

## Writeups de la compétition LoveCTF, 1<sup>ère</sup> édition

Team : W3hunters

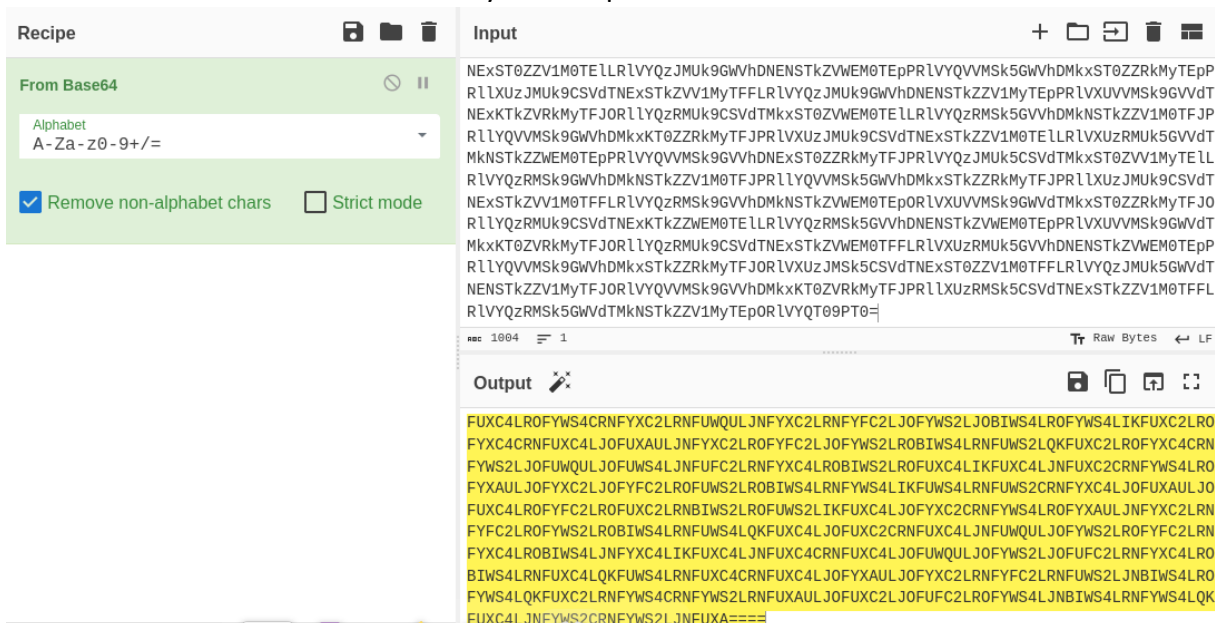
Catégorie du challenge : Crypto

Titre du challenge : Mise à l'épreuve

Une vraie mise à l'épreuve ce challenge. Un fichier txt a été soumis à notre réflexion.

```
([~/myHome/CTFs/LoveCTF/mise_a_l_epreuve]
$ cat chall.txt
RlVYQzRMUK9GWVdTNENStkZZWEMyTFJ0RlVXUVVMSk5GWVhDMkxStkZZRkMyTEpPRllXUzJMSk9CSvdTNEx
ST0ZZV1M0TEllRlVYQzJMUK9GWVhDNENStkZVWEM0TEpPRlVYQVMSk5GWVhDMkxST0ZZRkMyTEpPRllXUz
JMUK9CSvdTNExStkZVV1MyTFFLrVYQzJMUK9GWVhDNENStkZZV1MyTEpPRlVXUVVMSk9GVVdTNExKtkZVR
kMyTFJ0RllYQzRMUK9CSvdTMkxST0ZVWEM0TEllRlVYQzRMSk5GVVhDMkNStkZZV1M0TFJPRllYQVMSk9G
WVhDMkxKT0ZZRkMyTFJPRlVXUzJMUK9CSvdTNExStkZZV1M0TEllRlVXUzRMUK5GVVdTMkNStkZZWEM0TEp
PRlVYQVMSk9GVVhDNExST0ZZRkMyTFJPRlVYQzJMUK5CSvdTMkxST0ZVV1MyTEllRlVYQzRMSk9GVVhDMk
NStkZZV1M0TFJPRllYQVMSk5GWVhDMkxStkZZRkMyTFJPRllXUzJMUK9CSvdTNExStkZVV1M0TFFLrVYQ
zRMSk9GVVhDMkNStkZVWEM0TEpORlVXUVVMSk9GVVdTMkxST0ZZRkMyTFJ0RllYQzRMUK9CSvdTNExKtkZZ
WEM0TEllRlVYQzRMSk5GVVhDNENStkZVWEM0TEpPRlVXUVVMSk9GVVdTMkxKT0ZVRkMyTFJ0RllYQzRMUK9
CSvdTNExStkZVWEM0TFFLrVXUzRMUK5GVVhDNENStkZVWEM0TEpPRllYQVMSk9GVVhDMkxStkZZRkMyTF
J0RlVXUzJMSk5CSvdTNExST0ZZV1M0TFFLrVYQzJMUK5GVVdTNENStkZZV1MyTFJ0RlVYQVMSk9GVVhDM
kxKT0ZVRkMyTFJPRllXUzRMSk5CSvdTNExStkZZV1M0TFFLrVYQzRMSk5GVVdTMkNStkZZV1MyTEpORlVY
QT09PT0=
```

