04Reverse基础（五）

原创

beloved_fjq 于 2020-05-30 17:49:42 发布 452 收藏 1

04Reverse基础（五）

任务内容

1.完成攻防世界 REVERSE 新手区 12道题目并编写WriteUp。（网上有其他人写的WP，可以参考着学习，做完这些题再回过头做之前的四题）
2.学习密码学知识，推荐阅读书籍《图解密码学》，阅读第一章、第二章
3.阅读书籍《程序员的自我修养》

学习笔记

一、《图解密码学》

第一章 环游密码世界
1.2密码
Alice与Bob

表 1-1　本书中的主要角色一览

名称	说明
Alice	一般角色
Bob	一般角色
Eve	窃听者，可窃听通信内容
Mallory	主动攻击者，可妨碍通信、伪造消息等
Trent	可信的第三方
Victor	验证者

1.3对称密码与公钥密码
1.对称密钥是指在加密与解密时使用同一密钥的方式
2.公钥密码（非对称密钥）是指在加密与解密时使用不同密钥的方式
3.混合密码系统：二者结合
1.4其他密码技术

1.4其他密码技术
（1）单向散列函数：通过散列值是否和真正文件相同来验证下载的文件是否相同，是否被篡改，是否完整，而不是机密
（2）消息认证码：提供认证机制
（3）数字签名：防止伪装、篡改、否认
（4）伪随机数生成器：密钥生成
1.5密码学的工具箱

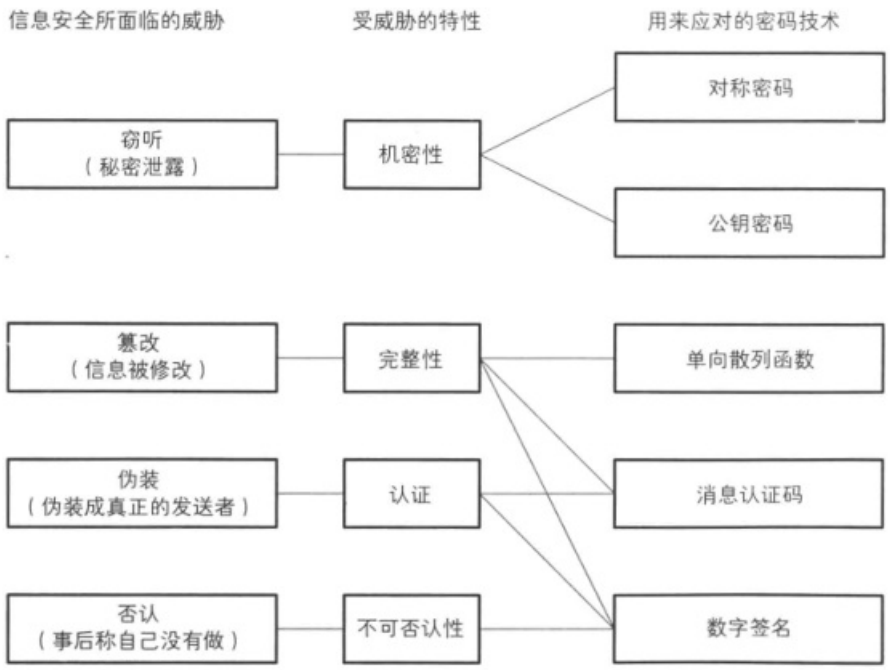


图 1-8　信息安全所面临的威胁与应对这些威胁的密码技术

1.6隐写术与数字水印
1.隐写术：可以隐藏消息本身，若搞清楚了嵌入消息的方法，则可以搞清消息的内容。
2.数字水印：运用隐写术的技术，单凭此不可以进行保密。
3.结合：将要嵌入的文章进行加密生成密文，再通过隐写术将密文隐藏到图片里。
4.实质：密码隐藏的是内容，隐写术隐藏的是消息本身
1.7密码与信息安全常识
1.不要使用保密的密码算法（隐蔽式安全性危险愚蠢）
2.使用低强度的密码比不进行任何加密更危险
3.任何密码总有一天都会破解（严格说：绝对不会被破解的密码是存在的，叫做一次性密码本，但是现实不可用；另一种被认为可能造就完美的密码技术叫量子密码）
4.密码指示信息安全的一部分：还可以通过社会工程学
第二章 历史上的密码
2.2恺撒密码
1.恺撒密码是将明文中所使用的字母表按照一定的字数平移来加密的。
2.暴力破解：所有可能的密钥每一种都试一遍。
2.3简单替换密码
1.简单替换密码就是字母之间有一种一一对应的关系。
2.简单替换密码很难通过暴力破解来破译，因为密钥数量过多，一种密码能够使用的所有密钥的集合称为密钥空间。
3.频率分析可以破译简单替换密码：
（1）首先统计密文中每个字母出现的频率
（2）再找到英语中单词或字母出现的频率
（3）二者对应替换
一些结论：
（1）高频和低频字母都可作为线索
（2）开头和结尾可作为线索，还有单词之间的分隔

