FLAG

Préparé pour : Projet FIC
18 décembre 2014

SOLUTION

Objectif

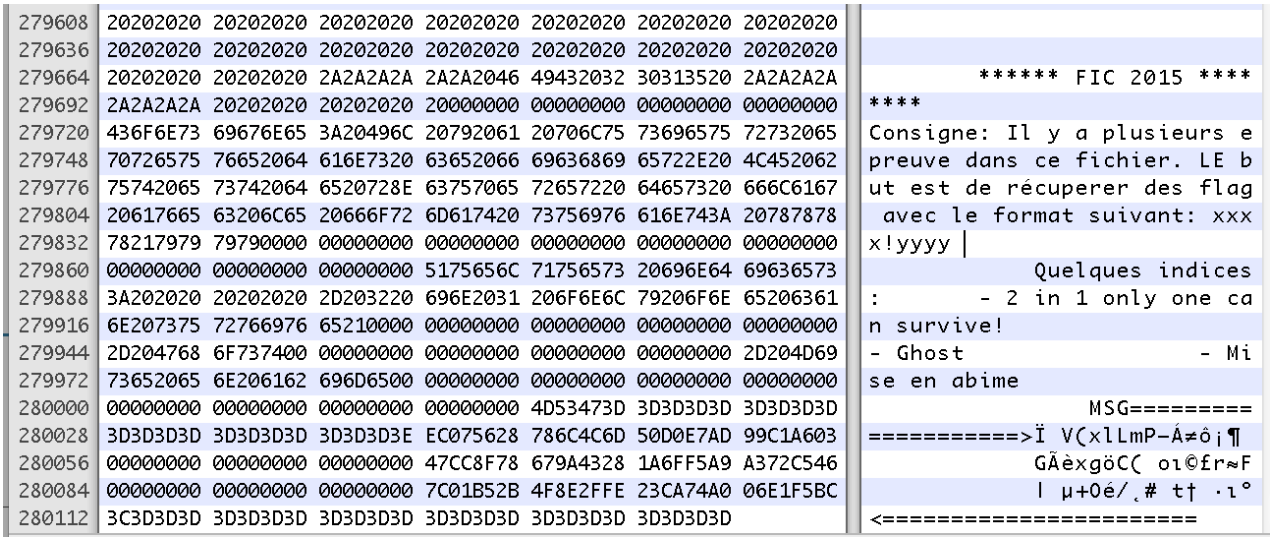
Flag est un ensemble d ‘épreuves dont le but est de retrouver des flags. Chaque épreuve dissimule un flag, ces épreuves reposent principalement sur de la recherche d’information et de la crypto.

Démarrage

On doit tout d’abord déterminer la nature du fichier et ce qu’il faut faire.

Trouver le mode d’emploi

Vu la taille réduite du fichier, un simple drag-n-drop dans un éditeur hexadécimal permet de trouver des chose intéressantes, telles que les consignes.



```
279608 20202020 20202020 20202020 20202020 20202020 20202020 20202020
279636 20202020 20202020 20202020 20202020 20202020 20202020 20202020
279664 20202020 20202020 2A2A2A2A 2A2A2046 49432032 30313520 2A2A2A2A
279692 2A2A2A2A 20202020 20202020 20000000 00000000 00000000 00000000
279720 436F6E73 69676E65 3A20496C 20792061 20706C75 73696575 72732065
279748 70726575 76652064 616E7320 63652066 69636869 65722E20 4C452062
279776 75742065 73742064 6520728E 63757065 72657220 64657320 666C6167
279804 20617665 63206C65 20666F72 6D617420 73756976 616E743A 20787878
279832 78217979 79790000 00000000 00000000 00000000 00000000 00000000
279860 00000000 00000000 00000000 5175656C 71756573 20696E64 69636573
279888 3A202020 20202020 2D203220 696E2031 206F6E6C 79206F6E 65206361
279916 6E207375 72766976 65210000 00000000 00000000 00000000 00000000
279944 2D204768 6F737400 00000000 00000000 00000000 00000000 2D204D69
279972 73652065 6E206162 696D6500 00000000 00000000 00000000 00000000
280000 00000000 00000000 00000000 00000000 4D53473D 3D3D3D3D 3D3D3D3D
280028 3D3D3D3D 3D3D3D3D 3D3D3D3E EC075628 786C4C6D 50D0E7AD 99C1A603
280056 00000000 00000000 00000000 47CC8F78 679A4328 1A6FF5A9 A372C546
280084 00000000 00000000 00000000 7C01B52B 4F8E2FFE 23CA74A0 06E1F5BC
280112 3C3D3D3D 3D3D3D3D 3D3D3D3D 3D3D3D3D 3D3D3D3D 3D3D3D3D

                ****** FIC 2015 ****
****
Consigne: Il y a plusieurs e
preuve dans ce fichier. LE b
ut est de récuperer des flag
 avec le format suivant: xxx
x!yyyy
            Quelques indices
:       - 2 in 1 only one ca
n survive!
- Ghost                  - Mi
se en abime
                  MSG=========
==========>Ï V(xlLmP-Á≠ô¡¶
            GÃèxgöC( oı©£r≈F
            | µ+Oé/¸# t† ·ı°
<=======================
```

