

Writeup pour le challenge KeyQuest de la phase de qualification pour le HackerLab2024

Nom d'utilisateur : takeoff

E-mail : robertohoungbo@gmail.com

Affiliation : IFRI

Pays : Bénin

KeyQuest

200

Reverse

[FR]

Réalisez un audit du système d'authentification de la centrale électrique.

L'objectif est de déterminer s'il est possible de retrouver la clé de l'utilisateur BJIZ-HACKERLAB.

Voici l'OPCODE de la fonction.


[EN]

Conduct an audit of the authentication system at the power plant.

The objective is to determine if it is possible to retrieve a user's key from their username.

Here is the function's OPCODE.

Author: 5c0r7

 opcode

L'objectif de ce challenge est de retrouver la clé de l'utilisateur BJIZ-HACKERLAB en réalisant un audit sur le système d'authentification de la centrale électrique. Pour cela un fichier opcode nous a été fourni. Le fichier contenait un code décompilé. Lorsqu'on regarde de plus près, on constate que le code retrace le processus de reconnaissance de la clé de l'utilisateur.

Après maintes lectures, j'avais toujours des points d'ombres alors j'ai simplement demandé à Chatgpt de m'expliquer le code. Ce qu'il a fait avec brio. Grâce à ça j'ai pu comprendre le processus complet.

En gros la reconnaissance de la clé se fait avec la fonction check qui implique trois fonctions principales dans l'ordre whippin3 puis whippin4 puis whippin5. Après le résultat de whippin5 est comparé à la valeur de y_key et si les valeurs correspondent ça nous affiche que le flag est la clé de l'utilisateur sinon ça nous affiche une erreur.

Voici le code réécrit en python :

```
def check(username, y_key, real_password):
    hint = '\n password = whippin3(key)(real_password) to keep real_password safe\n'
    key = -9

    def whippin5(inpt):
        sh = md0()
        sh.update(inpt.encode())
        return sh.hexdigest()

    def whippin3(n):
        lc = string.ascii_lowercase
        uc = string.ascii_uppercase
        dc = string.digits
        trans = str.maketrans(lc + uc + dc, lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n])
        return lambda s: s.translate(trans)
```

```
def whippin4(a, b):
    b_etx = (len(a) // len(b) + 1) * b
    return ''.join(chr(ord(c) ^ ord(d)) for c, d in zip(a, b_etx))

if whippin5(whippin4(username, real_password)) == y_key:
    if username == 'BJIZ-HACKERLAB':
        print('Congratz, you can use this flag to validate : HLB2024{' + y_key
    else:
        print("Good, but the key of BJIZ-HACKERLAB' is the flag")
else:
    print('Error, checking failed')
```

Après cette claire explication, j'ai pu décrypter le `crypted_password` qui nous a été donné en inversant la fonction `whippin3` comme suit :

```
def reverse_whippin3(n):
    lc = string.ascii_lowercase
    uc = string.ascii_uppercase
    dc = string.digits
    trans = str.maketrans(lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n], lc +
    return lambda s: s.translate(trans)

# Application de la fonction inverse
key = -9
crypted_password = "dpjLgviGRJJN1IUUFeku1ls8"
reverse_trans = reverse_whippin3(key)
decrypted_password = reverse_trans(crypted_password)
```

Après exécution de ce code j'obtiens le résultat suivant :
mysUpErPASSW0RDDOnTd0ub7

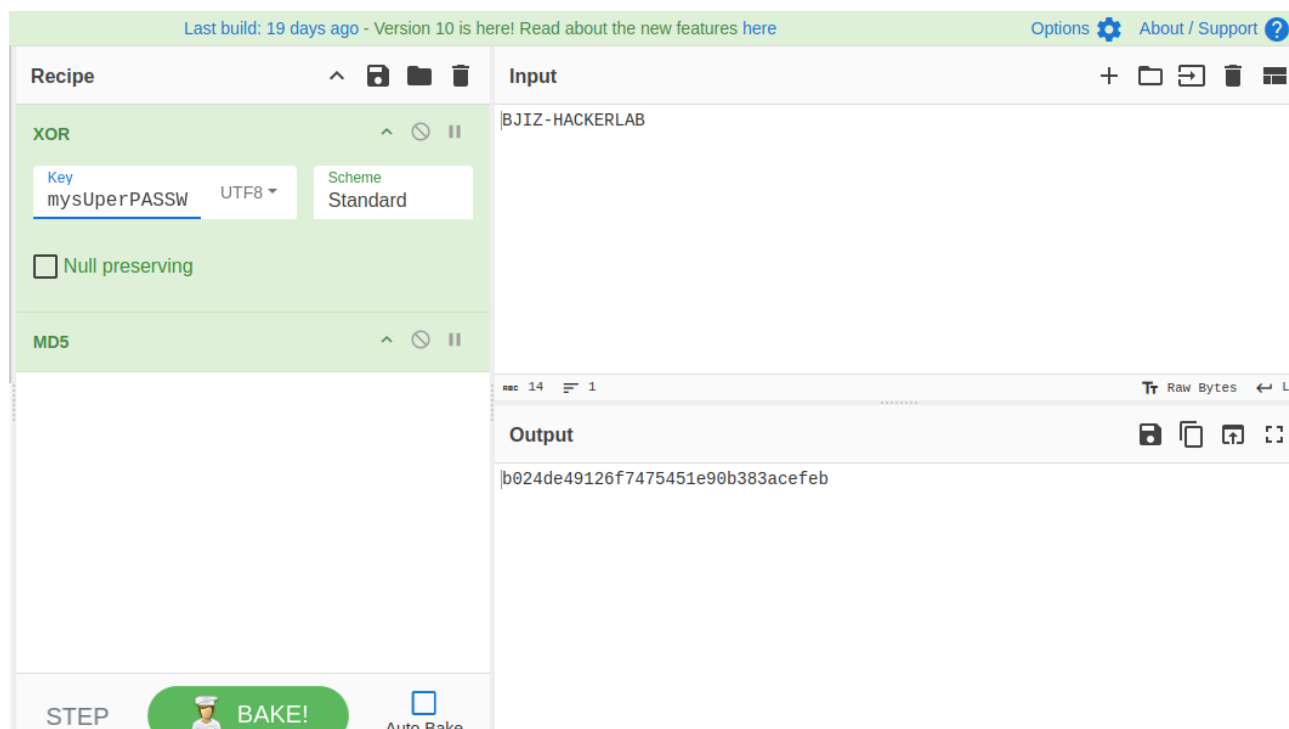
J'ai dans un premier cru que c'était le flag et je l'ai soumis mais c'était pas le flag. J'ai demandé à un admin, il m'a confirmé que non. Je me suis alors retourné vers le code et c'est là je me suis rendu compte qu'il restait encore deux fonctions :

- le whippin4 qui fait le xor entre le mot de passe en clair que j'ai obtenu et le username de l'utilisateur ;

- le whippin5 qui fait le hachage md5 du résultat de whippin5 et le compare à y_key.

Donc si tout est bien fait le résultat de whippin5 est le flag que je cherche.

J'ai alors continué sur cyberchef en faisant le xor puis en encodant le résultat en md5 et j'ai eu le contenu du flag.



FLAG : HLB2024{b024de49126f7475451e90b383acefeb}

Merci !