# Lab 1: ELF逆向分析

实验目的：熟悉Linux可执行文件的格式和调试方法。

实验环境：Ubuntu 16.04 LTS 实验工具：GCC、IDA Pro，GDB 实验目的：熟悉使用IDA Pro静态分析ELF文件 实验代码：

```
#include  <stdio.h>
int main(int argc,char **argv,char **env)
{
    char encrypted_text[] = "\x04\x0d\x16\x1e\x1b\x0b\x1c\x1d\x16\x16\x15";
    char flag[15];
    memset(flag,0,15);
    puts("please input the flag:");
    scanf("%11s",flag);
    for(int i=0;i<11;i++)
    {
        encrypted_text[i] ^= flag[i];
    }
    for (int i=0;i<11;i++)
    {
        encrypted_text[i] = (((encrypted_text[i] - 0x61) + 11)%26) + 0x61;
    }
    if(strcmp(encrypted_text,"qwertyuiopa")==0)
    {
        puts("flag is correct!");
    }
    else
    {
        puts("wrong!");
    }
    return 0;
}
```

1. 使用gcc编译源代码并执行观察输入输出

   ```
   gcc flag.c -o flag
   ./flag
   ```

2. 使用IDA Pro分析可执行文件并使用IDA Pro的hex ray插件(f5)进行反编译

```
.text:0000000000400686 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:0000000000400686 public main
.text:0000000000400686 main proc near                          ; DATA XREF: _start+1D↑o
.text:0000000000400686
.text:0000000000400686 var_58= qword ptr -58h
.text:0000000000400686 var_50= qword ptr -50h
.text:0000000000400686 var_44= dword ptr -44h
.text:0000000000400686 var_38= dword ptr -38h
.text:0000000000400686 var_34= dword ptr -34h
.text:0000000000400686 s1= byte ptr -30h
.text:0000000000400686 var_28= dword ptr -28h
.text:0000000000400686 s= byte ptr -20h
.text:0000000000400686 var_8= qword ptr -8
.text:0000000000400686
.text:0000000000400686 ; __unwind {
.text:0000000000400686 push    rbp
.text:0000000000400687 mov     rbp, rsp
.text:000000000040068A sub     rsp, 60h
.text:000000000040068E mov     [rbp+var_44], edi
.text:0000000000400691 mov     [rbp+var_50], rsi
.text:0000000000400695 mov     [rbp+var_58], rdx
.text:0000000000400699 mov     rax, fs:28h
.text:00000000004006A2 mov     [rbp+var_8], rax
.text:00000000004006A6 xor     eax, eax
.text:00000000004006A8 mov     rax, 1D1C0B1B1E160D04h
.text:00000000004006B2 mov     qword ptr [rbp+s1], rax
.text:00000000004006B6 mov     [rbp+var_28], 151616h
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int i; // [rsp+28h] [rbp-38h]
  signed int j; // [rsp+2Ch] [rbp-34h]
  char s1[8]; // [rsp+30h] [rbp-30h]
  int v7; // [rsp+38h] [rbp-28h]
  char s[24]; // [rsp+40h] [rbp-20h]
  unsigned __int64 v9; // [rsp+58h] [rbp-8h]

  v9 = __readfsqword(0x28u);
  *(_QWORD *)s1 = 2097563737544592644LL;
  v7 = 1381910;
  memset(s, 0, 0xFuLL);
  puts("please input the flag:");
  __isoc99_scanf("%11s", s);
  for ( i = 0; i <= 10; ++i )
    s1[i] ^= s[i];
  for ( j = 0; j <= 10; ++j )
    s1[j] = (s1[j] - 86) % 26 + 97;
  if ( !strcmp(s1, "qwertyuiopa") )
    puts("flag is correct!");
  else
    puts("wrong!");
  return 0;
}
```

3. 分析程序中分析的算法并试解出正确flag。
4. 尝试使用gdb动态分析程序的输入如何被处理。