（3）密文越长越容易破译
（4）同一个字母连续出现能够成为线索
（5）破译的速度会越来越快
2.4Enigma
1.它使用可以完成加解密两种操作的工具。
2.构造：

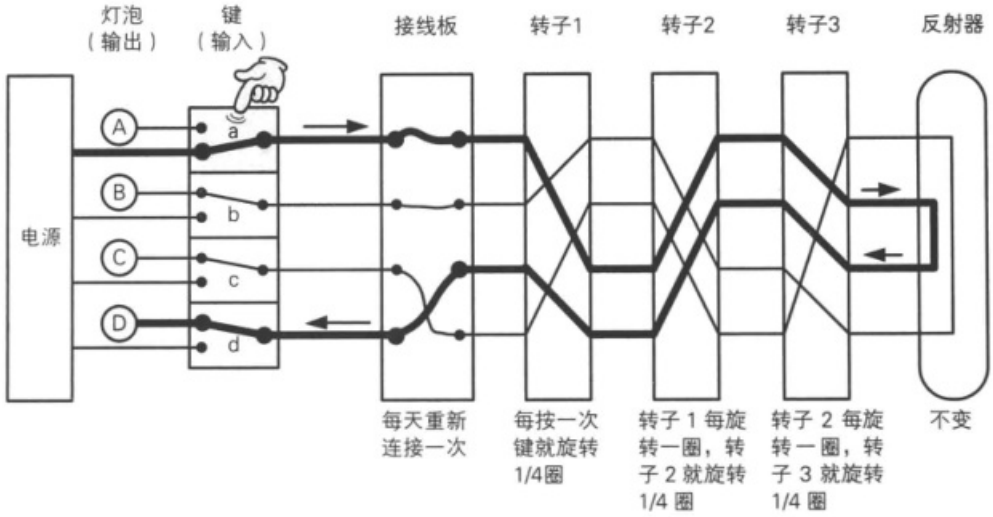


图 2-6　Enigma 的构造（只有 4 个字母的情况）

接线板：改变接线方式来改变字母对应关系
3.加密方式：

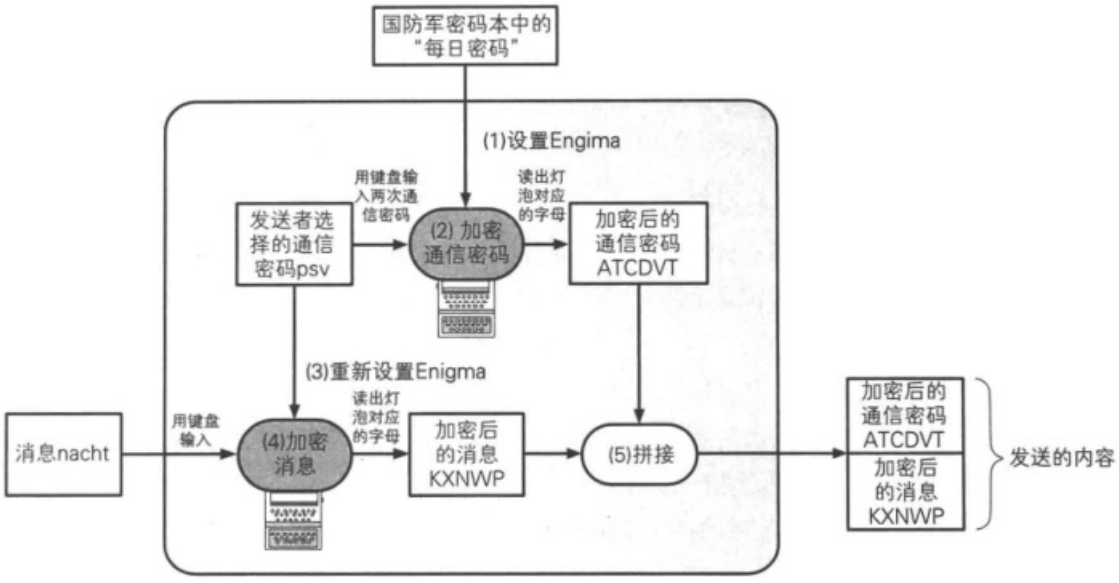


图 2-8　用 Enigma 加密 nacht

（1）设置Engima：每日密码对接线，转子进行排列
（2）加密通信密码
（3）重新设置Engima
（4）加密消息
（5）拼接：通信密码+加密后的消息拼接
注意：每日密码与通信密码是两种不同的密钥：每日密码不是用来加密消息的，而是用来加密通信密码的，即加密密钥的密钥（称为密钥加密密钥，两重加密）。即每日密码加密通信密码，通信密码加密消息。

