难度：入门级

实践内容：

这次实践是为逆向工程零基础的学生准备的，目标是通过使用 IDA pro 等反汇编工具，猜测出 crackme 的口令，从而通过 crackme 的测试。

Crackme 字面意思是“破解我”，是人工构造用于练习破解的小程序。这次实践需要有汇编语言基础，以及函数调用时参数在栈的位置等知识。通过这次实践，你能够熟悉 IDA pro 反汇编工具，对逆向工程达到一定的了解，体验其中的乐趣。

问题：

对给定的两个 crackme，通过使用 IDA pro 工具或者其他调试工具，猜测出其口令，通过其测试。

1．计算两个 crackme 的 md5 值，检查文件完整性。

crackme1: 4357BF8C20FABB642FAFC0B73FB0ABFB

crackme2: 46AA577172D8F37AE2AE281515C99AFB

2．使用 file 命令查看 crackme 的文件类型。

Linux 环境下自带 file 命令，

Windows 版本可以去 http://gnuwin32.sourceforge.net/packages/file.htm 下载

3．对于每个 crackme，尝试运行，根据提示信息，使用工具，猜测出口令，通过测试。

分析与解答：

1． 检查文件完整性

下载两个 crackme，计算其 md5 值，验证文件的完整性。

crackme1: 4357BF8C20FABB642FAFC0B73FB0ABFB

crackme2: 46AA577172D8F37AE2AE281515C99AFB

2． 使用 file 命令查看 crackme 的文件类型。

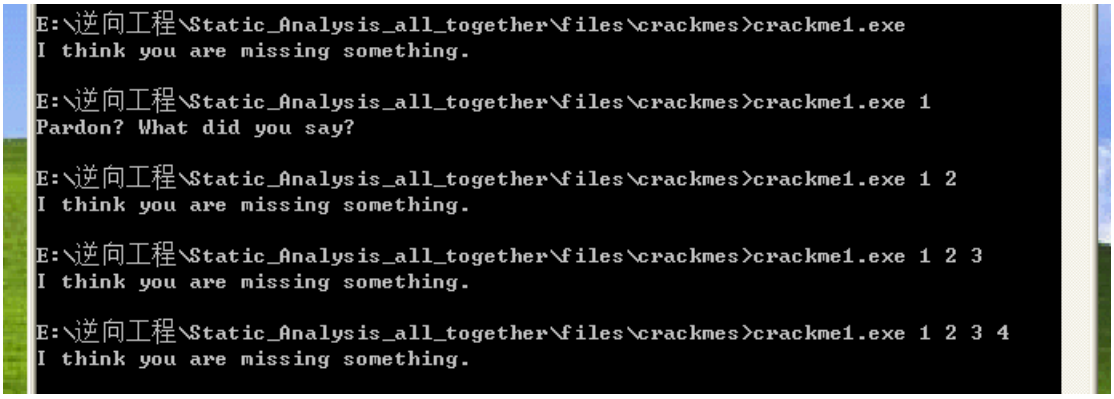
```
E:\逆向工程\Static_Analysis_all_together\files\crackmes>file crackme1.exe
crackme1.exe; PE32 executable for MS Windows (console) Intel 80386 32-bit

E:\逆向工程\Static_Analysis_all_together\files\crackmes>file crackme2.exe
crackme2.exe; PE32 executable for MS Windows (console) Intel 80386 32-bit
```

