Involucrypt 1:

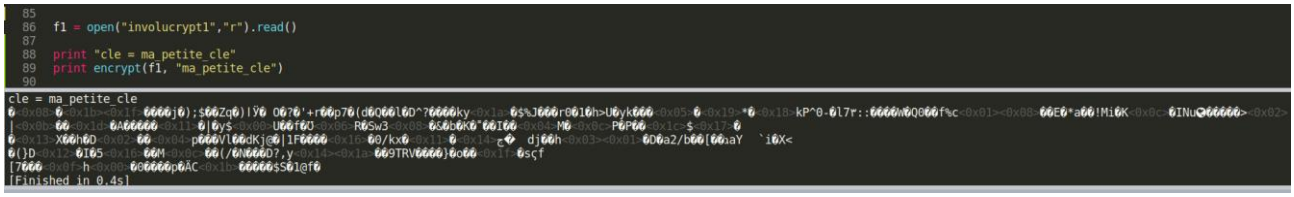
Dans ce nouveau challenge, nous avons en notre possession 2 fichiers. Le premier est un fichier python qui permet le chiffrement d'un fichier. Le second est justement le fichier chiffré à déchiffrer.

```python
import itertools
import operator
import sys
import string

BLOCK = 150
global list_ascii

class mersenne_rng(object):
    def __init__(self, seed=5489):
        self.state = [0] * 624
        self.f = 1812433253
        self.m = 397
        self.u = 11
        self.s = 7
        self.b = 0x9D2C5680
        self.t = 15
        self.c = 0xEFC60000
        self.l = 18
        self.index = 624
        self.lower_mask = (1 << 31) - 1
        self.upper_mask = 1 << 31

        # update state
        self.state[0] = seed
        for i in range(1, 624):
            self.state[i] = self.int_32(
                self.f * (self.state[i - 1] ^ (self.state[i - 1] >> 30)) + i
            )

    def twist(self):
        for i in range(624):
            temp = self.int_32(
                (self.state[i] & self.upper_mask)
                + (self.state[(i + 1) % 624] & self.lower_mask)
            )
            temp_shift = temp >> 1
            if temp % 2 != 0:
                temp_shift = temp_shift ^ 0x9908B0DF
            self.state[i] = self.state[(i + self.m) % 624] ^ temp_shift
        self.index = 0

    def get_random_number(self):
        if self.index >= 624:
            self.twist()
        y = self.state[self.index]
        y = y ^ (y >> self.u)
        y = y ^ ((y << self.s) & self.b)
        y = y ^ ((y << self.t) & self.c)
        y = y ^ (y >> self.l)
        self.index += 1
        return self.int_32(y)
```

```python
    def get_rand_int(self, min, max):
        return (self.get_random_number() % (max - min)) + min

    def int_32(self, number):
        return int(0xFFFFFFFF & number)

    def shuffle(self, my_list):
        for i in range(len(my_list) - 1, 0, -1):
            j = self.get_rand_int(0, i + 1)
            my_list[i], my_list[j] = my_list[j], my_list[i]


def keystream(seeds, length, base=None):
    key = base if base else []
    for seed in seeds:
        random = mersenne_rng(seed)
        for _ in range(BLOCK):
            if len(key) == length:
                break
            key.append(random.get_rand_int(0, 255))
            random.shuffle(key)
        if len(key) == length:
            break
    return key

def encrypt(string, key):
    key = keystream(map(ord, key), len(string))
    stream = itertools.starmap(operator.xor, zip(key, map(ord, string)))
    return bytearray(stream)
```

On remarquera que j'ai juste supprimé le main car je n'en n'aurai pas besoin dans la suite.

Parfait, ne nous attardons pas trop sur ce programme (car je comprends pas tout 😉). Commençons par ouvrir notre fichier chiffré et tentons de le déchiffrer avec une clé au hazard.



Nous nommons notre fichier « f1 » et nous le déchiffrons avec la clé « ma_petite_cle »

Répétons l'opération avec une autre clé.