4.解密方式

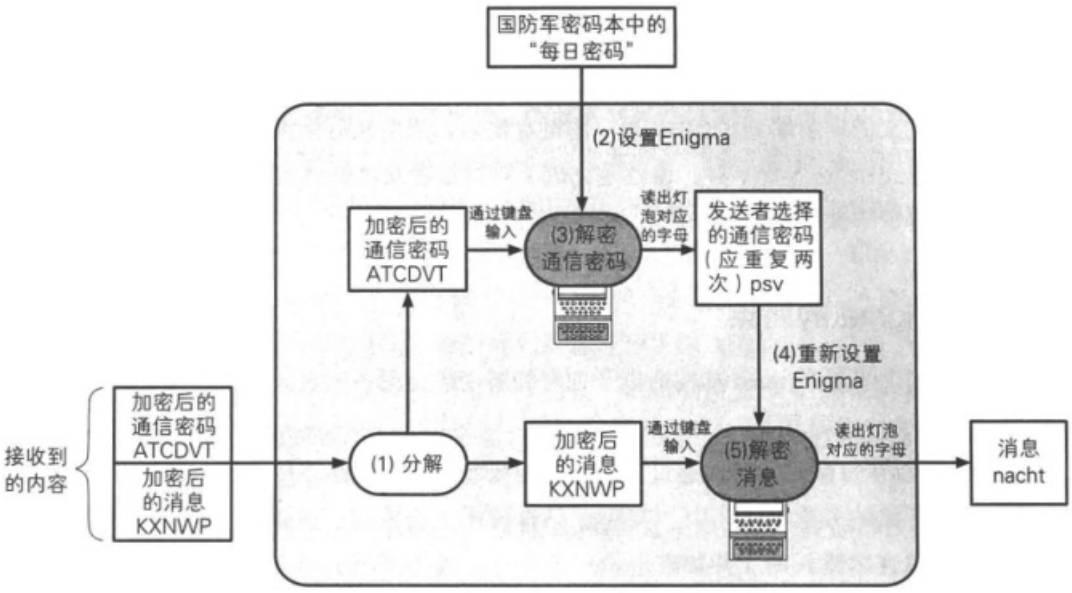


图 2-9　用 Enigma 解密

（1）分解成两部分：通信密码+消息
（2）设置Engima
（3）解密通信密码
（4）重新设置Engima
（5）解密消息

5.弱点：
（1）只有一个转子会旋转
（2）将通信密码连续输入两次并加密
（3）通信密码是人为选定的
（4）必须派发国防军密码本1

6.破译
艰难：因为设计不依赖于隐蔽式安全性：只要不知道Engima的设置（密钥），就无法破译。

2.5思考为何将密码算法与密钥分开

恺撒密码

密码算法：将明文中的各个字母按照指定的字母数平移
密钥：平移的字母数量

简单替换密码

密码算法：按照替换表对字母表进行替换
密钥：替换表

Enigma（通信密码的加密）

密码算法：使用 Enigma 密码机，通过接线板的接线方式、三个转子的顺序、每个转子的旋转位置对字母进行替换
密钥（每日密码）：接线板的接线方式、三个转子的顺序、每个转子的旋转位置

Enigma（通信电文的加密）

密码算法：使用接线板的接线方式和三个转子的顺序固定的 Enigma 密码机，按照每个转子

密码算法：使用接线板的接线方式和三个转子的顺序固定的 Enigma 密码机，按照每个转子的旋转位置对字母进行替换

密钥（通信密码）：每个转子的旋转位置

答案：希望重复使用，并且不会增加风险。

二、《程序员的自我修养》

6.操作系统的两个功能（尽可能的发挥CPU Memory IO的潜力）：
（1）提供抽象的接口
（2）管理硬件资源
①CPU：由分时操作系统转向实时操作系统的发展；
②内存管理：（要解决的问题是如何将计算机上有限的物理资源分配给多个应用程序来使用），由直接使用物理内存再到虚拟地址空间（控制虚拟地址到物理地址的映射过程）的出现，再到分段–分页–段页式等一系列的发展；
③IO：设备驱动的发展，应用程序员是不需要与硬件直接打交道的，在如今的操作系统中，硬件被抽象成一系列概念，在Unix中，硬件也被抽象成文件，所有繁琐的硬件交互都是由硬件驱动程序来完成的。

