

CÁTEDRA DE CIBERSEGURIDAD CIBERUGR, INCIBE-UGR

Nombre	Protected File
Categoría	MISC (CRIPTO + FORENSE)
Dificultad	MEDIA
Puntos	300

DESCRIPCIÓN DEL RETO

Un cliente nos ha enviado un archivo protegido con contraseña, pero no nos han proporcionado la clave para ver el contenido de este. ¿Serás capaz de recuperar el contenido que se encuentra dentro del archivo?

WRITEUP

1. Inicialmente, al leer la descripción del reto se nos proporciona un fichero adjunto. Este fichero se trata de un zip con contraseña.
2. Para ello, vamos a intentar conseguir la contraseña a través de ataques de fuerza bruta, usando “zip2john” para sacar el hash del zip y “john” para crackearlo.
3. Sacamos el hash del zip usando zip2john.

```
zip2john Protected_File.zip > hash.txt
```

```
└─$ zip2john Protected_File.zip > hash.txt
ver 2.0 efh 5455 efh 7875 Protected_File.zip/file_transfer_protoco
l.pcapng PKZIP Encr: TS_chk, cmplen=43574, decmplen=115656, crc=17
25AD31 ts=5A2B cs=5a2b type=8
```

4. Usamos la herramienta “John the Ripper” y el diccionario “rockyou.txt” para crackear la contraseña de este zip.

```
john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

```

└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
!!!secret!!! (Protected_File.zip/file_transfer_protocol.pcapng)
1g 0:00:00:00 DONE (2024-04-24 17:35) 1.923g/s 27583Kp/s 27583Kc/s
27583KC/s "2parrow" .. *7;Vamos!
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

- Ahora que tenemos la contraseña “!!!secret!!!” vamos a descomprimir el archivo. **NOTA:** Es necesario escapar los caracteres especiales como en este caso son todas las exclamaciones.

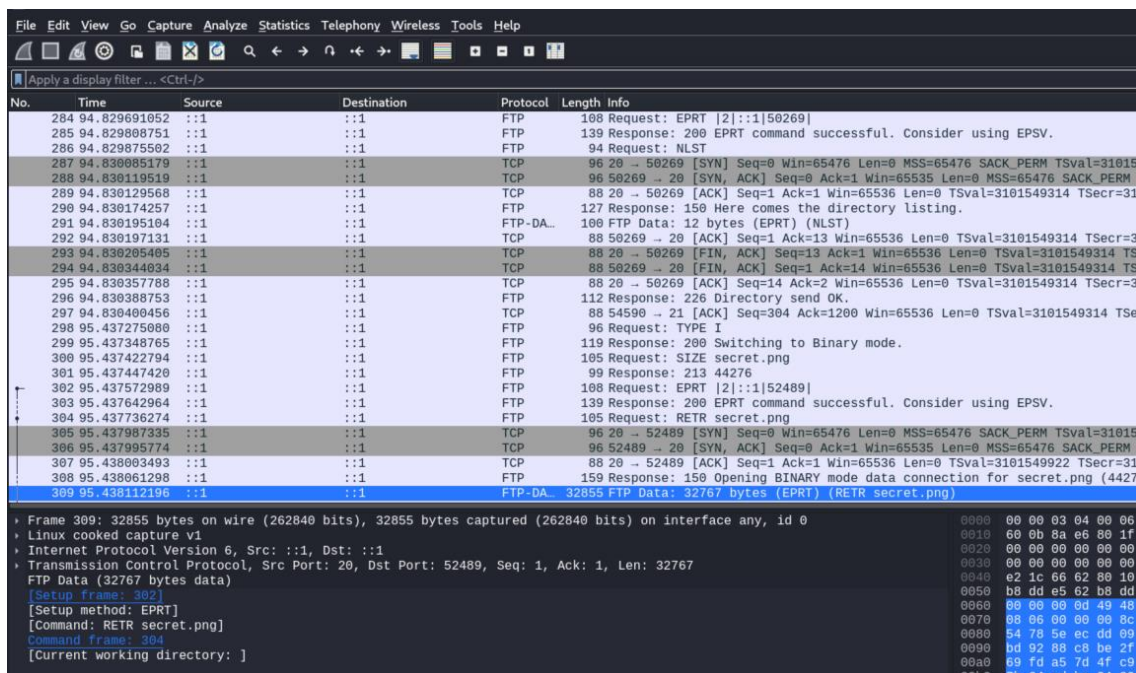
```
unzip -P \\!\\!secret\\!\\! Protected_File.zip
```

```

└─$ unzip -P \\!\\!secret\\!\\! Protected_File.zip
Archive: Protected_File.zip
  inflating: file_transfer_protocol.pcapng