Vous remarquez ? Ma clé est maintenant « ma_grande_cle » mais le résultat est le même.

On se rend vite compte (quelques tests en plus si nécessaire) que la taille de la clé est de taille 3.

Parfait, lancons un brute force bête et méchant.

```
import string
f1 = open("involucrypt1","r").read()
f2 = open("res.txt","w")
for i in string.printable:
    for j in string.printable:
        for k in string.printable:
            text = encrypt(f1, i+j+k)
            f2.write(text)
```

Nous allons donc déchiffrer notre fichier avec toutes les combinaisons possible des caractères imprimables et rediriger chaque résultat vers un fichier text. On attend… (14 h pour ma part)

Une fois bien avancé (pas obligé que le programme soit terminé pour vérifier si on a déjà un résultat) nous faisons quelques grep. Et on retrouve facilement avec : strings res.txt | grep flag

Voilà, la clé est « ane » et notre flag est : TheKeyIsTooDamnWeak!

```
cle = ane
The flag is: TheKeyIsTooDamnWeak!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
```

On remarquera que la clé ne comportait que des minuscules, on aurait pu gagner du temps, mais on ne pouvait pas savoir.

Involucrypt 2:

Nouveau challenge Involucrypt, trop bien la crypto!!! 😊

Encore une fois, nous avons un programme python et un fichier chiffré.

```python
import itertools
import operator
import sys

BLOCK = 150

class mersenne_rng(object):
    def __init__(self, seed=5489):
        self.state = [0] * 624
        self.f = 1812433253
        self.m = 397
        self.u = 11
        self.s = 7
        self.b = 0x9D2C5680
        self.t = 15
        self.c = 0xEFC60000
        self.l = 18
        self.index = 624
        self.lower_mask = (1 << 31) - 1
        self.upper_mask = 1 << 31

        # update state
        self.state[0] = seed
        for i in range(1, 624):
            self.state[i] = self.int_32(
                self.f * (self.state[i - 1] ^ (self.state[i - 1] >> 30)) + i
            )

    def twist(self):
        for i in range(624):
            temp = self.int_32(
                (self.state[i] & self.upper_mask)
                + (self.state[(i + 1) % 624] & self.lower_mask)
            )
            temp_shift = temp >> 1
            if temp % 2 != 0:
                temp_shift = temp_shift ^ 0x9908B0DF
            self.state[i] = self.state[(i + self.m) % 624] ^ temp_shift
        self.index = 0

    def get_random_number(self):
        if self.index >= 624:
            self.twist()
        y = self.state[self.index]
        y = y ^ (y >> self.u)
        y = y ^ ((y << self.s) & self.b)
        y = y ^ ((y << self.t) & self.c)
        y = y ^ (y >> self.l)
        self.index += 1
        return self.int_32(y)

    def get_rand_int(self, min, max):
        return (self.get_random_number() % (max - min)) + min
```

```python
    def int_32(self, number):
        return int(0xFFFFFFFF & number)

    def shuffle(self, my_list):
        for i in range(len(my_list) - 1, 0, -1):
            j = self.get_rand_int(0, i + 1)
            my_list[i], my_list[j] = my_list[j], my_list[i]


def keystream(seeds, length, base=None):
    key = base if base else []
    for seed in seeds:
        random = mersenne_rng(seed)

        for _ in range(BLOCK):
            if len(key) == length:
                break
            key.append(random.get_rand_int(0, 255))
            random.shuffle(key)
        if len(key) == length:
            break
    return key


def encrypt(string, key):
    key = keystream(map(ord, key), len(string))
    stream = itertools.starmap(operator.xor, zip(key, map(ord, string)))

    return bytearray(stream)
```

On se retrouve avec un programme globalement ressemblant au premier mais encore une fois, ne regardons pas le programme (toute façon j'y comprendrais rien 😉).

Quelques tests comparables à ceux réalisé au premier challenge nous permettent de dire que la clé est maintenant de taille 10. Cela fait beaucoup pour un bf d'autant plus qu'on ne connait pas les caractères compris dans la clé (chiffres, minuscules, majuscules et caractères spéciaux).