可以知道这两个 crackme 都是 32 位 windows 下 PE 文件，没有图形界面，是命令行程序。

3．

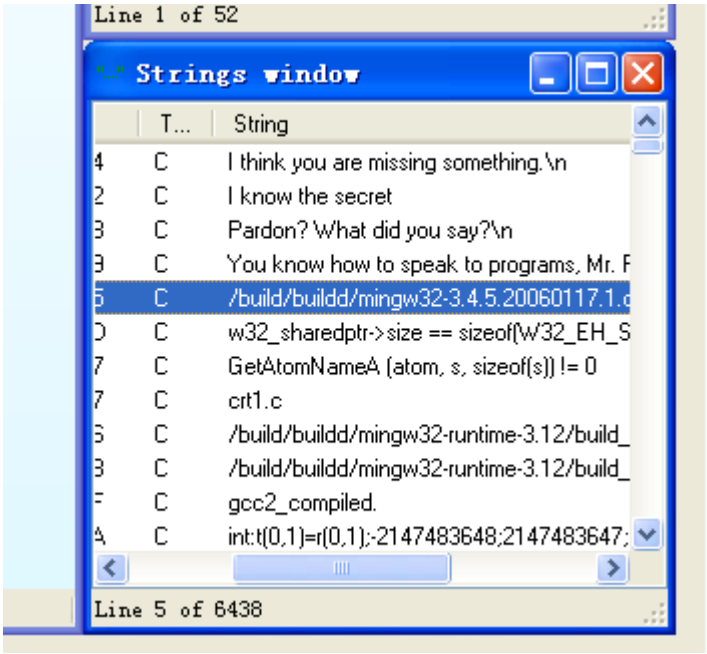
首先，我们先对 crackme1.exe 进行破解。

一开始是尝试运行该程序，试探其输入格式。



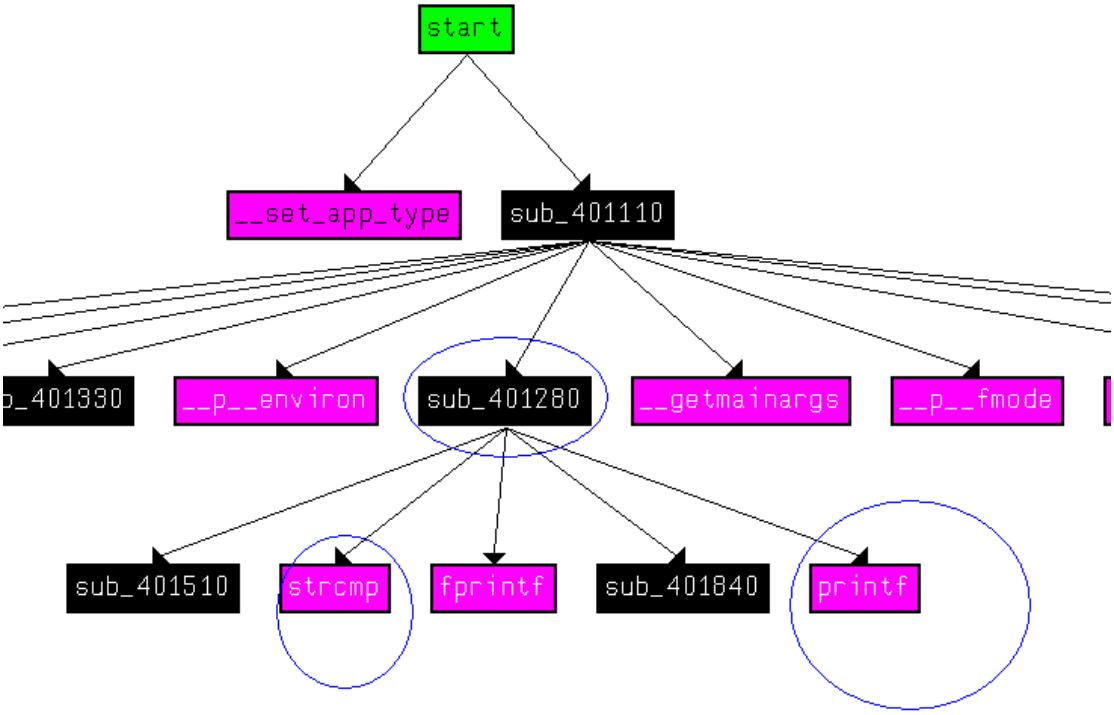
我们可以发现，在接受 1 个参数时，程序的反馈与众不同，所以我们猜测该程序接受一个参数。目前为止，我们已经发现了程序的两种反馈信息。一种是“I think you are missing something.”，这个猜测是参数数目不对的提示；另一种是“Pardon? What did you say?”，对于这种反馈信息，我们猜测是参数错误的提示。

