concretamente los de un *.zip* son: `50 4B 03 04`, por lo que conocemos parte del texto plano.

> ⚠️ Hay que tener en cuenta que el "ransomware" **ha ignorado los null bytes al hacer el XOR**, pues de lo contrario filtraría la clave de cifrado.
>
> Por ello, al hacer los XOR durante el reto, debemos ignorar los null bytes, o de lo contrario no se realizará correctamente el descifrado.
>
> Por ejemplo, utilizando *CyberChef*, la opción ***Null Preserving*** debe estar activa.



XOR con Magic Bytes

Ahora conocemos los primeros cuatro bytes de la clave: `X0r1`, pero vemos que el archivo no se ha recuperado del todo.

Obteniendo el resto de la clave

Mirando strings, vemos que al final del todo aparece la palabra `flag`. Teniendo en cuenta la estructura de los archivos zip, este nombre corresponde al archivo (o uno de los archivos) contenido dentro del zip.

```
¥.õòSÍ$.<.y.M.˜Ã1«X].ìùF.TÁ)sø.¬...à.o}5v0.Í..nám[`.Q#..S..oU·.Óh[..lí»áÑ.0O.Úp¸.V!5Ó´«.»¿g.
<¸F..#«×@P@ü/.>.îWg}1y¹³..®..¸ªPU¼.£c.ÜV..yjß.
k.ÓúÓ$)Ë:bb..rPD.Y..9.¤S-ø.¶...¹/
ÔÖFhà'..BÛX;..*w.lÿdwD¼=Ãwy¬..JÔ.;.ñsu9P'Ø<É.hIA.óó½.$Yx..APÈì§+AðSÓRÕC.ÓõÀ.V$..ûCÂWÖ.]÷.ök)dT.)`j/b.
(..ÿ'.".mú.fä±1×NUw$Ýû?.ç#Õÿ.C¾n3û°.þ÷
Ô.. }Ãÿ¾Ê. h.º.ÅÒ.-.ßü§..Z'®6÷õûd~.½ÿ.PK*t.\....#.é.CT...y;}..ÏF*.....................flag..i8PK.............b"....
```

Podemos deducir que **la longitud de la clave es múltiplo de 4**, ya que si no no se habría descifrado esa parte.

Si además nos damos cuenta de otro detalle: `flag..i8PK`

PK son los **2 bytes identificadores del ZIP**. Estos aparecen en dos ocasiones: en la cabecera y al final del directorio central (después de que aparezcan el nombre del archivo contenido en el zip y su comentario, que en este caso se ve que es nulo, por lo que PK aparece directamente tras el nombre del archivo).

```
000009a0  28 f0 50 4b 01 02 17 03  14 00 00 00 08 00 1c 7d  |(.PK...........}|
000009b0  4b 35 a6 e1 90 7d 45 00  00 00 4a 00 00 00 05 00  |K5...}E...J.....|
000009c0  0d 00 1c 00 00 00 01 00  00 00 a4 81 00 00 00 00  |................|
000009d0  66 69 6c 65 31 55 54 05  00 03 c7 48 2d 45 55 78  |file1UT....H-EUx|
000009e0  00 00 74 68 69 73 20 69  73 20 61 20 63 6f 6d 6d  |..this is a comm|
000009f0  65 6e 74 20 66 6f 72 20  66 69 6c 65 20 31 50 4b  |ent for file 1PK|
```

Estructura interna de un zip

Por ello, podemos deducir también que la clave tiene concretamente **8 bytes de longitud**, por lo que nos faltan los 4 bytes restantes de la clave.

Como en el enunciado nos han dicho que **la flag está en una fotografía**, y su nombre tiene que estar compuesto por 8 bytes, es decir, **8 caracteres** (es la distancia que hay entre *flag* y *PK*), la extensión del archivo tendrá 3 caracteres.

Las únicas opciones son que el nombre de archivo sea flag.png, flag.jpg...

De nuevo, gracias a la propiedad autoinversa del cifrado XOR, si conocemos el nombre del archivo completo podremos conseguir la clave entera (solamente con su extensión también valdría para conseguir los últimos 4 bytes) .

Haciendo un xor con *flag.png* y fijándonos en esa misma línea, podremos ver que hemos obtenido la clave completa: `X0r1sFun`.

```
tp-æFX¸¢V¡.I!..èit§?.ÁXPè.sçæ%¢ä[¢Í.öC.ìè¼É¼>"æ..¤X1Uá ´Ïu.;ø.«æ..>.ëÁYC.\4.
*...U.õS}...ÿ9'+?.Ü.\.................X0r1sFun..P....?...k...\~....
```

Descifrar el archivo y obtener la flag

Conociendo la clave `X0r1sFun`, podremos obtener el archivo original simplemente realizando la operación XOR. Es importante recordar que se deben mantener los bytes nulos para que se descifre correctamente.

Nos encontraremos con el zip, y en su interior la forografía *flag.png*. Abriéndola podremos encontrar la flag:

Flag

`hackon{x0r_X0R_M4G1c_BYt3z}`