任务实现

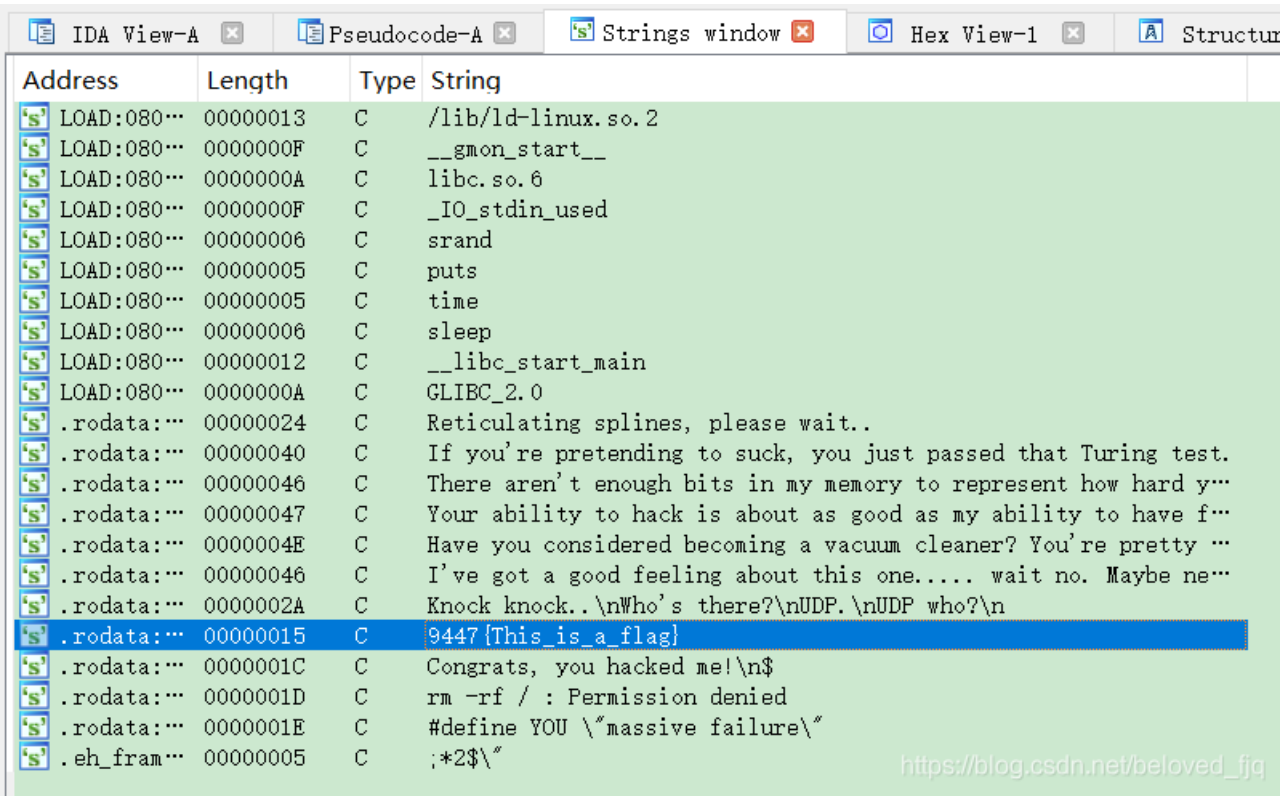
一、IDA补充学习

1.关于静态分析和动态分析：
静态分析就是通过浏览程序代码来理解程序的行为，查看的就是反汇编之后的代码。
动态分析是指在严格控制的环境（沙盒）中执行恶意软件，并使用系统检测实用工具记录其所有行为。
2.关于交叉引用：查看字符串（shift+12），然后按“x”看看还在哪些其他地方出现过，而后可以点进去用F5反汇编，反汇编代码中的数字可以用“r”（查看ASCⅡ值）和“h”相互切换（查看十六进制），“n”用于重命名（也可还原：用过n之后不进行输入就按ok）
3.搜索字符串：Alt+T进行搜索
4.代码段数据段
5.跳转地址：用“G”
6.如何到main函数：ctrl+E，main函数一定之前有个CommandLine，有三个参数（argc，argv，envp环境变量）

二、攻防世界 REVERSE 新手区 12道题目并编写WriteUp

001 open-source

直接找字符串strings：



双击进去：

```
.rodata:08048788 a9447ThisIsAFla db '9447{This_is_a_flag}',0
.rodata:08048788                                         ; DATA XREF: .data:strs↓o
.rodata:0804879D aCongratsYouHac db 'Congrats, you hacked me!',0Ah
.rodata:0804879D                                         ; DATA XREF: .data:080499C4↓o
.rodata:0804879D                 db '$ ',0
```