接着我们使用 IDA pro 工具来打开文件，尝试阅读其汇编语言，验证我们的猜想。



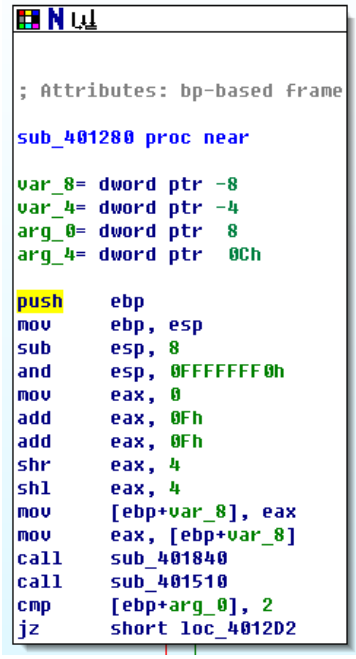
通过 Strings 页面可以查看到该程序中出现的明文字符串，我们发现了前面的两种反馈信息，“I think you are missing something.” “Pardon? What did you say?”，还发现了“I know the secret”和“You know how to speak to programs, Mr. Reverse-Engineer”这两个字符串。有内容我们可以猜测，前者就是我们需要的口令，后者就是输入口令正确时程序的反馈信息。

通过查看整个程序的 call flow，



我们可以得出结论，1 程序是用 C 语言写的，2 程序估计是使用 strcmp 函数来比较口令的，3 关键的部分在 sub_401280 这里。

接下来我们开始查看 sub_401280 的汇编代码。

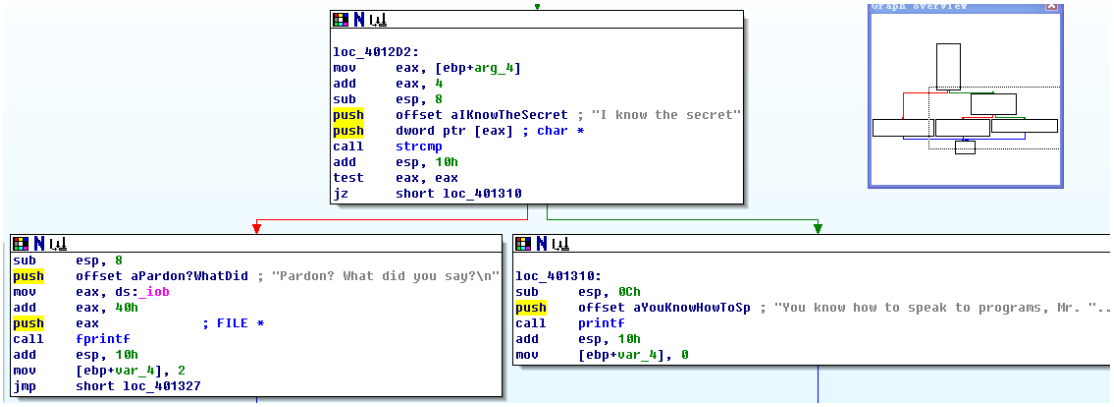


```
; Attributes: bp-based frame

sub_401280 proc near

var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 8
and     esp, 0FFFFFFF0h
mov     eax, 0
add     eax, 0Fh
add     eax, 0Fh
shr     eax, 4
shl     eax, 4
mov     [ebp+var_8], eax
mov     eax, [ebp+var_8]
call    sub_401840
call    sub_401510
cmp     [ebp+arg_0], 2
jz      short loc_4012D2
```

