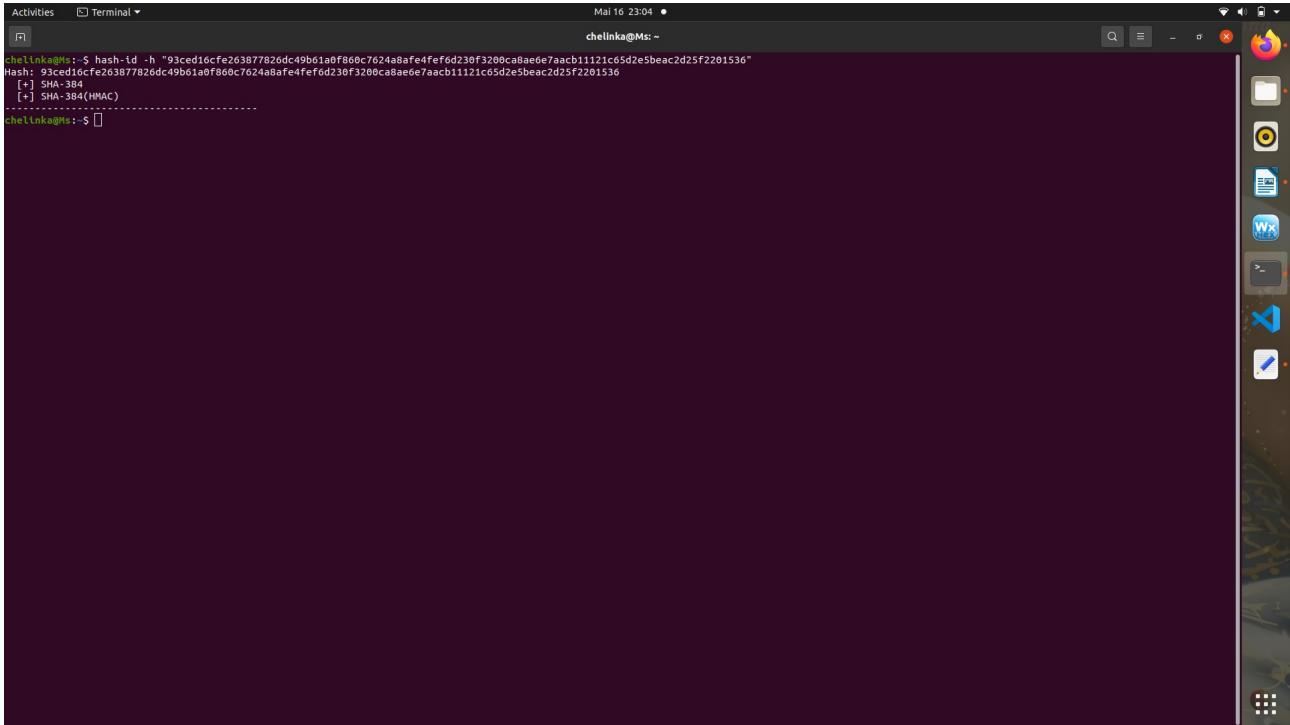


Hashes, Maths and Bruteforce

Ce challenge nous demande de faire de la concaténation, du hachage et du bruteforce. C'est parti !