找到答案！

002 simple-unpack

看标题就知道这题是要脱壳。

先查看一下是什么壳，winHex打开看到：

```
000000000044F1C0  6E 66 6F 3A 20 54 68 69  73 20 66 69 6C 65 20 69  nfo:·This·file·i
000000000044F1D0  73 20 70 61 63 6B 65 64  20 77 69 74 68 20 74 68  s·packed·with·th
000000000044F1E0  65 20 55 50 58 20 65 78  65 63 75 74 61 62 6C 65  e·UPX·executable
000000000044F1F0  20 70 61 63 6B 65 72 20  68 74 74 70 3A 2F 2F 75  ·packer·http://u
000000000044F200  70 78 2E 73 66 2E 6E 65  74 20 24 0A 00 24 49 64  px.sf.net·$..$Id
000000000044F210  3A 20 55 50 58 20 33 2E  39 31 20 43 6F 70 79 72  :·UPX·3.91·Copyr
```

因此先脱壳：

003 logmein

将文件拖IDA，shift+F12

第一次进You entered the correct password!\nGreat job!\n，发现反编译出来的函数没用，所以第二次进Enter your guess：

```c
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
    if (argc != 4) {
     printf("what?\n");
     exit(1);
    }
    unsigned int first = atoi(argv[1]);
    if (first != 0xcafe) {
     printf("you are wrong, sorry.\n");
     exit(2);
    }
    unsigned int second = atoi(argv[2]);
    if (second % 5 == 3 || second % 17 != 8) {
     printf("ha, you won't get it!\n");
     exit(3);
    }
    if (strcmp("h4cky0u", argv[3])) {
     printf("so close, dude!\n");
     exit(4);
    }
    printf("Brr wrrr grr\n");
    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;
    printf("Get your key: ");
    printf("%x\n", hash);
    return 0;
}
```

解题：

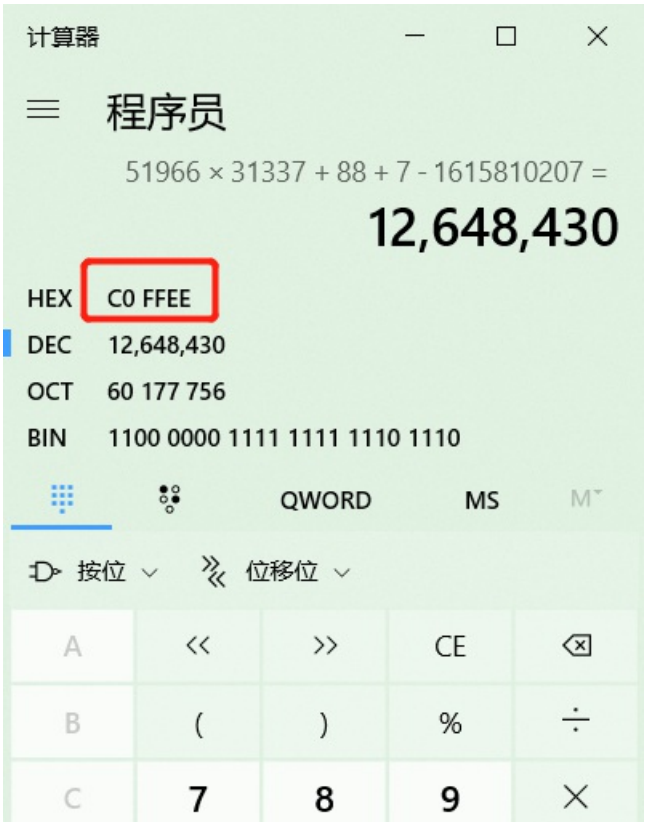
hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207

first=0xcafe：十进制是51966

second % 17=8

argv3='h4cky0u'

注意 printf("%x\n", hash);是输出十六进制



答案：c0ffee

004 insanity

```
void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
{
  size_t v3; // rsi
  int i; // [rsp+3Ch] [rbp-54h]
  char s[36]; // [rsp+40h] [rbp-50h]
  int v6; // [rsp+64h] [rbp-2Ch]
  __int64 v7; // [rsp+68h] [rbp-28h]
  char v8[8]; // [rsp+70h] [rbp-20h]
  int v9; // [rsp+8Ch] [rbp-4h]

  v9 = 0;
  strcpy(v8, ":\"AL_RT^L*.?+6/46");
  v7 = 'ebmarah';
  v6 = 7;
  printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
  printf("To continue, you must enter the correct password.\n");
  printf("Enter your guess: ");
  __isoc99_scanf("%32s", s);
  v3 = strlen(s);
  if ( v3 < strlen(v8) )
    sub_4007C0();
  for ( i = 0; i < strlen(s); ++i )
  {
    if ( i >= strlen(v8) )
      sub_4007C0();
    if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
      sub_4007C0();
  }
  sub_4007F0();
}
```