Faisons un listing de ce que l'on sait:

-la clé est de 10

-le programme est presque identique au 1

Conclusion, retournons sur le challenge Involucrypt1.

Nous allons faire une série de tests sur le premier challenge pour voir s'il y'a des choses à voir…

Ce que l'on sait du 1 maintenant que nous l'avons réussi:

-La clé est de taille 3

-La clé est "ane"

Ne reste plus qu'à savoir ce que nous pouvons realiser comme tests.

Un peu mal au point au début mais une idée bat rapidement de l'aile dans mon esprit.

Tentons de déchiffrer notre fichier (toujours celui de involucrypt1 !!) avec les clé suivantes:

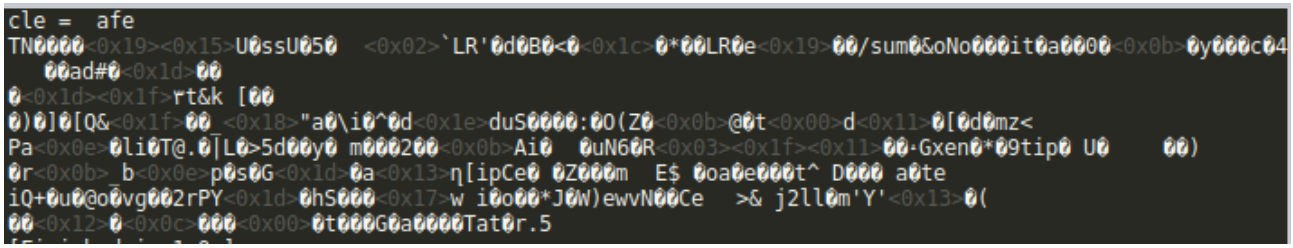
1ère série, nous changeons le "a" de "ane"

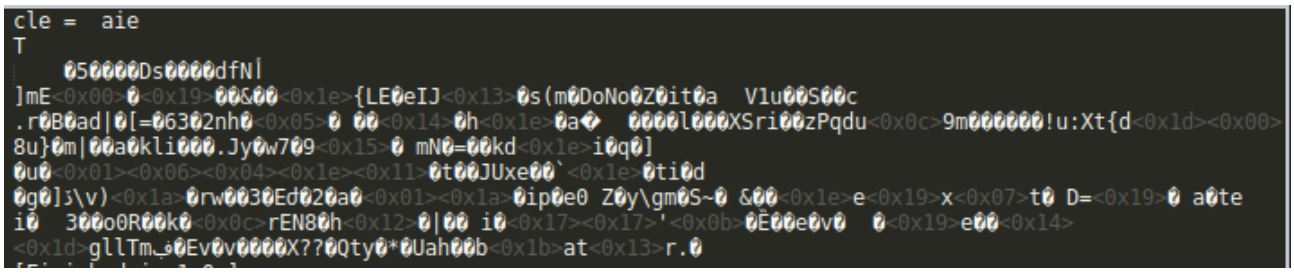
```
cle =  zne
The0fG<0x04>~<0x1a>zs:<0x0e><0x04>h0Key0sTooDamn$?akF0
Lo0e010qs0m solor Zit av00,Oc'nsdct0tu0 adip0s2i<0x15>0<nli0, sdd Io Y0u7m.d tem0br0if00;iduntOOO
laqo0e <0x18>t d0lo<0x7f>( mlgna0Uliqua. <0x16>" 0nim0a} mi/0m<0x00>0bni0m,
0uBs00~#t0ud0exer00<0x1d>0ti00 00<0x19>aA0o K00ori* %i<0x16>i 0&Xali0uip ex <0x15>a comhod~
0ojAequat<0x1b> DuiR aute i0are0do0o: 06Qrh0@ehwZde0it in v0luptU0eRveE~t e7sk 0<0x05>llum 50lo(u 00
<0x10>00iat n0lla pE5atUr.
```