```

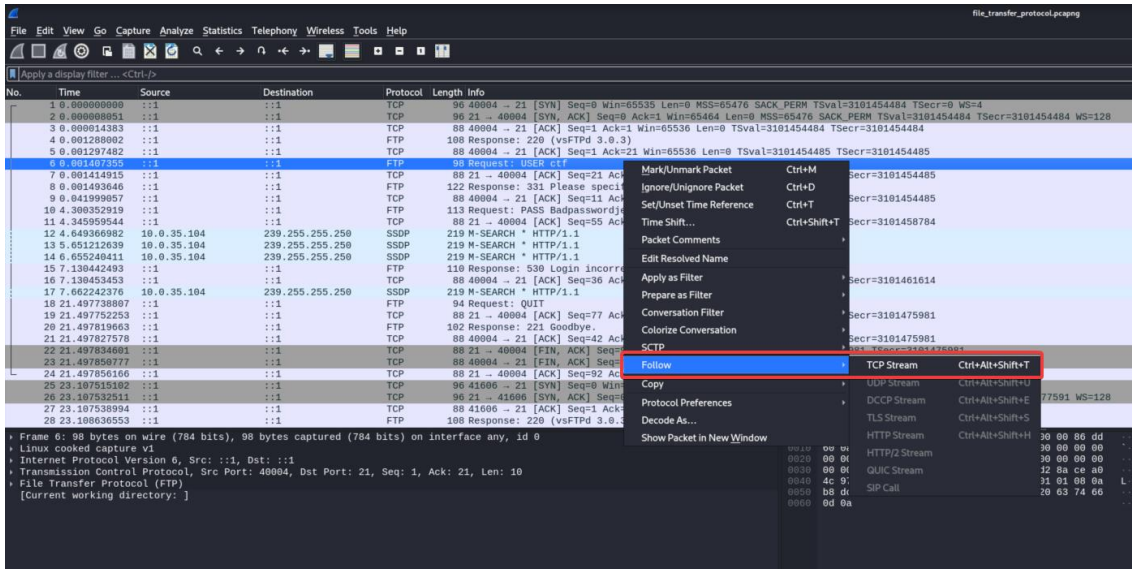
- Una vez hemos descomprimido el zip, vamos a ver qué es lo que contiene dentro.
- En este caso vemos un archivo pcap, el cual se trata de una captura de tráfico.
- Abrimos el pcap usando “Wireshark”.



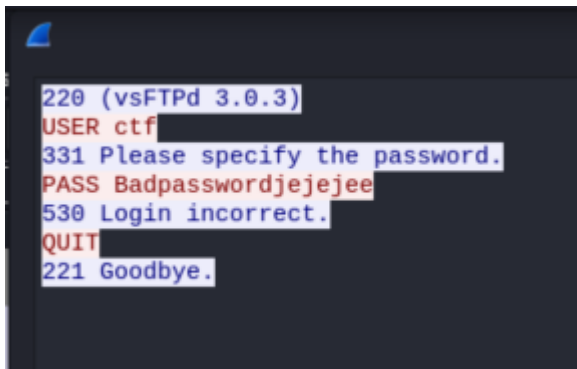
The image shows a Wireshark interface with a packet list table. The selected packet (No. 389) is an FTP-DA packet containing 32767 bytes of data. The packet details pane shows the command 'RETR secret.png' and the current working directory.