Il s'agit de toute évidence d'une séquence encodée en base 64. Nous nous orientons donc vers Cyberchef pour le décoder.



Après décodage de base64, on obtient du base32 qu'on décode également.



```

[~/.myHome/CTFs/LoveCTF/mise_a_l_epreuve]
$ cat 4_script.py
def ajouter_espaces(texte, taille_bloc=8):
    blocs = [texte[i:i+taille_bloc] for i in range(0, len(texte), taille_bloc)]
    return ' '.join(blocs)

# Exemple d'utilisation
sequence_binaire = "011111010110100000110101001100010111101001011111001101010011011
10011001101100001010111110100010001001000010111110011011001100010010111110111001101
10001101101010001100000111010101011111011010100011000001101110010111110011010101110
01101100011011010100011000001100111010111110100111001100011001101000110001001011111
01100111001100110011011101110101010000001111011010101010010010101001001110100011
010110110010001000001"
sequence_avec_espaces = ajouter_espaces(sequence_binaire)
print("Séquence avec un espace entre chaque bloc de 8 caractères :\n", sequence_ave
c_espaces)

[~/.myHome/CTFs/LoveCTF/mise_a_l_epreuve]
$ python3 4_script.py
Séquence avec un espace entre chaque bloc de 8 caractères :
 01111101 01101000 00110101 00110001 01111010 01011111 00110101 00110111 00110011 0
1100001 01011111 01000100 01001000 01011111 00110110 01100010 01011111 01110011 011
00011 01101010 00110000 01110101 01011111 01101010 00110000 01101110 01011111 00110
101 01110011 01100011 01101010 00110000 01100111 01011111 01001110 01100011 0011010
0 01100010 01011111 01100111 00110011 00110111 01110101 01000000 01111011 01010101
01001001 01010010 01110100 01101011 01100100 01000001

```

Donc nous avons deux possibilités de séquences binaires. Nous écrivons donc un script pour traduire le binaire en texte.

```

[~/.myHome/CTFs/LoveCTF/mise_a_l_epreuve]
$ cat 2_script.py
def binary_to_text(binary_string):
    # Divise la chaîne binaire en octets de 8 bits
    octets = [binary_string[i:i+8] for i in range(0, len(binary_string), 8)]

    # Convertit chaque octet en un caractère ASCII et les concatène
    text = ''.join(chr(int(octet, 2)) for octet in octets)

    return text

# Exemple d'utilisation
binary_string = "01111101 01101000 00110101 00110001 01111010 01011111 00110101 001
10111 00110011 01100001 01011111 01000100 01001000 01011111 00110110 01100010 01011
111 01110011 01100011 01101010 00110000 01110101 01011111 01101010 00110000 0110111
0 01011111 00110101 01110011 01100011 01101010 00110000 01100111 01011111 01001110
01100011 00110100 01100010 01011111 01100111 00110011 00110111 01110101 01000000 01
111011 01010101 01001001 01010010 01110100 01101011 01100100 01000001"
text = binary_to_text(binary_string.replace(" ", ""))
print("Texte converti :", text)

[~/.myHome/CTFs/LoveCTF/mise_a_l_epreuve]
$ python3 2_script.py
Texte converti : }h51z_573a_DH_6b_scj0u_j0n_5scj0g_Nc4b_g37u@fUIRtkdA

```



AAAAAAAHAH ! L'une des deux séquences binaires traduites en caractères ASCII semble être une piste pour obtenir le flag.