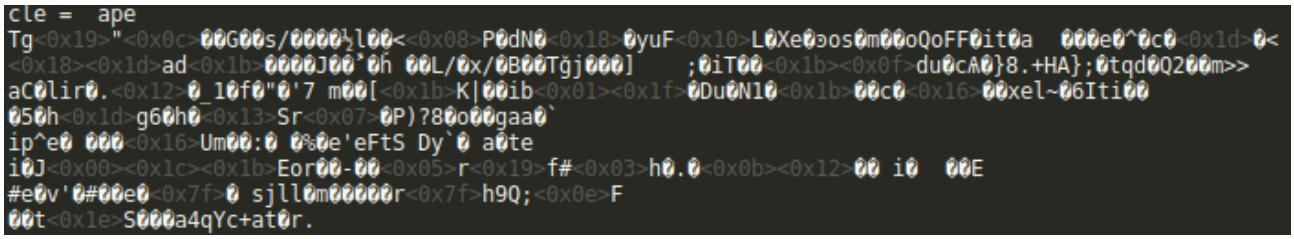
```
cle =  xne
The0f000/0s:Nzh0Key0sTooDamn00ak00
Lo0e4u00s\m 0olor Yit a0x,0cAns[ct0tu0 adipD0&<0x01>i0KliZ, s\d 0o [0u0m0d tem00r0i09uFiduntO0R
la0oQe 0t d0lo<0x1f><0x02> m0gna00liqua. 00 0nimma| mi0mmy<0x04>0ni`m, 7u0smK<0x13>0t0ud0exer0000tiL
 <0x1e>0<0x1c>a00o *0%orij <0x08>i0i 0>#ali0uip ex 0a com0od8 xo'0equat0 Dui0 aute i0~re2do<0x7f>oz
 0<0x02>0rN$0eh,0de<0x0f>it in v<0x1e>lupt00e0ve
2t e0s0 0\llum Tlo!0 %
 0<0x11>0iat n0lla p00Mat0r.
```

```
cle =  jne
TheUfI00r9s:+0h0Key0sTooDamn04ak<0x0b>0
Lose00!<0x08>s0m 0olor 0it a0<0x07>x,Dc0ns]ct0tu0 adip000<0x02>000~li0, s7d .o 03u0m~d
tem0wrDiIuvidunt0;0 la0o0e et d]lo]
 m0gna0<0x12>liqua. 0<0x1a> 0nimDa0 mi<0x9b>m0X0ni0m, Ru0sg00nt0udIexer0<0x1d>0<0x1a>ti06 J0a0So
 00#orih Ti0i 0:0ali~uip ex 0a comtod0 0o00equat0 Dui0 aute i
0re0doNop "<0x17>0r00<0x0f>eh0<0x1b>de<0x05>it in v<0x13>lupt0=e0ve00t e<0x1e>sr 2-llum
0<0x00>lo<0x1c>% 0L 00eiat n0lla p<0x14>0~at0r.
```

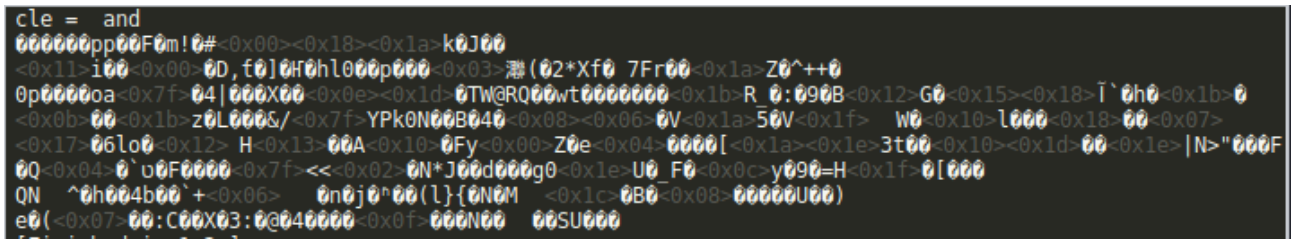
2ème série nous changeonsle “n” de “ane”

