

Write-up - Quick Response Code 3/3

Auteur : Wa0k

CTF Flag'Malo 2022

Le challenge Quick Response Code 3/3 se base sur les deux challenges précédents en mélangeant toutes les mécaniques utilisées jusqu'à présent.

Voici donc plusieurs exemples de QR Code :



Pour retrouver le QR Code original, les opérations à mener varient en fonction du QR Code. En général, nous retrouvons : une translation horizontale ou verticale de la moitié du QR Code, et une inversion des couleurs sur deux parties de l'image.

Comme pour le challenge N°2, une méthode pourrait être :

- Diviser le QR Code en 4 quarts.
- Inverser les couleurs sur le quart 2 et le quart 3.
- Effectuer une translation horizontale et lire le QR Code. Si cela se résulte par un échec, alors effectuer une translation verticale et lire le QR Code.
- Renvoyer les données.

Ainsi, pour diviser le QR Code en 4 quarts, le principe est identique au challenge N°2. Par la même occasion et pour gagner du temps d'exécution, nous pouvons ajouter à cela l'inversion des couleurs sur le quart 2 et le quart 3.

La différence ici réside dans la méthode : en effet, au lieu de modifier directement le pixel comme dans le challenge N°2, nous pouvons récupérer la valeur (triplet) de chaque pixel dans une liste. Et à partir de cette liste, créer une nouvelle image RGB de plus petite taille.

De ce fait, lors de la translation, nous aurons juste à associer chacune de ses images pour reconstruire le QR Code à la taille originale.

En terme de code, voici ce que cela donne :

```
# Liste des valeurs des pixels pour chaque quart
quarter_list = [[],[],[],[]]

# Quart 1
for y in range(0, height//2):
    for x in range(0, width//2):
        quarter_list[0].append(qrcode.getpixel((x,y)))
quarter1 = Image.new("RGB", (width//2, height//2))
quarter1.putdata(quarter_list[0])

# Quart 2 + inversion des couleurs
for y in range(0, height//2):
    for x in range(width//2, width):
        if qrcode.getpixel((x,y)) == (255, 255, 255):
            quarter_list[1].append((0, 0, 0))
        else:
            quarter_list[1].append((255, 255, 255))
quarter2 = Image.new("RGB", (width//2+1, height//2))
quarter2.putdata(quarter_list[1])
```

```
# Quart 3 + inversion des couleurs
for y in range(height//2, height):
    for x in range(0, width//2):
        if qrcode.getpixel((x,y)) == (255, 255, 255):
            quarter_list[2].append((0, 0, 0))
        else:
            quarter_list[2].append((255, 255, 255))
quarter3 = Image.new("RGB", (width//2, height//2+1))
quarter3.putdata(quarter_list[2])

# Quart 4
for y in range(height//2, height):
    for x in range(width//2, width):
        quarter_list[3].append(qrcode.getpixel((x,y)))
quarter4 = Image.new("RGB", (width//2+1, height//2+1))
quarter4.putdata(quarter_list[3])
```

Les variables quarter1, quarter2, quarter3 et quarter4 sont donc des images du QR Code.

La dernière étape consiste à créer une nouvelle image all_quarter de la taille originale.

```
# Reconstitution du QR Code original
all_quarter = Image.new('RGB', (width, height)).convert("RGB")
```

Puis, nous effectuons une translation horizontale ou verticale en "collant" à des coordonnées précises chacun des quarts.

```
# Translation verticale
all_quarter.paste(quarter3, (0, 0))
all_quarter.paste(quarter4, (width//2, 0))
all_quarter.paste(quarter1, (0, height//2))
all_quarter.paste(quarter2, (width//2, height//2))
```

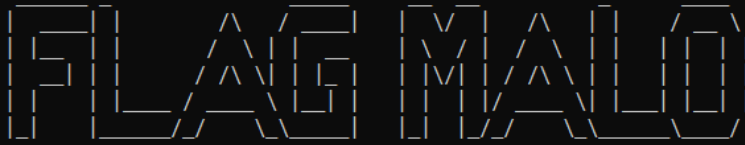
```
# Translation horizontale
all_quarter.paste(quarter2, (0, 0))
all_quarter.paste(quarter1, (width//2, 0))
all_quarter.paste(quarter4, (0, height//2))
all_quarter.paste(quarter3, (width//2, height//2))
```

Si la lecture est possible, la translation est correcte sinon une exception est levée et la deuxième translation est alors effectuée. En Python, cela peut se faire comme suit :

```
try:
    # Translation verticale
    all_quarter.paste(quarter3, (0, 0))
    all_quarter.paste(quarter4, (width//2, 0))
    all_quarter.paste(quarter1, (0, height//2))
    all_quarter.paste(quarter2, (width//2, height//2))
    result = decode(all_quarter)
    assert(len(result) > 0), "Erreur: Lecture impossible"
except:
    # Translation horizontale
    all_quarter.paste(quarter2, (0, 0))
    all_quarter.paste(quarter1, (width//2, 0))
    all_quarter.paste(quarter4, (0, height//2))
    all_quarter.paste(quarter3, (width//2, height//2))
    result = decode(all_quarter)
    assert(len(result) > 0), "Erreur: Lecture impossible"
return result[0].data
```

Au final, la réponse (c'est-à-dire les données) est renvoyée au serveur. Si le renvoi de la réponse a été fait en moins de 2 secondes, nous pouvons alors continuer le challenge. La suite consiste à répéter les mêmes opérations et ce 5 fois de suite au total.

Au final, en passant les 5 étapes nous obtenons le flag :



Ce challenge consiste à décoder une série de 5 QR codes : vous recevrez en base 64 un QR code à la fois que vous devez décoder puis renvoyez le message correspondant en moins de 2 secs.

Quand vous êtes prêt, envoyez : start !

>> start

```
>> b'D7e4wuQ7gexi46qz5dVmMMSWbfScbQA8Q0E8HqCvbyJaEn0iDYLjzR8bLX89myrE4A4GtND5T33kAhMYMB2Moj1XD87ooVd7VVmp'  
>> b'VCik0ipqGMTP4VXnp3ZA06EN5Eul1QhHvSdIGH3D0jmBhiz05ePDEXYoJLya5n96gtxjGebq02ok0KVPASzK19jESVSw5Q21Dgm'  
>> b'seijCnFE7uXXHsk4FDticuPAW1Smu19gy7IxPurRoIxt0RxGVgzyM1qNtpHSfvzgPOSDcdKYNA0ga1hyJZdq2gXgQwAbgh1sKDLR'  
>> b'9gxw13PzlymEW6Nc4Dia5eVLS81ylQiViXm9p43RCJdUninZGgaNep5oHTR4d02Ndr2UIGewKMCx1dGoj5sNnk485T1ADC1I51V'  
>> b'UnVChdVDGG2N3Xq47Guz7BydIvalYjyQHqbDZEHFk6Mgnsxic11A0ykLFWZ0Caj0IHY8QCP0hi5HU8fwIwQ5cSM02iF4mL9o6ax0'
```

Bravo ! Vous avez réussi, voici le flag : FMCTF{QrC0D3_h@s_No_\$ecR3t\$_4_U}

Le flag du challenge est : FMCTF{QrC0D3_h@s_No_\$ecR3t\$_4_U}.