倒数第二行 cmp [ebo+arg_0],2 为判断程序是否有两个参数

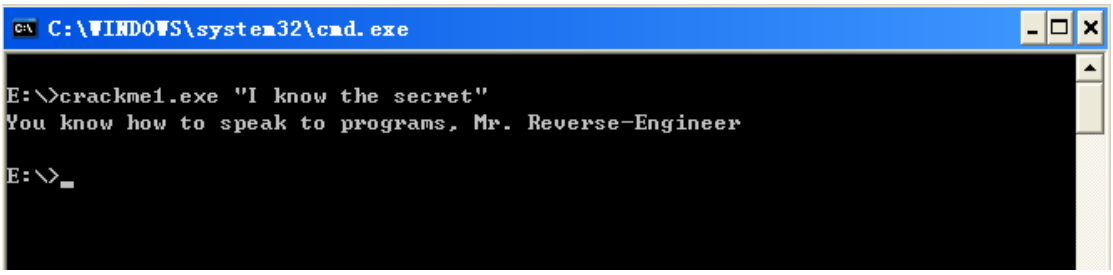
在 c 语言中，main 函数通常为 int main(int argc,const char **argv)，即第一个参数 argc 对应 argv 的大小，第二个参数对应命令行的格式。如在命令行输入 crackme1.exe 1，那么参数对应的值为 argc=2,argv={”crackme1.exe”,”1”}。

如果 argc=2，那么进行下一步判断

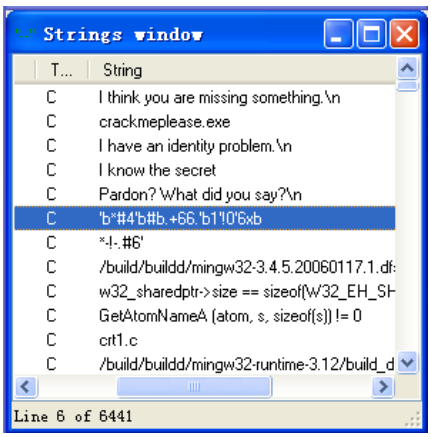


call strcmp 中，程序用“I know the secret”对应的字符串和[eax]对应的字符串（用户输入的口令）相比较，通过比较的结果反馈口令是否正确。

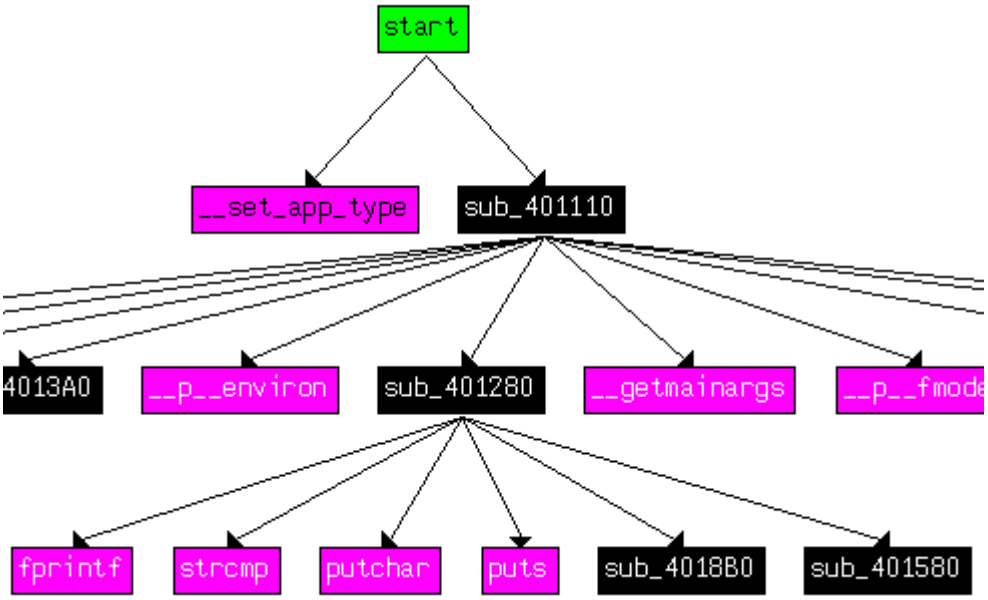
那么尝试输入口令 I know the secret，我们可以通过程序的测试。



crackme2



String 界面，明文字符串“I know the secret”和“crackmeplease.exe”