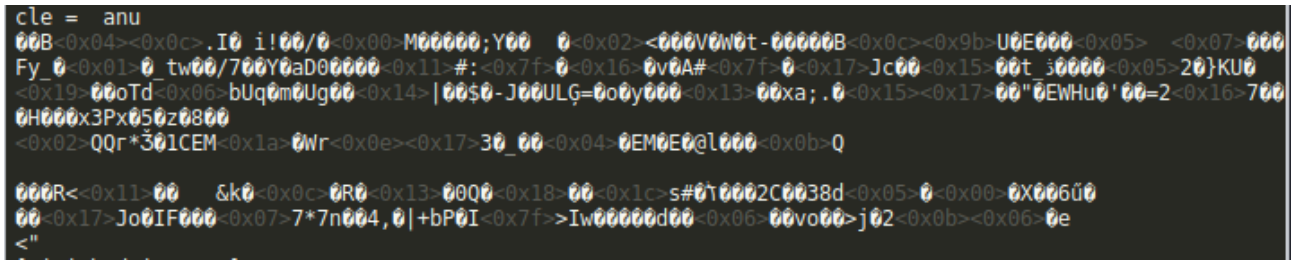


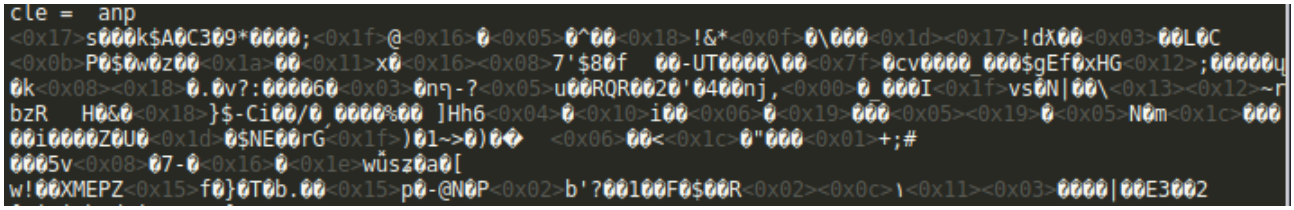




3ème série nous changeonsle “e” de “ane”







Regardez bien tous ces résultats…

On peut facilement remarquer qu'en function de la letter que l'on change notre résultat change beaucoup.

Je m'explique: dans les premiers tests (le "a" est changé), certes nous ne pouvons pas lire le résultat néanmoins, nous remarquons une multitude de caractères ASCII.
Dans le deuxième, c'est encore pire. Nous sommes incapable de deviner un ne serait ce qu'un seul mot et le trois c'est encore pire que celui du 2. C'est justement ici que viens ce contruire la base du raisonnement qui nous aidera à résoudre le challenge Involucrypt 2.
La quantité de caractères ASCII dans la troisième série de tests est inférieure à celle de la deuxième série.

En bref:

Donc : Soit X un caractère ASCII (sauf le bon)

Soit Q(clé) la quantité de caractères ASCII dans le résultat du déchiffrement

Q(Xne) > Q(aXe) > Q(anX)

Quelques tests supplémentaires à realiser et nous pouvons maintenant affirmer que:

En bref:

Donc : Soit X un caractère ASCII (sauf le bon)

Soit Q(clé) la quantité de caractères ASCII dans le résultat du déchiffrement

Q(XXe) > Q(XnX) > Q(aXX)

On peut donc dire que nous pouvons trouver notre clé juste en cherchant le dernier caractère de celle-ci est ainsi de suite nous décaler vers la gauche. Tout cela en comptant le nombre de caractères ASCII que nous obtenons dans notre résultat.

Bien, maintenant retournons sur Involucrypt 2.