Ce résultat a d'abord l'air inversé, nous l'invertissons à nouveau.

```
$ python3
Python 3.11.8 (main, Feb 7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> sequence = "}"h51z_573a_DH_6b_scj0u_j0n_5scj0g_Nc4b_g37u@{UIRtkdA"
>>> print(sequence[::-1])
AdktRIU{@u73g_b4cN_g0jcs5_n0j_u0jcs_b6_HD_a375_z15h}
>>>
```

A cette étape, nous sommes à 200% sûrs d'être sur la bonne voie, il reste comment déchiffrer cette séquence. Pourquoi ne pas essayer dcode.fr ? Dcode nous parle donc de chiffre ROT, on essaie de déchiffrer avec et dans la liste de propositions voilà le coquin FLAG !!!!

ASCII[!-~] %HUXb-y\_3YUkCFVGZCKINGWwKINLYI  
+28 NGWCFxC,(CEuywC^swLa  
ASCII[!-~] ?birPGSy>s5le]`2aL]e.haq3]l.h]s.  
+2 haq]`4]FB]\_153]x/3f{  
ASCII+16 1T[dB9Ek0e'#W0R\$S>OW ZSc%0^ Z0e  
ZScOR&0840Q#'%0j!%Xm  
[A-Z0-9]+6 47enLC0{@01xa\_5y6H\_aud6mz\_hud\_ou  
d6m\_50\_B7\_4x1z\_tvzb}  
[A-Z0-9]+19 Ru1a8ZB{@bokx\_slt4\_xh0t9m\_4h0\_bh  
0t9\_sn\_YU\_rkom\_gimy}  
[A-Z0-9]+22 0ry75W8{@8lhu\_piq1\_uexq6j\_1ex\_8e  
xq6\_pk\_VR\_ohlj\_dfvj}  
[A-Z0-9]+18 Sv2b90C{@ply\_tmu5\_yiluan\_5i1\_ci  
lua\_to\_ZV\_slpn\_hjnz}  
[A-Z]+15 LoveCTF{f73r\_m4nY\_r0und5\_y0u\_f0  
und\_m6\_50\_1375\_k15s}  
ASCII[!-~] EhoxVMY!Dy;7kcf8gRck4ngw9cr4ncy4  
+90 ngwcf:cLHce7;9c~59l#  
ASCII[!-~] Qt{&bYe-  
+78 P'GCwoiDs^ow@zs%Eo~@zo'@zs%oizFox  
ToqCGEo,AEx/

ALPHABET PERSONNALISÉ (SENSIBLE À LA CASSE)

1234567890AZERTYUIOPQSDFGHJKLMWXCVBN1234567890azertyuiopq...

DÉCHIFFRER

Voir aussi : Code César — Chiffre par Décalages — Chiffre ROT-13 — Chiffre ROT-47

CHIFFREMENT PAR ROT-N

MESSAGE CLAIR À CHIFFRER PAR ROT

dCode ROT

ROTATION À UTILISER ROT-N, N= 13

ALPHABET À UTILISER

ABCDEF GHIJ KLMNOPQRSTUVWXYZ (EX: ROT13 / CÉSAR)

ABCDEF GHIJ KLMNOPQRSTUVWXYZ0123456789 (EX: ROT18)

94 CARACTÈRES ASCII IMPRIMABLES ENTRE ! (33) ET ~ (126) (EX: ROT47)

TABLE ASCII COMPLETE (128 CARACTÈRES)

ALPHABET PERSONNALISÉ (INSENSIBLE À LA CASSE)

1234567890AZERTYUIOPQSDFGHJKLMWXCVBN

ALPHABET PERSONNALISÉ (SENSIBLE À LA CASSE)

Faire un don

Paypal  
Amazon  
Autre

Forum/Aide

DISCORD

Mots-clés

rot, rotation, cesar, code, decalage, rot13, rot47, alphabet, circulaire

Liens

Contact  
A propos  
Application  
Prise2Tete