注意：

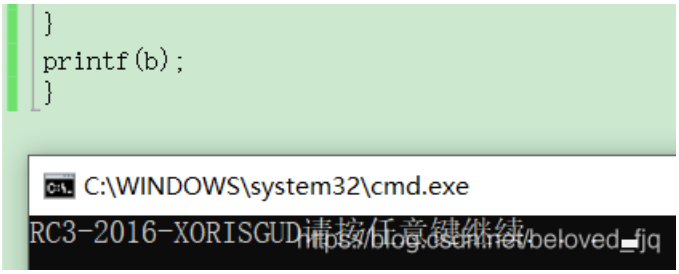
(_BYTE *)&v7的意思是，把longlong型的v7强制转化为byte型的地址，简单的说，就是把它看成字符串（C语言字符串本质都是指针首地址+偏移）；所以我们用先用v7的值10进制转16进制，然后16进制转文本得到：ebmarah；但是在机器虚拟化内存后，规定地址排列规则时使用了小端法（最低有效字节在前面）。因此我们真正的解码文本应该是把上面的答案倒过来写：harambe；

ord函数可以将字符转化为你所需要的ASCII码,

将代码写成C：

```
#include<stdio.h>
int main()
{
long long a = 28537194573619560;
char * p = (char*)&a;
char b[] = ":\"AL_RT^L*.?+6/46";
for(int i = 0;b[i]!=0;i++)
{
    b[i] = b[i]^p[i%7];
```

```
}
printf(b);
}
```



005 python-trade

文件发现是一个.pyc文件，用python反编译在线工具反编译这个.pyc文件得到源码：

```python
#!/usr/bin/env python
# encoding: utf-8
# 如果觉得不错，可以推荐给你的朋友！http://tool.lu/pyc
import base64

def encode(message):
    s = ''
    for i in message:
        x = ord(i) ^ 32
        x = x + 16
        s += chr(x)

    return base64.b64encode(s)

correct = 'XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
flag = ''
print 'Input flag:'
flag = raw_input()
if encode(flag) == correct:
    print 'correct'
else:
    print 'wrong'
```


注意：ord()函数是Python的内置函数，是chr()函数的配对函数，以一个字符作为参数，返回对应的Unicode数值。

关键点：encode(flag) == “XlNkVmtUI1MgXWBZXCFeKY+AaXNt”

006 game

直接拖到IDA打开，shift+F12，Alt+T（搜索字符串），搜索： flag

直接跳出来：done!!!the flag is，双击，跳到IDA View-A，Ctrl+X（交叉引用）

F5（生成伪代码）

```
sub_45A7BE("done!!! the flag is ");
 v59 = 18;
 v60 = 64;
 v61 = 98;
 v62 = 5;
 v63 = 2;
 v64 = 4;
 v65 = 6;
 v66 = 3;
 v67 = 6;
 v68 = 48;
```

```
v69 = 49;
v70 = 65;
v71 = 32;
v72 = 12;
v73 = 48;
v74 = 65;
v75 = 31;
v76 = 78;
v77 = 62;
v78 = 32;
v79 = 49;
v80 = 32;
v81 = 1;
v82 = 57;
v83 = 96;
v84 = 3;
v85 = 21;
v86 = 9;
v87 = 4;
v88 = 62;
v89 = 3;
v90 = 5;
v91 = 4;
v92 = 1;
v93 = 2;
v94 = 3;
v95 = 44;
v96 = 65;
v97 = 78;
v98 = 32;
v99 = 16;
v100 = 97;
v101 = 54;
v102 = 16;
v103 = 44;
v104 = 52;
v105 = 32;
v106 = 64;
v107 = 89;
v108 = 45;
v109 = 32;
v110 = 65;
v111 = 15;
v112 = 34;
v113 = 18;
v114 = 16;
v115 = 0;
v2 = 123;
v3 = 32;
v4 = 18;
v5 = 98;
v6 = 119;
v7 = 108;
v8 = 65;
v9 = 41;
v10 = 124;
v11 = 80;
v12 = 125;
v13 = 38;
v14 = 124;
```

```
  v14 = 124;
  v15 = 111;
  v16 = 74;
  v17 = 49;
  v18 = 83;
  v19 = 108;
  v20 = 94;
  v21 = 108;
  v22 = 84;
  v23 = 6;
  v24 = 96;
  v25 = 83;
  v26 = 44;
  v27 = 121;
  v28 = 104;
  v29 = 110;
  v30 = 32;
  v31 = 95;
  v32 = 117;
  v33 = 101;
  v34 = 99;
  v35 = 123;
  v36 = 127;
  v37 = 119;
  v38 = 96;
  v39 = 48;
  v40 = 107;
  v41 = 71;
  v42 = 92;
  v43 = 29;
  v44 = 81;
  v45 = 107;
  v46 = 90;
  v47 = 85;
  v48 = 64;
  v49 = 12;
  v50 = 43;
  v51 = 76;
  v52 = 86;
  v53 = 13;
  v54 = 114;
  v55 = 1;
  v56 = 117;
  v57 = 126;
  v58 = 0;
  for ( i = 0; i < 56; ++i )
  {
    *(&v2 + i) ^= *(&v59 + i);
    *(&v2 + i) ^= 0x13u;
  }
  return sub_45A7BE("%s\n");
}
```