Voici notre petit script à rajoutter:

```
f1 = open("involucrypt2","r").read()
alp = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"]
nb_caract_ascii = []
for i in alp:
    print "cle = XXXXXXXXX",i
    res =  encrypt(f1, "XXXXXXXXX"+i)
    count = 0  # va nous permettre de savoir cb de caracteres ascii sont presents
    for x in res:
        if (x>64) and (x<128):   # l'intervalle dasn lequel on considere qu'un caractere est ascii
            count += 1
    nb_caract_ascii.append(count) # on affiche la liste
print nb_caract_ascii
```

Et puis nous attendons…

“petit à petit l’oiseau fait son nid”

Voici ici les résultats obtenu à chaque tour.

```
[372, 332, 353, 380, 369, 391, 384, 399, 487, 397, 381, 357, 347, 388, 375, 356, 360, 377, 337, 350, 360, 338, 371, 384, 355, 365] cle = XXXXXXXXXi
[487, 451, 447, 465, 430, 455, 428, 444, 434, 455, 435, 441, 445, 423, 523, 423, 436, 443, 462, 433, 486, 435, 438, 450, 417, 439] cle = XXXXXXXXoi
[523, 517, 545, 523, 532, 567, 545, 560, 568, 550, 541, 521, 541, 549, 549, 553, 528, 533, 534, 624, 539, 557, 542, 562, 556, 525] cle = XXXXXXXtoi
[624, 634, 632, 630, 637, 610, 609, 619, 637, 622, 657, 605, 620, 661, 625, 708, 632, 618, 630, 613, 660, 628, 628, 639, 641, 629] cle = XXXXXXptoi
[708, 716, 721, 725, 720, 718, 688, 698, 707, 717, 698, 711, 724, 699, 701, 713, 732, 715, 803, 728, 751, 707, 719, 691, 718, 700] cle = XXXXXsptoi
[803, 803, 816, 840, 806, 802, 816, 825, 900, 796, 821, 803, 800, 797, 796, 811, 797, 803, 804, 802, 806, 784, 818, 832, 807, 814] cle = XXXXisptoi
[900, 891, 902, 904, 894, 980, 888, 911, 901, 889, 898, 890, 899, 908, 903, 904, 898, 894, 885, 926, 901, 915, 892, 899, 915, 904] cle = XXXfisptoi
[980, 999, 995, 990, 984, 985, 1009, 988, 972, 1075, 977, 983, 971, 995, 988, 977, 977, 995, 989, 975, 995, 980, 994, 987, 989, 992] cle = XXjfisptoi
[1075, 1078, 1075, 1075, 1066, 1092, 1082, 1077, 1072, 1257, 1073, 1080, 1082, 1071, 1091, 1087, 1084, 1080, 1086, 1091, 1074, 1069, 1091, 1078, 1062, 1070] cle = Xjjfisptoi
[1257, 1162, 1170, 1161, 1162, 1164, 1160, 1159, 1164, 1153, 1163, 1156, 1166, 1166, 1161, 1170, 1159, 1163, 1163, 1166, 1160, 1160, 1166, 1158, 1168, 1174] cle = ajjfisptoi
```

Plus qu’à verifier:

```
84
85  f1 = open("involucrypt2","r").read()
86  print encrypt(f1, "ajjfisptoi")
It's something about ya girl,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
that just makes my head wanna twirl,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
oh you got me want to tell all them other girls,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
there's nothing else better in this world!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
the moment i seen her i was in shock
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
so shocked you would've think I've just been shot,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
shot down right now in this spot.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
But too bad this doesn't happen a lot.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Oh girl you got me visualizing me on top,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
on top of your hot body while were sweating a lot,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
a lot of time on the clock before we have to stop.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
too bad shes not into that stuff a lot,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
oh man shes really super hot,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
hotter than the sun that's right on top,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Oh man there she goes i had to stop,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
and ask her some questions that i had in stock.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
She said she want's to take it slow,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
i'm not that type of guy ill let cha know,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
when i see that red light all i know is go,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
so baby girl lets do this on the floor.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
show me your moves that makes you such a pro


*the flag is supahotfire*

[Finished in 14.2s]
```

Nous avons notre Flag: supahotfire.