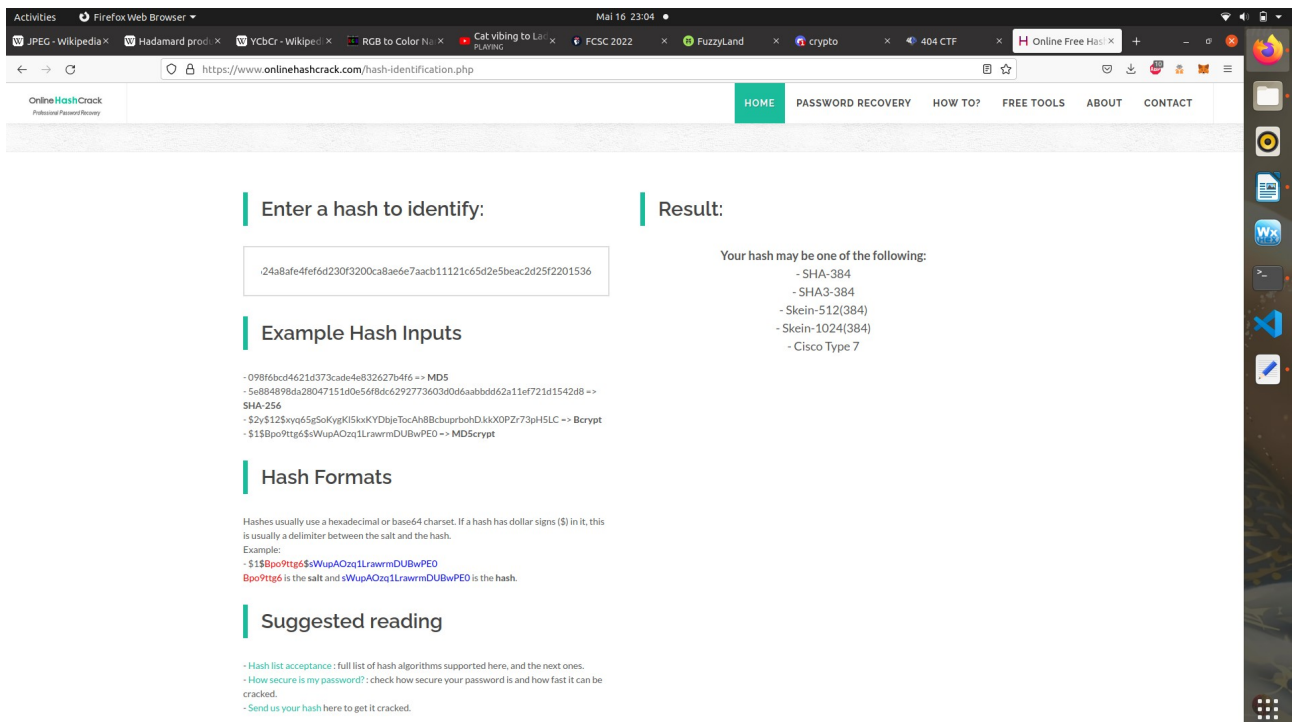
Il nous faut d'abord trouver le type du hash qui n'est pas donné dans l'énoncé. Pour cela, au moins deux méthodes sont disponibles :

Une simple recherche à travers hash-id nous permet d'identifier le type de hash :



```
chellinka@ms:~$ hash-id -h "93ced16cfe263877826dc49b61a9f860c7624a8afe4fed230f320eca8ae6e7aacb11121c65d2e5beac2d25f2201536"
Hash: 93ced16cfe263877826dc49b61a9f860c7624a8afe4fed230f320eca8ae6e7aacb11121c65d2e5beac2d25f2201536
[+] SHA-384
[+] SHA-384(HMAC)
chellinka@ms:~$
```

L'internet est pourvu de sites permettant d'identifier quel hash est probablement la chaîne de caractères qu'on lui pourvoit :



Online HashCrack
Professional Password Recovery

HOME | PASSWORD RECOVERY | HOW TO? | FREE TOOLS | ABOUT | CONTACT

Enter a hash to identify:

Result:

Your hash may be one of the following:

- SHA-384
- SHA3-384
- Skein-512(384)
- Skein-1024(384)
- Cisco Type 7

Example Hash Inputs

- 098f6bcd4621d373cade4e832627b4f6 => MD5
- 5e884898da28047151d0e56f8dc6292773603d0daabdd62a11ef721d1542d8 => SHA-256
- \$2y\$12\$xyq65g5oKyqKI5kxKYDbeTocAh8BcbupbohDkxXOPZr73p45LC -> Bcrypt
- \$1\$Bpo9ttg6\$WupAQzq1LrawmDUBwPEO -> MD5crypt

Hash Formats

Hashes usually use a hexadecimal or base64 charset. If a hash has dollar signs (\$) in it, this is usually a delimiter between the salt and the hash.
Example:
- \$1\$Bpo9ttg6\$WupAQzq1LrawmDUBwPEO
Bpo9ttg6 is the salt and sWupAQzq1LrawmDUBwPEO is the hash.