No.	Time	Source	Destination	Protocol	Length	Info
284	94.829691052	:::1	:::1	FTP	108	Request: EPRT [2]::1[50269]
285	94.829808751	:::1	:::1	FTP	139	Response: 200 EPRT command successful. Consider using EPSV.
286	94.829875502	:::1	:::1	FTP	94	Request: NLST
287	94.830085179	:::1	:::1	TCP	96	20 → 50269 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM TSval=3101549314 TSecr=3101549314
288	94.830119519	:::1	:::1	TCP	96	50269 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65476 SACK_PERM TSval=3101549314 TSecr=3101549314
289	94.830129568	:::1	:::1	TCP	80	20 → 50269 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3101549314 TSecr=3101549314
290	94.830174257	:::1	:::1	FTP	127	Response: 150 Here comes the directory listing.
291	94.830195104	:::1	:::1	FTP-DA..	100	FTP Data: 12 bytes (EPRT) (NLST)
292	94.830197131	:::1	:::1	TCP	88	50269 → 20 [ACK] Seq=1 Ack=13 Win=65536 Len=0 TSval=3101549314 TSecr=3101549314
293	94.830205405	:::1	:::1	TCP	88	20 → 50269 [FIN, ACK] Seq=13 Ack=1 Win=65536 Len=0 TSval=3101549314 TSecr=3101549314
294	94.830344034	:::1	:::1	TCP	88	50269 → 20 [FIN, ACK] Seq=14 Ack=14 Win=65536 Len=0 TSval=3101549314 TSecr=3101549314
295	94.830357788	:::1	:::1	TCP	88	20 → 50269 [ACK] Seq=14 Ack=2 Win=65536 Len=0 TSval=3101549314 TSecr=3101549314
296	94.830388753	:::1	:::1	FTP	112	Response: 226 Directory send OK.
297	94.830400456	:::1	:::1	TCP	88	54590 → 21 [ACK] Seq=304 Ack=1200 Win=65536 Len=0 TSval=3101549314 TSecr=3101549314
298	95.437275080	:::1	:::1	FTP	96	Request: TYPE I
299	95.437348765	:::1	:::1	FTP	119	Response: 200 Switching to Binary mode.
300	95.437422794	:::1	:::1	FTP	105	Request: SIZE secret.png
301	95.437447420	:::1	:::1	FTP	99	Response: 213 44276
302	95.437572989	:::1	:::1	FTP	108	Request: EPRT [2]::1[52489]
303	95.437642964	:::1	:::1	FTP	139	Response: 200 EPRT command successful. Consider using EPSV.
304	95.437736274	:::1	:::1	FTP	105	Request: RETR secret.png
305	95.437987335	:::1	:::1	TCP	96	20 → 52489 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM TSval=3101549314 TSecr=3101549314
306	95.437995774	:::1	:::1	TCP	86	52489 → 20 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65476 SACK_PERM TSval=3101549314 TSecr=3101549314
307	95.438083493	:::1	:::1	TCP	88	20 → 52489 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3101549922 TSecr=3101549922
308	95.438061298	:::1	:::1	FTP	159	Response: 150 Opening BINARY mode data connection for secret.png (44276 bytes data)
309	95.438112196	:::1	:::1	FTP-DA..	32855	FTP data: 32767 bytes (EPRT) (RETR secret.png)

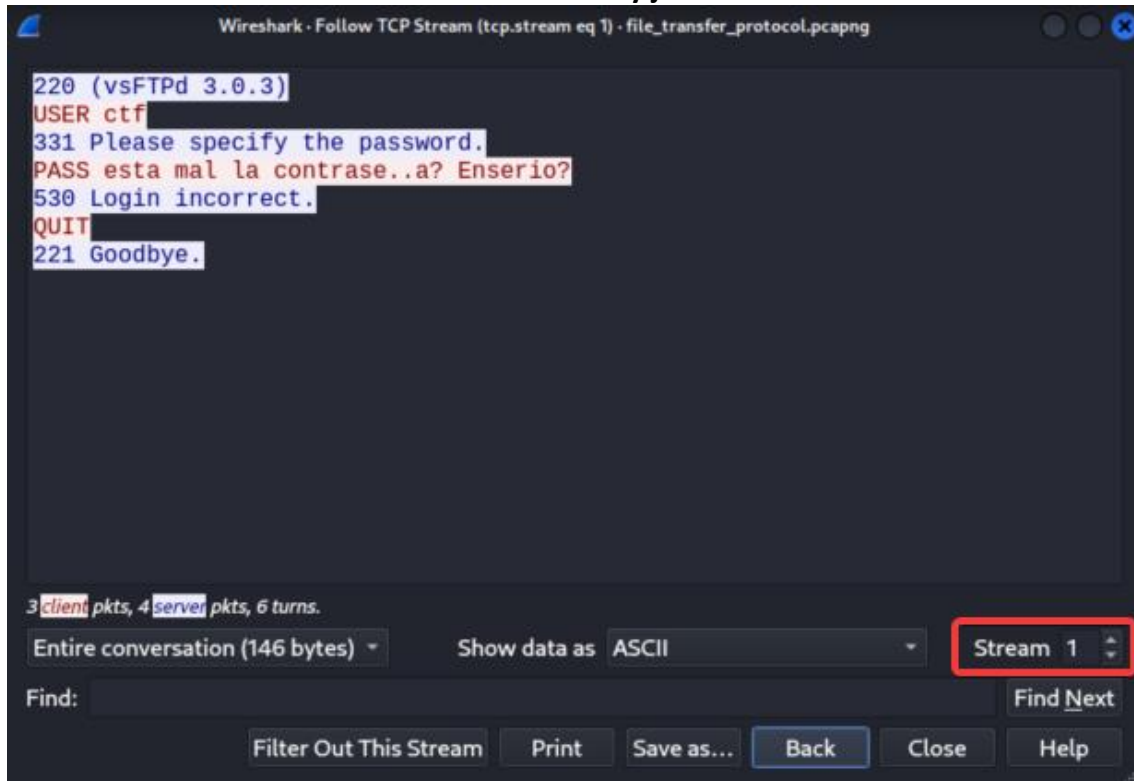
9. Al abrir la captura de tráfico, se ve bastante tráfico FTP, por lo tanto, es necesario hacerle un análisis para identificar que es lo que está pasando en esta comunicación. **NOTA:** Para analizarlo, podemos seguir el flujo TCP.



