No Abre

Se nos entrega un archivo llamado no_abree.png. Aunque tiene extensión PNG, ningún visor de imágenes logra abrirlo. El objetivo es descubrir qué ocurre con el archivo, repararlo y visualizar su contenido para obtener la flag.

Cuando una imagen no abre, lo primero no es descartarla, sino verificar si el archivo es realmente lo que dice ser.

Para ello, se revisa el archivo a nivel binario con un visor hexadecimal:

```
└─$ hexdump -C no_abree.png | head -n 20

00000000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52  |.PNG........IHDR|
00000010  00 00 03 80 00 00 02 71  08 02 00 00 00 00 00 00  |.......q........|
00000020  00 00 00 00 03 73 42 49  54 08 08 08 db e1 4f e0  |.....sBIT.....O.|
00000030  00 00 20 00 49 44 41 54  78 9c ed dd 7f 70 9c f7  |..  .IDATx....p..|
00000040  7d 1f f8 5d 62 77 09 80  04 44 00 24 08 92 58 81  |}..]bw...D.$..X.|
00000050  b2 44 d9 57 aa e9 c9 1e  eb 04 e7 72 76 27 37 b5  |.D.W.......rv'7.|
00000060  27 6d 74 97 c6 44 7d 99  64 26 67 ab 4d 2f e9 b8  |'mt..D}.d&g.M/..|
00000070  43 aa b9 89 5a df d9 6e  3d e3 5e 33 22 2f be 3a  |C...Z..n=.^3"/.:|
00000080  57 37 4a ea 99 7a 32 3d  4a 69 ae be c6 13 67 c6  |W7J..z2=Ji....g.|
00000090  17 bb 49 0d 8d 32 67 37  ae e4 58 a2 65 93 5a 50  |..I..2g7..X.e.ZP|
000000a0  e2 2f 80 14 40 e2 c7 ee  02 7b 7f ac b5 78 b0 00  |./..@....{...x..|
000000b0  c1 05 76 f7 fb 3c 0b bc  5e 7f 2d 40 70 f1 e0 bb  |..v..<..^.-@p...|
000000c0  bb cf f3 7e be 3f 3e df  f4 c0 c0 40 0a 00 00 42  |...~.?>....@...B|
000000d0  d9 13 f7 01 00 00 b0 bb  08 a0 00 00 04 25 80 02  |.............%..|
000000e0  00 10 94 00 0a 00 40 50  02 28 00 00 41 09 a0 00  |......@P.(..A...|
000000f0  00 04 25 80 02 00 10 94  00 0a 00 40 50 02 28 00  |..%........@P.(.|
00000100  00 41 09 a0 00 00 04 25  80 02 00 10 94 00 0a 00  |.A.....%........|
00000110  40 50 02 28 00 00 41 09  a0 00 00 04 25 80 02 00  |@P.(..A.....%...|
00000120  10 94 00 0a 00 40 50 02  28 00 00 41 09 a0 00 00  |.....@P.(..A....|
00000130  04 25 80 02 00 10 94 00  0a 00 40 50 02 28 00 00  |.%........@P.(..|
```

Al analizar los primeros bytes se observa:

89 50 4E 47 0D 0A 1A 0A

Estos bytes corresponden a la firma oficial de un archivo PNG, lo que confirma que el archivo sí es una imagen válida.

Los archivos PNG están formados por bloques internos llamados chunks.
Cada uno de estos bloques termina con un valor especial llamado CRC, que funciona como una “huella” para verificar que los datos no fueron alterados.

Si el CRC de un bloque es incorrecto el contenido puede estar bien, pero el visor de imágenes rechaza el archivo.

El archivo existe, pero el visor no confía en él.

Al revisar el archivo con un visor hexadecimal, se identifica el primer bloque importante de la imagen, llamado IHDR.
Este bloque contiene información básica como el ancho, alto y tipo de imagen, los cuales son correctos.

Cada bloque de un archivo PNG termina con 4 bytes finales, llamados CRC, que sirven para verificar la integridad del bloque.

En este archivo, los 4 bytes del CRC del bloque IHDR se encuentran al final de dicho bloque y tienen el siguiente valor:

00 00 00 00

Este valor es incorrecto para el contenido del bloque IHDR.

Una vez identificado que el CRC del bloque IHDR es incorrecto, se procede a reemplazar únicamente esos 4 bytes, sin modificar ningún otro dato del archivo.

El valor correcto del CRC para el bloque IHDR es:

9c ef 4b 08
El valor 9c ef 4b 08 corresponde al CRC-32 correcto del bloque IHDR.
Este valor se obtiene al calcular el CRC sobre el tipo del bloque (IHDR) y sus datos.
Al no coincidir el CRC original con el valor esperado, el visor rechaza la imagen.