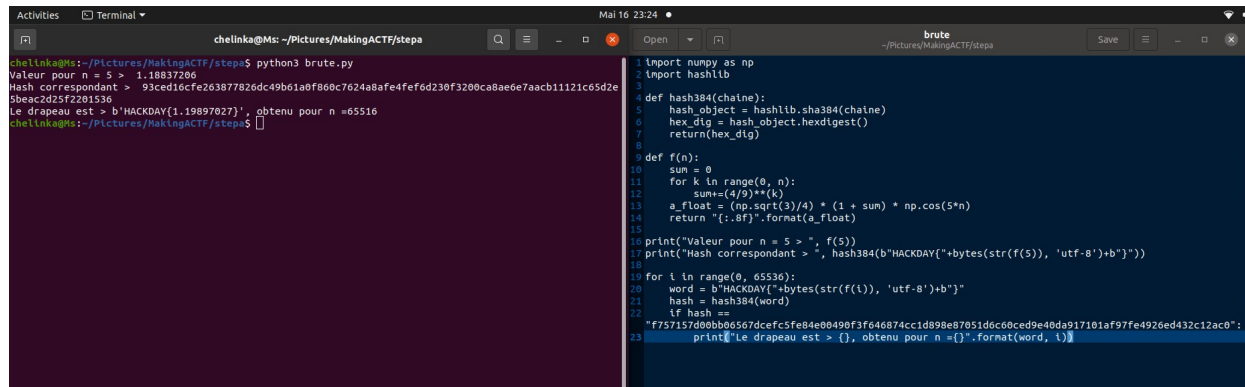
Suggested reading

- Hash list acceptance : full list of hash algorithms supported here, and the next ones.
- How secure is my password? : check how secure your password is and how fast it can be cracked.
- Send us your hash here to get it cracked.

Le hash est donc en **SHA-384**

J'ai décidé de faire un algorithme de résolution en python3, car c'est un langage qui est simple à mettre en place, et il dispose de paquets qui feront le hashage à ma place (encore heureux).

Nous mettons la formule dans une fonction prenant en argument n, et nous calculons le hash pour toutes les valeurs positives :



```
chellinka@Ms: ~/Pictures/MakingACTF/steps
chellinka@Ms:~/Pictures/MakingACTF/steps$ python3 brute.py
Valeur pour n = 5 > 1.18837206
Hash correspondant > 93ced16cfe263877826dc49b61a0f860c7624a8afe4fef6d230f3200ca8ae6e7aacb11121c65d2e
5beac2d25f220b1536
le drapeau est > b'HACKDAY{1.19897027}', obtenu pour n =65516
chellinka@Ms:~/Pictures/MakingACTF/steps$
```

```
1 import numpy as np
2 import hashlib
3
4 def hash384(chaine):
5     hash_object = hashlib.sha384(chaine)
6     hex_dig = hash_object.hexdigest()
7     return(hex_dig)
8
9 def f(n):
10     sum = 0
11     for k in range(0, n):
12         sum += (4/9)**(k)
13     a_float = (np.sqrt(3)/4) * (1 + sum) * np.cos(5*n)
14     return "{:.8f}".format(a_float)
15
16 print("Valeur pour n = 5 > ", f(5))
17 print("Hash correspondant > ", hash384(b'HACKDAY{"'+bytes(str(f(5)), 'utf-8')+b'"'))
18
19 for i in range(0, 65536):
20     word = b'HACKDAY{"'+bytes(str(f(i)), 'utf-8')+b'"')
21     hash = hash384(word)
22     if hash ==
23         "f757157d00bb06567dcfc5fe04e00498f3f646874cc1d898e07851d0c00ced9e40da917101af97fe4926ed432c12ac0":
24         print("Le drapeau est > {} , obtenu pour n = {}".format(word, i))
```

Le flag tombe plus ou moins rapidement, **HACKDAY{1.19897027}**