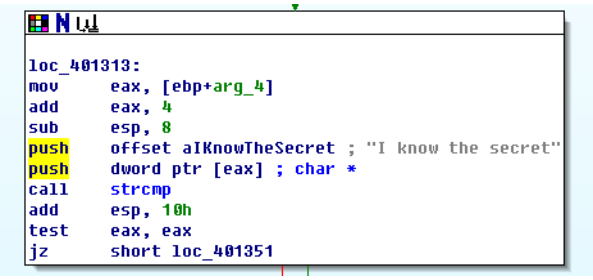
Function call，主要查看 sub_401280

```
; Attributes: bp-based frame

sub_401280 proc near

var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 18h
and     esp, 0FFFFFFF0h
mov     eax, 0
add     eax, 0Fh
add     eax, 0Fh
shr     eax, 4
shl     eax, 4
mov     [ebp+var_C], eax
mov     eax, [ebp+var_C]
call    sub_4018B0
call    sub_401580
cmp     [ebp+arg_0], 2
jz      short loc_4012D5
```

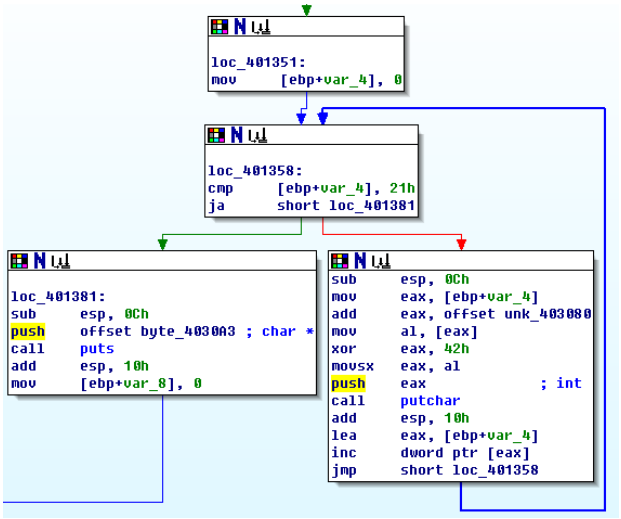
倒数第二行，cmp [ebp+arg_0],2 判断程序参数是否为两个

```
loc_4012D5:
mov     eax, [ebp+arg_4]
sub     esp, 8
push    offset aCrackmeplease_ ; "crackmeplease.exe"
push    dword ptr [eax] ; char *
call    strcmp
add     esp, 10h
test    eax, eax
jz      short loc_401313
```

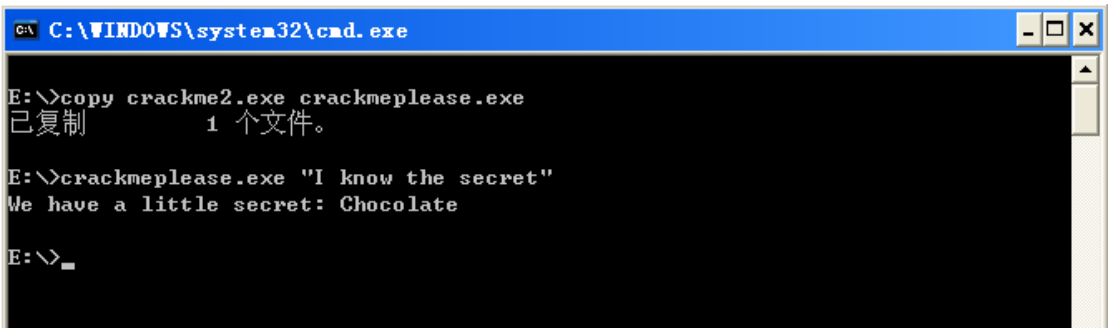
通过参数个数的判断后，接着用 strcmp 函数对 argc 里面的第一个字符串，即程序名，和“crackmeplease.exe”进行判断



通过程序名判断后，用户输入的口令与”I know the secret”判断。



通过口令判断后，通过一定规则输出通过测试的信息。
具体是 unk_403080 中的字符串分别与 0x42h 进行异或运算。



通过测试得到的反馈信息。