解密：

```
#v2：原代码v2-v58的值
v2 = [123,32,18,98,119,108,65,41,124,80,125,38,124,111,74,49,83,108,94,108,84,6,96,83,44,121,104,110,32,95,117,1
01,99,123,127,119,96,48,107,71,92,29,81,107,90,85,64,12,43,76,86,13,114,1,117,126,0]

#v59：原代码v59-v115的值
v59 = [18,64,98,5,2,4,6,3,6,48,49,65,32,12,48,65,31,78,62,32,49,32,1,57,96,3,21,9,4,62,3,5,4,1,2,3,44,65,78,32,1
6,97,54,16,44,52,32,64,89,45,32,65,15,34,18,16,0]

s = ""

for i in range(57):
    v2[i] = v2[i] ^ v59[i]
    v2[i] = v2[i] ^ 19
    s += chr(v2[i])

print(s)
```

得到flag：zsctf{T9is_tOpic_1s_v5ry_int7resting_b6t_others_are_n0t}

007 Hello CTF

发现找不到flag的字符串，但是有wrong和success，双击进IDA View-A，Ctrl+X，F5:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int v3; // ebx
  char v4; // al
  int result; // eax
  int v6; // [esp+0h] [ebp-70h]
  int v7; // [esp+0h] [ebp-70h]
  char v8; // [esp+12h] [ebp-5Eh]
  char v9[20]; // [esp+14h] [ebp-5Ch]
  char v10; // [esp+28h] [ebp-48h]
  __int16 v11; // [esp+48h] [ebp-28h]
  char v12; // [esp+4Ah] [ebp-26h]
  char v13; // [esp+4Ch] [ebp-24h]

  strcpy(&v13, "437261636b4d654a757374466f7246756e");
  while ( 1 )
  {
    memset(&v10, 0, 0x20u);
    v11 = 0;
    v12 = 0;
    sub_40134B(aPleaseInputYou, v6);
    scanf(aS, v9);
    if ( strlen(v9) > 0x11 )
      break;
    v3 = 0;
    do
    {
      v4 = v9[v3];
      if ( !v4 )
        break;
      sprintf(&v8, asc_408044, v4);
      strcat(&v10, &v8);
      ++v3;
    }
    while ( v3 < 17 );
    if ( !strcmp(&v10, &v13) )
      sub_40134B(aSuccess, v7);
    else
      sub_40134B(aWrong, v7);
  }
  sub_40134B(aWrong, v7);
  result = stru_408090._cnt-- - 1;
  if ( stru_408090._cnt < 0 )
    return _filbuf(&stru_408090);
  ++stru_408090._ptr;
  return result;
}
```

注意这几句代码：

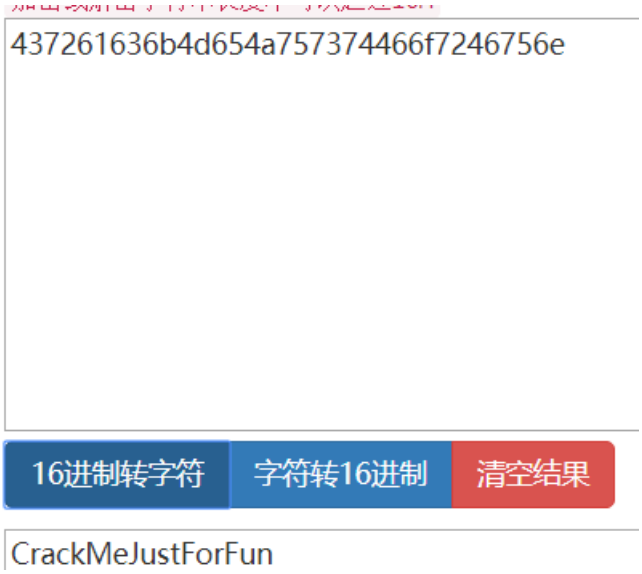
```
strcpy(&v13, "437261636b4d654a757374466f7246756e");

scanf(aS, v9);
if ( strlen(v9) > 0x11 )
      break;

sprintf(&v8, asc_408044, v4);
strcat(&v10, &v8);