10. En el primer paquete, se ve como el usuario "ctf" se ha intentado autenticar con una contraseña invalida.



11. Para seguir viendo las comunicaciones podemos ir aumentando el valor de "stream".

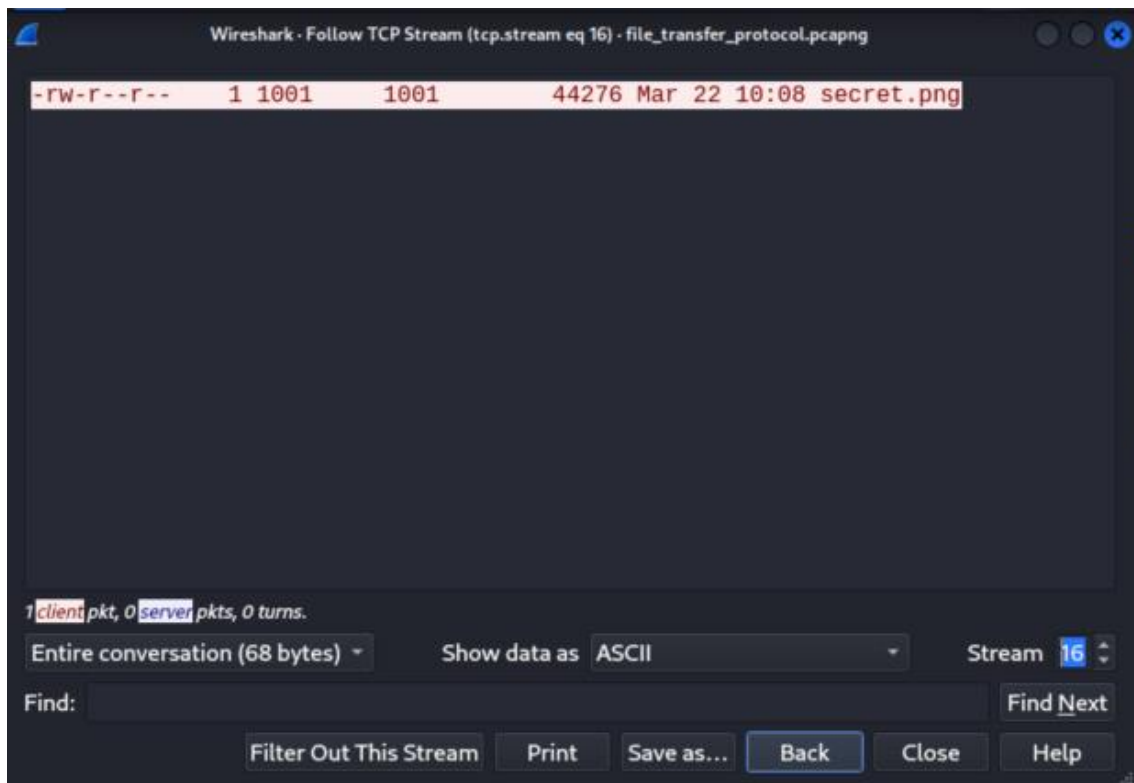


```
Wireshark · Follow TCP Stream (tcp.stream eq 1) · file_transfer_protocol.pcapng

220 (vsFTPd 3.0.3)
USER ctf
331 Please specify the password.
PASS esta mal la contrase..a? Enserio?
530 Login incorrect.
QUIT
221 Goodbye.

3 client pkts, 4 server pkts, 6 turns.
Entire conversation (146 bytes) Show data as ASCII Stream 1
Find: Find Next
Filter Out This Stream Print Save as... Back Close Help
```