49 48 44 52 ← "IHDR"

00 00 03 80 ← ancho

00 00 02 71 ← alto

08 ← bit depth

02 ← color type

00 ← compresión

00 ← filtro

00 ← interlace

En hexadecimal continuo:

4948445200000380000002710802000000

Calculamos el crc

```python
import zlib

data = bytes.fromhex("49484452000003800000027108020000000")
crc = zlib.crc32(data) & 0xffffffff
print(hex(crc))
```

```
└─$ python3 solucion.py
0x9cef4b08
```

Arreglamos el crc con:

printf '\x9c\xef\x4b\x08' | dd of=no_abree.png bs=1 seek=29 count=4 conv=notrunc

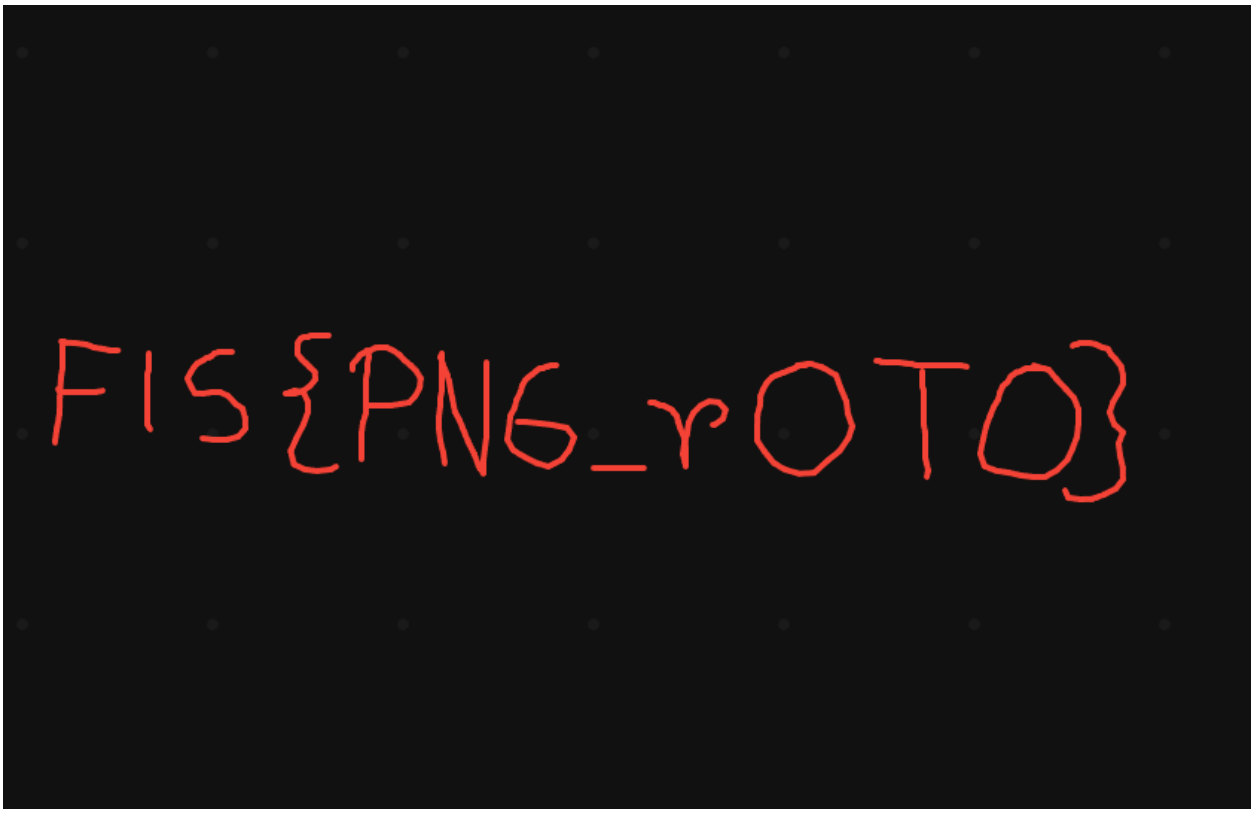
printf '\x9c\xef\x4b\x08'
Genera los 4 bytes correctos del CRC.

dd
Escribe datos directamente en el archivo, byte por byte.

seek=29
Indica que los bytes se escriban en la posición exacta donde empieza el CRC del IHDR.

count=4
Escribe solo 4 bytes.

conv=notrunc
Evita que el archivo se corte o se dañe.


FIS{PNG_rOTO}