if ( !strcmp(&v10, &v13) )
      sub_40134B(aSuccess, v7);
```

逻辑推理：
v9为我们的输入，长度≤17，而v8就是v9，v10清零加上v8就是v9
v10（=输入v9）和v13进行比较。相同就success
输入必须就是v13这个字符，用16进制转字符串得到flag：CrackMeJustForFun
注意：
（1）strcat函数2113的作用是将两个char类型连接；
（2）void *memset(void *s, int ch, size_t n);
函数解释：将s中当前位置后面的n个字节 （typedef unsigned int size_t ）用 ch 替换并返回 s 。
memset：作用是在一段内存块中填充某个给定的值，它是对较大的结构体或数组进行清零操作的一种最快方法 [1] 。
memset()函数原型是extern void *memset(void *buffer, int c, int count) buffer：为指针或是数组,c：是赋给buffer的值,count：是buffer的长度.
（3）sprintf(&v8, asc_408044, v4)这句话的意思是：送格式化输出到字符串中，即v4是输入的每一个字符，asc_408044是"%x"，将字符串转换为16进制存入v8，那么反过来就是这个437261636b4d654a757374466f7246756e是转换好的16进制，我们要找到原始字符串：
而后：十六进制转字符串：

下面编写求v13的python脚本

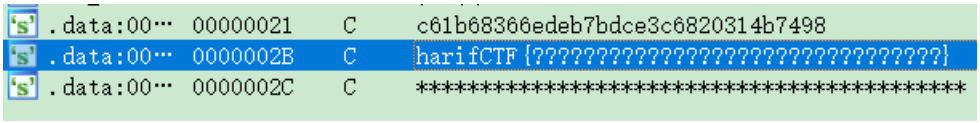
```python
a = [0x43,0x72,0x61,0x63,0x6b,0x4d,0x65,0x4a,0x75,0x73,0x74,0x46,0x6f,0x72,0x46,0x75,0x6e]
c = ""
for x in a:
    c += chr(x)
print(c)
```

或者直接用网上的：



008 getit

先找到一个像flag的字符串：



而后正常用F5：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v3; // al
  __int64 v5; // [rsp+0h] [rbp-40h]
  int i; // [rsp+4h] [rbp-3Ch]
  FILE *stream; // [rsp+8h] [rbp-38h]
  char filename[8]; // [rsp+10h] [rbp-30h]
  unsigned __int64 v9; // [rsp+28h] [rbp-18h]

  v9 = __readfsqword('(');
  LODWORD(v5) = 0;
  while ( (signed int)v5 < strlen(s) )
  {
    if ( v5 & 1 )
      v3 = 1;
    else
      v3 = -1;
    *(&t + (signed int)v5 + 10) = s[(signed int)v5] + v3;
    LODWORD(v5) = v5 + 1;
  }
  strcpy(filename, "/tmp/flag.txt");
  stream = fopen(filename, "w");
  fprintf(stream, "%s\n", u, v5);
  for ( i = 0; i < strlen(&t); ++i )
  {
    fseek(stream, p[i], 0);
    fputc(*(&t + p[i]), stream);
    fseek(stream, 0LL, 0);
    fprintf(stream, "%s\n", u);
  }
  fclose(stream);
  remove(filename);
  return 0;
}
```

重点代码：

下面是重点代码

```
LODWORD(v5) = 0;
  while ( (signed int)v5 < strlen(s) )
  {
    if ( v5 & 1 )
      v3 = 1;
    else
      v3 = -1;
    *(&t + (signed int)v5 + 10) = s[(signed int)v5] + v3;
    LODWORD(v5) = v5 + 1;
  }
```


可以看出将s中的值赋给了t，然后最后是将t写入文件flag.txt中，我们看一下s和t分别是什么：

```
.data:0000000006010A0 s               db 'c61b68366edeb7bdce3c6820314b7498',0
.data:0000000006010A0                                 ; DATA XREF: main+25↑o
.data:0000000006010A0                                 ; main+3F↑r
.data:0000000006010C1                 align 20h
.data:0000000006010E0                 public t
.data:0000000006010E0 ; char t
.data:0000000006010E0 t               db 53h          ; DATA XREF: main+65↑w
.data:0000000006010E0                                 ; main+C9↑o ...
.data:0000000006010E1 aHarifctf       db 'harifCTF{??????????????????????????????}',0
.data:000000000060110C                align 20h
```


最后写出代码：

```
v5 = 0
s = 'c61b68366edeb7bdce3c6820314b7498'
t = ['S','h','a','r','i','f','C','T','F','{','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?'
,'?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','?','}']
v3 = 0
l = len(s)
while(v5 < l):
    if( v5 & 1 ):
        v3 = 1
    else:
        v3 = -1
    t[10+v5] = chr(ord(s[v5])+v3)
    v5 += 1
flag = ''
for x in t:
    flag+=x
print(flag)
```

得到flag：SharifCTF{b70c59275fcfa8aebf2d5911223c6589}