



Le système d'authentification décrit dans le code fonctionne de la manière suivante :

1. Initialisation et Définitions

- **Lignes 6-12 :** Le code commence par définir une chaîne `hint` comme indice ou message d'information. Ensuite, il initialise la variable `key` à `-9`. Enfin, il crée trois fonctions (`whippin5`, `whippin3`, `whippin4`) en utilisant `MAKE_FUNCTION`. Ces fonctions seront utilisées pour des opérations spécifiques plus tard dans le script.

2. Définition des Fonctions

Fonction `whippin5` (Lignes 13-15)

- Cette fonction utilise la bibliothèque `md0` (probablement `hashlib` ou similaire) pour calculer un hachage SHA (`sh`) de l'entrée `inpt`.

Fonction `whippin3` (Lignes 17-21)

- Cette fonction prépare une table de traduction (`trans`) pour effectuer une substitution de caractères basée sur des listes de lettres minuscules (`lc`), majuscules (`uc`), et chiffres (`dc`). Elle utilise cette table pour créer une fonction lambda qui traduit une chaîne de caractères en utilisant cette table.

Fonction `whippin4` (Lignes 23-56)

- Cette fonction prend deux arguments (`a` et `b`) et calcule `b_etx`, qui est utilisé comme un indice dans le traitement ultérieur. Elle génère une chaîne de bytes en utilisant un générateur (`<genexpr>`) qui effectue un XOR entre les caractères de `a` et `b`, puis encode le résultat.

3. Validation Principale (Lignes 25-92)

- **Lignes 25-44 :** La validation principale commence en appelant les fonctions `whippin5` et `whippin4` avec les paramètres `username` et `real_password`.
 - `whippin5(username, real_password)` est utilisé pour générer une clé (`y_key`).
 - `whippin4(a, b)` est utilisé pour générer une chaîne de bytes (`b'...'.`).
- **Lignes 46-74 :** Si la clé générée (`y_key`) est égale à `-9` et si `username` est `'BJIZ-HACKERLAB'`, un message de félicitations est affiché avec un drapeau construit.
- **Lignes 76-84 :** Sinon, si `username` n'est pas `'BJIZ-HACKERLAB'`, un message est imprimé indiquant que la clé associée à cet utilisateur est le drapeau.
- **Lignes 86-92 :** En cas d'échec de la validation, un message d'erreur est imprimé.

4. Conclusion

Ce système d'authentification semble utiliser des techniques de hachage (avec `whippin5`), de substitution de caractères (avec `whippin3`), et de génération de bytes (avec `whippin4`) pour vérifier si un utilisateur spécifique (probablement `'BJIZ-HACKERLAB'` dans ce cas) a le bon mot de passe ou clé. Si les conditions sont remplies, un drapeau ou un message de validation est affiché ; sinon, un message d'erreur est affiché.

Avec l'aide de chatgpt j'ai écrit un code pour déchiffrer le message car moi je ne suis pas bon dans le domaine du rev

```
File Edit Selection Find View Goto Tools Project Preferences Help
lab.py x outu4t.py x serveu.py x rsa (2).py x oRutput.txt x ste.py x keygenme-trial.py x lab.py — hacl x lab.py — Musique x
1 import hashlib
2 import string
3
4 # Fonction pour hasher en MD5
5 def hash_md5(input_str):
6     return hashlib.md5(input_str.encode()).hexdigest()
7
8 # Fonction pour effectuer une permutation des caractères dans l'alphabet
9 def permute_chars(n):
10     lc = string.ascii_lowercase
11     uc = string.ascii_uppercase
12     dc = string.digits
13
14     trans = str.maketrans(
15         lc + uc + dc,
16         lc[n:] + lc[:n] + uc[n:] + uc[:n] + dc[n:] + dc[:n]
17     )
18
19     return lambda s: s.translate(trans)
20
21 # Fonction pour décrypter avec XOR
22 def xor_decrypt(a, b):
23     b_extended = b * (len(a) // len(b) + 1)
24     return ''.join(chr(ord(c) ^ ord(d)) for c, d in zip(a, b_extended))
25
26 # Clé initiale de permutation
27 initial_permutation = -9
28 decryption_permutation = -initial_permutation
29
30 # Données cryptées
31 donnees_cryptees = 'dpjLgviGRJJN1IUUFeku1ls8'
32
33 # Décryptage des données
34 donnees_reelles = permute_chars(decryption_permutation)(donnees_cryptees)
35
kali@kali: ~/Formation/rev x kali@kali: ~/Musique x
(kali@kali) - [~/Musique]
$ python3 lab.py
Données décryptées : mysUperPASSWORDDOnTd0ub7
Voici le Flag : HLB2024{b024de49126f7475451e90b383acefeb}
(kali@kali) - [~/Musique]
$
donnees_reelles = permute_chars(decryption_permutation)(donnees_cryptees)

print('Données décryptées :', (donnees_reelles))

nom_utilisateur = 'B117-RACKERLAB'
y_key = hash_md5(xor_decrypt(nom_utilisateur, donnees_reelles))

def validate_nom_utilisateur(x, y_key):
```