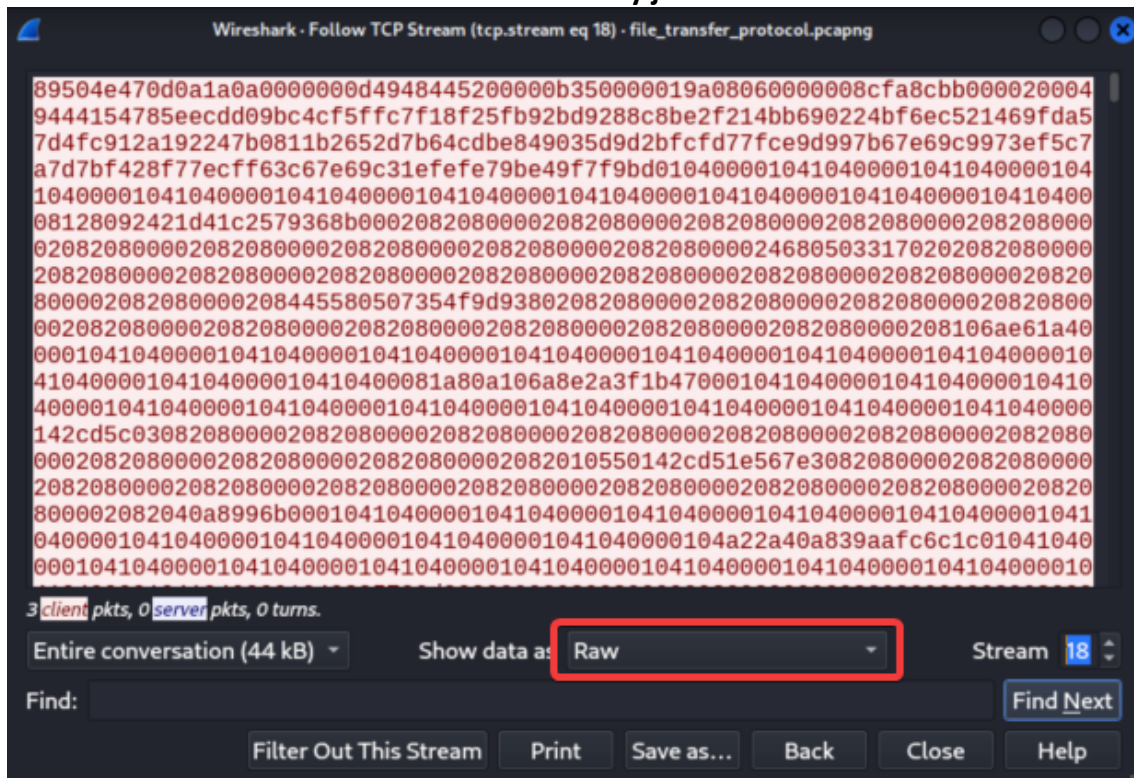
12. En el flujo 16 podemos observar como el usuario ha listado un directorio en el que se encuentra un PNG con el nombre de secret.png.



```
Wireshark · Follow TCP Stream (tcp.stream eq 16) · file_transfer_protocol.pcapng

-rw-r--r-- 1 1001 1001 44276 Mar 22 10:08 secret.png

1 client pkt, 0 server pkts, 0 turns.
Entire conversation (68 bytes) Show data as ASCII Stream 16
Find: Find Next
Filter Out This Stream Print Save as... Back Close Help
```

15. Comparamos los primeros bytes de la imagen con los *magic bytes* de un PNG, en este caso coinciden, por lo que ya podemos asumir que esto es un PNG.

