

Hidden Messages

EL reto nos entrega una base de datos SQLite. Entonces deberíamos acceder a esta para obtener información útil.

```
└$ sqlite3 secretKEY.db
SQLite version 3.46.1 2024-08-13 09:16:08
Enter ".help" for usage hints.
sqlite> .tables
media_files  messages    users
sqlite> select * from messages;
2999|Log interno: referencia clave 63 79 62 65 72 6D 69 6E 64 73 31 33
3001|Re: ref #3001 - nota: JTAXHiEZG11QHm
3002|Re: ref #3002 - nota: 53Vw1WOicDBV4
3003|Re: ref #3003 - nota: HGAJXPEtSV0cQ
3101|docref=JTAXHiEZG11QHm
3102|docref=53Vw1WOicDBV4
3103|docref=HGAJXPEtSV0cQ
sqlite> █
```

En la tabla messages encontramos varios mensajes y un log interno que dice que es una referencia clave. Notamos que esta en hexadecimal, si la pasamos a texto nos podría dar información.

De A

Hexadecimal Texto

Abrir archivo Muestra

Pegar números de código hexadecimal o soltar archivo

63 79 62 65 72 6D 69 6E 64 73 31 33

Codificación de caracteres

ASCII

Convertir Restablecer Intercambiar

cyberminds13

Al unir estos fragmentos en el orden correcto, se obtiene una única cadena Base64 completa, lo que indica que los datos originales se encuentran cifrados y deben ser decodificados antes de su análisis.

Este código toma una cadena codificada en Base64, la decodifica para obtener datos binarios cifrados y luego los descifra aplicando una operación XOR con la clave cyberminds13, repitiendo el clave byte a byte sobre todo el contenido; como resultado, se recuperan los bytes originales del mensaje oculto (la flag).

```
import base64

b64 = "JTAxHiEZG1lQHm53Vw1WOicDBV4HGAJXPEtSV0cQ"
data = base64.b64decode(b64)

key = "cyberminds13"
```

```
def xor_with_key(data, key):
key = key.encode()
return bytes(
b ^ key[i % len(key)]
for i, b in enumerate(data)
)

flag_bytes = xor_with_key(data, key)
print(flag_bytes)
print(flag_bytes.decode())
```

```
[vfric@vfric ~]~/.../CTF-cybermind
$ python3 solucion.py
b'F1S{Str74m_D4t4_Unl0ck3d_2025}'\nF1S{Str74m_D4t4_Unl0ck3d_2025}
```