On notera:

- qu’il faut trouver un flag avec sous la forme de .{4,5}(-|!).{4,5}.
- une série d’indices
- un message

Trouver le bon format

Indice: 2 in 1, only one can survive

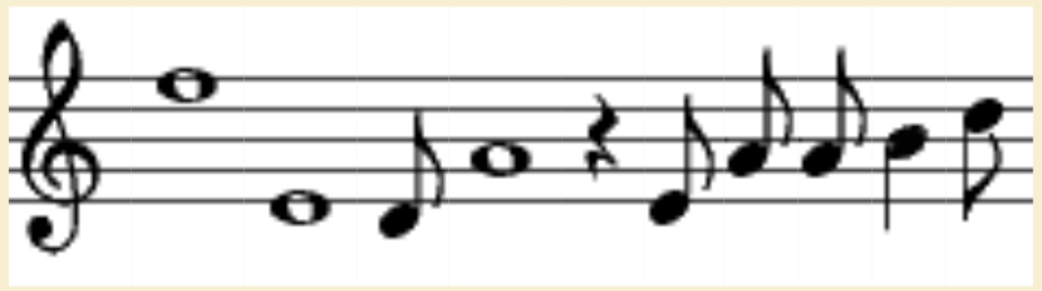
En cherchant avec un éditeur hexadécimal ou avec la commande file, on s'aperçoit que le fichier est une image ISO. Une fois renommé en iso, l'image n'est pas reconnu par la plupart des logiciels excepté la commande mount.

Si on regarde plus précisément dans le fichier, on retrouve la signature MZ qui correspond à un PE. En supprimant le PE, l'image iso se monte normalement

L'image iso révèle une image au format png mais ne contient rien de plus en apparence, l'accès au reste du contenu étant rendu difficile.

En regardant l'image, on pense directement sténographie. Cette image est la représentation du flag chiffré en notes de musique.

Indice: Croches, noires



Une fois déchiffré, on trouve le premier flag: 91A4-BEETH

A = Ré 1, B = Mi 1, ..., G = Do 2, H = Ré 2, ..., M = si 2, N = Do 1, ... etc. 0 = Ré 1, 1 = Mi 1.

On utilise d'abord les croches, puis les noires. Les chiffres sont représentés par des blanches ou des rondes. Il est possible d'inverser les types de notes. Suivant l'ordre utilisé, on obtient un message different:

- 91A4- Croches, noires
- 91O4- Noires, croches
- 9104- Blanches, noires
- 9114- Noires, blanches

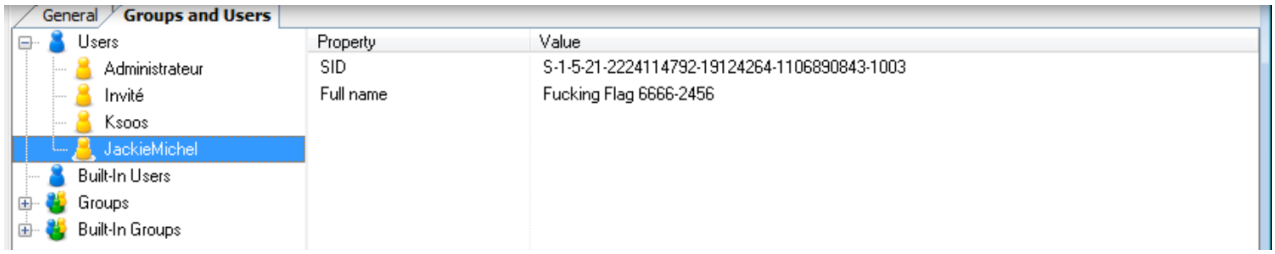
Trouver le contenu caché:

Indice correspondant: Ghost

En utilisant un logiciel comme IsoBuster ou avec une recherche manuelle, l'iso révèle un fichier compressé RAR et un dossier FIC

Dossier compressé:

Ce fichier est une base SAM, en utilisant MITEC WINDOW REGISTRY ou en recherchant manuellement, on trouve un utilisateur JackieMichel et un flag 6666-2456



Dossier FIC

Le dossier FIC contient un pdf,celui-ci ne contient rien d'utile.

Une analyse manuelle ou via pdfid.py montre des données (au format Base64) à la fin du fichier :

```
$> python pdfid.py -e pdf.pdf!
[...]
After last %%EOF  23437
[...]
Total entropy:  112144 bytes)
```

Récupération des données Base64 :

```
$> dd bs=1 skip=88707 if=CheatSheet.pdf of=data.b64
$> base64 -d data.b64 > data.bin
$> file data.bin
=> PDF document, version 1.4
$> mv data.bin data.pdf!
=> Le PDF  contient en texte non compressé le mot de passe du level: 3917!l4s1
```

PDF embarqué

Indice: mise en abime

Si on fouille un peu dans le pdf,on s'aperçoit qu'il embarque un autre pdf

En fouillant l'arborescence, on retrouve un flag 8142!val7.pdf

```
1 0 obj
<<
        /Type /Catalog
        /Outlines 2 0 R
        /Pages 3 0 R
        /Names <<
                /EmbeddedFiles <<
                        /Names [ (8142!val7.pdf) 7 0 R ]
                >>
        >>
>>
endobj
```

Executable

Au début de l'épreuve, nous avons découvert un PE pour pouvoir monter l'image iso. On peut l'extraire une fois de plus en supprimant tout ce qui est situé au-dessus de la signature "MZ", et en sauvegardant le résultat dans un fichier avec l'extension .exe

Celui-ci, une fois exécuté, donne les instructions et indices pour faire cette épreuve.

```
********** FIC 2015 **********
Il y a un message dans un fichier.
Voici la clee publique: 8F 3B 5E 1B 54 F0 00 48 9B 91 B7 66 10 2E 26 91
Un indice pour la route:  0 X 17
```

On y découvre une clé publique et un indice: 0 X 17

17 converti en binaire donne 10001 ce qui donne 0 X 10001 => format hexadécimal

Le fait de voir 0x10001 doit titiller un peu ceux qui ont fait du RSA car c'est une valeur d'exponentiation très fréquemment employée. L'épreuve consiste donc à déchiffrer des chaînes de caractères chiffrées en RSA.

Afin de s'en assurer, lançons *RSA-Tool 2 by Te!* puis entrons les valeurs e et n dans les champs correspondants, puis vérifions la taille de la clef publique en cliquant sur "Exact size" : celle-ci fait bien 128 bits, donc ce sera rapide !!

Le message a déchiffrer se trouve dans le fichier de départ

```
280000 00000000 00000000 00000000 00000000 4D53473D 3D3D3D3D 3D3D3D3D                    MSG=========
280028 3D3D3D3D 3D3D3D3D 3D3D3D3E EC075628 786C4C6D 50D0E7AD 99C1A603 ============>Ï V(xlLmP-Á≠ô¡¶
280056 00000000 00000000 00000000 47CC8F78 679A4328 1A6FF5A9 A372C546                GÃèxgöC( o1©£r≈F
280084 00000000 00000000 00000000 7C01B52B 4F8E2FFE 23CA74A0 06E1F5BC                | µ+Oé/¸# t† ·ı°
280112 3C3D3D3D 3D3D3D3D 3D3D3D3D 3D3D3D3D 3D3D3D3D 3D3D3D3D <=======================
```

=> EC075628786C4C6D50D0E7AD99C1A63

47CC8F78679A43281A6FF5A9A372C546

7C01B52B4F8E2FFE23CA74A006E1F5BC

Une fois déchiffré, on trouve le dernier flag 2987!2rsa