Additional scanning signature analysis

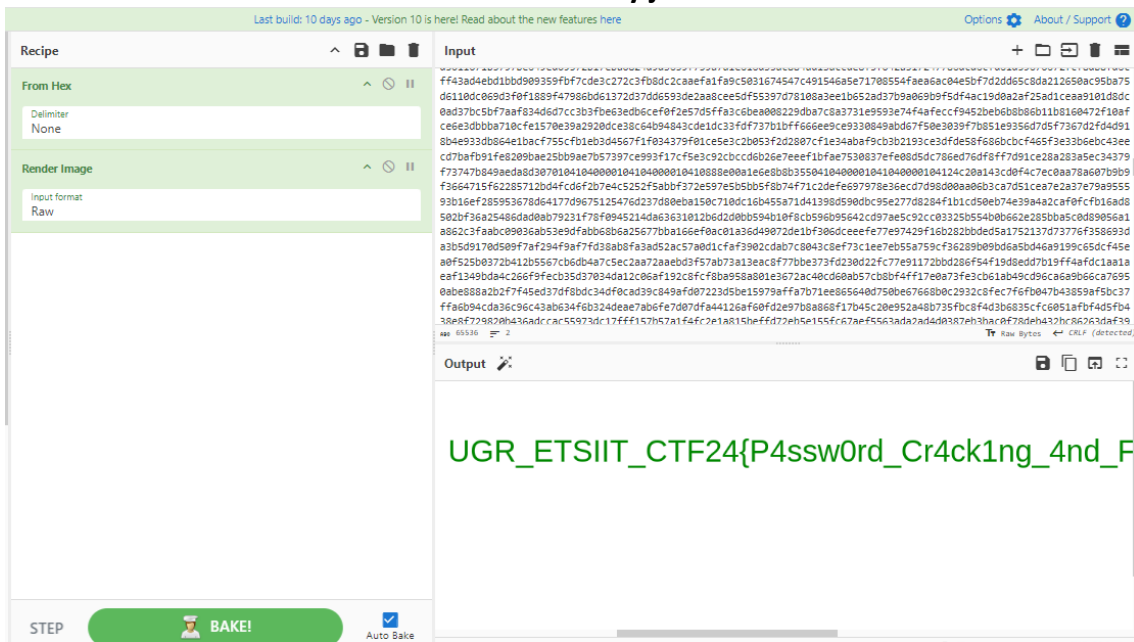
Possible: [PNG file, sig: 89504E470D0A1A0A] Count: 1

File hex characters

```
[00000000] 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 .PNG.....IHDR
[00000016] 00 00 00 20 00 00 00 20 01 00 00 00 00 5B 01 47 .....[.G
[00000032] 59 00 00 00 04 67 41 4D 41 00 01 86 A0 31 E8 96 Y....gAMA...1..
[00000048] 5F 00 00 00 5B 49 44 41 54 78 9C 2D CC B1 09 03 _...[IDATx.-...
[00000064] 30 0C 05 D1 EB D2 04 B2 4A 20 0B 7A 34 6F 90 15 0.....J..z4o..
[00000080] 3C 82 C1 8D 0A 61 45 07 51 F1 E0 8A 2F AA EA D2 <....aE.Q.../...
[00000096] A4 84 6C CE A9 25 53 06 E7 53 34 57 12 E2 11 B2 ..l..%S..S4W....
[00000112] 21 BF 4B 26 3D 1B 42 73 25 25 5E 8B DA B2 9E 6F !.K&=..Bs%^.....o
[00000128] 6A CA 30 69 2E 9D 29 61 6E E9 6F 30 65 F0 BF 1F j.0i..)an.o0e...
[00000144] 10 87 49 2F D0 2F 14 C9 00 00 00 00 49 45 4E 44 ..I/./.....IEND
[00000160] AE 42 60 82
```

-- Sector 1 -- Assuming 512 Bytes ---

16. Por último, podemos ver la imagen usando CyberChef (<https://gchq.github.io/CyberChef/>), aplicando el "from Hex" y como sabemos que es una imagen, añadimos el "Render Image", para finalmente poder visualizar la imagen que contiene la *flag*.



The screenshot shows a web interface for a CTF challenge. On the left, there is a 'Recipe' panel with 'From Hex' selected and 'Delimitter' set to 'None'. The 'Render Image' panel shows 'Input format: Raw'. The main 'Input' field contains a long string of hexadecimal characters. The 'Output' field displays the decoded result: `UGR_ETSIIIT_CTF24{P4ssw0rd_Cr4ck1ng_4nd_F`. At the bottom, there is a 'BAKE!' button and an 'Auto Bake' checkbox.