《汇编语言程序设计实验》

第七次上机实验任务

实验名称：实验七 跟踪与反跟踪程序设计

1. 实验目的和要求
2. 熟悉跟踪与反跟踪的技术；
3. 提升对计算机系统的理解与分析能力。
4. 实验内容

任务1：数据加密与反跟踪

在实验二的基础上，增加查询前输入密码的功能，密码不对则程序退出，只有密码正确之后才能完成后续的功能。密码采用密文的方式存放在数据段中。各科成绩也以密文方式存放在数据段中。加密方法自选。

可以采用计时、中断矢量表检查、堆栈检查、间接寻址等方式中的一种或多种方式反跟踪（建议只采用一种，最多两种反跟踪方法，重点是深入理解和运用好所选择的反跟踪方法）。

成绩表中要有编程者自己的名字（姓的全拼+名字的拼音首字母，但大小写可以随意组合）和各科成绩（姓名和成绩都密文存放）。成绩表中只需要定义三个学生的信息即可。

**提示：**为了使源程序的数据段中定义的密码、学生姓名、各科成绩能在汇编之后变成密文（也就是在最后交付出去的执行程序中看不到明文），可以使用数值运算符（参见教材P48）对变量的初始值进行变换。例如，如果想使语文成绩90分变成密文，加密算法是与密钥字符“W”做异或运算，则可写成：

YUWEN DB 90 XOR ‘W’

任务2：跟踪与数据解密

解密同组同学的加密程序，获取该同学的各科成绩。

**注意：**两人一组，每人实现一套自己选择的加密与反跟踪方法，把执行程序交给对方解密。如何设计反跟踪程序以及如何跟踪破解，是本次实验报告中重点需要突出的内容。

**上述任务中，重要操作环节（纳入到学生自己写的实验步骤中）提示：**

1. 若密码是用明文存放在数据段中的，如何更快地获取密码？（可替代的操作：每个学生可以把自己实验二的执行程序用二进制编辑工具打开，直接在文件里寻找所定义的学生姓名及其成绩）
2. 如何对密码实现暴力破解？（描述一下实现的具体思路即可）
3. 如何综合利用静态反汇编和动态反汇编的信息破解程序？（在体会里总结一下即可）
4. 举例说明如何观察到程序中存在反跟踪的代码?举例说明如何应对反跟踪程序? （在体会里具体描述）
5. 思考一下，如何用C语言（不嵌入汇编语言）实现反跟踪？是否能发现汇编语言的特殊之处? （在体会里具体描述）
6. 当存在修改中断矢量表的代码时，一般会先关掉中断（也即执行CLI指令）。如果不想因为关中断指令的出现让跟踪者容易判断出后续存在反跟踪代码，应如何设计修改中断矢量表的代码，达到不用关中断的目的？

**参考资料：**

关于跟踪与反跟踪、加密与解密已经有了很系统的理论与方法，其完整、深入的知识点不是本课程的内容，本次实验仅仅是以加解密为背景，目的是初步体会汇编语言在此领域中的作用。下面介绍的内容对大家实现本次实验的目的有一定的帮助作用。

1. **反汇编、跟踪与密码破解方法简介**

**1、暴力破解密码**

（1）对于密码以明文方式存放在待破解程序中的程序，可以利用文件编辑工具直接打开该待破解程序的文件并查看里面的内容，找到疑似密码的字符串，再通过实际测试，最终确定真实的密码。

（2）对于仅仅需要输入密码且没有出错次数限制的程序，可以采用暴力破解密码的方法来获取密码，也即编写一个程序，自动调用与执行待破解的程序，并按照枚举方式自动给待破解程序输入可能的密码，直至密码正确为止。

**2、静态反汇编**

利用反汇编工具直接处理待破解程序的执行文件，将执行文件翻译成比较直观易读的汇编语言的源程序（甚至C语言的源程序），再由人工阅读该源程序，可以找出加密算法（从而推出真实的密码），或者找到密码比较正确之后的程序入口，再用二进制文件编辑工具把执行文件中要求输入密码的代码直接改掉，改成直接跳转到密码比较正确之后的程序入口之处（当然需要事先知道转移指令的机器码）。

静态反汇编方法对于未加入反跟踪代码的待破解程序比较有效，在逆向工程中常用。若静态反汇编得到的源程序正确无误，则可以直接修改源程序，然后再汇编成执行程序，就可以得到我们希望的新程序了。

**3、动态调试跟踪**

单步执行待破解的程序，是可以很直观地获得程序的执行流程和完成的操作的。若碰到反跟踪程序段，则需要配合设置断点的方法，绕过反跟踪程序段设置的陷阱。

**4、自动跟踪**

为了避免人工跟踪时易被反跟踪以及时间较长的问题，可以通过自动跟踪工具进行跟踪。自动跟踪工具可以是类似TD的一个程序，只是不再有显示界面，而是自动把每次单步执行时获取的指令语句等信息按照反汇编的源程序的形式保存到文件中，直至待破解程序退出。然后，人工阅读所保存文件里的汇编语言指令语句序列，分析出程序的流程及算法，再推出密码。这种方法由于仍然使用了单步中断，所以，不适合于采用了堆栈检查或中断矢量表检查的反跟踪措施的程序的破解。

自动跟踪工具也可以采用类似虚拟机的方法实现。待破解程序被自动跟踪工具调入内存后被自动跟踪工具解释执行，在解释的过程中记录待破解程序的指令语句序列并保存到文件中，供后续分析之用。这种实现方法可以避免计时、堆栈检查、中断矢量表检查等绝大多数的反跟踪措施的影响。

**在实际破解过程中，往往会将多种方法交叉使用，以利于更快地得到结果。**

1. **反跟踪方法简介**

**1、采用间接转移抵制静态反汇编**

无论是JMP指令还是CALL指令，都支持间接转移/调用，尤其是可以使用寄存器寻址方式实现间接转移/调用。由于寄存器的内容只有在程序执行之后才能确定具体的值，因此，这种方法可以阻止静态反汇编程序获取程序模块之间的调用关系。当然，这种方法也会让程序的可读性降低，因此，它也加大了人工动态调试跟踪时理解程序的难度。

与间接转移指令配套的措施是建立地址表。地址表可以保存任何标号或子程序的地址，当程序要转移到某个标号或子程序处执行时，可以先将地址表中保存的地址值送到某个寄存器中，再用JMP或CALL去转移(例如，在16位段中，段内转移用字寄存器BX保存地址值，用JMP BX 或 CALL BX进行转移)。为了扰乱视线，可以在JMP或CALL指令之后定义一些初始值随意的变量存储区，由于在程序正常控制下是不会执行到这段存储区的，所以，反汇编程序把这段存储区的数据反汇编成指令语句是没有任何意义的，这就达到了扰乱视线的目的。

地址表可以是静态的，也就是在定义地址表的时候就已经初始化好了对应的地址值。但地址表也可以是动态的，也就是只有在程序运行之后，通过程序赋值或修改之后才形成正确的地址值（对应地址值等数据需要程序动态赋值或修改后才能使用的做法，代码区也是可以动态生成的，也即通过执行一段程序后才能把后续要执行的程序代码恢复出来。这就可以有效地抵制静态反汇编工具了）。

**2、通过计时的方法来抵制动态调试跟踪**

人工单步调试时的指令执行间隔很容易达到秒级，而连续执行的程序中的指令执行间隔一般不会达到毫秒级（即使中间有系统中断打断），因此，通过计算指定的几条指令执行所花费的时间，就可以发现该程序是否被调试。

比较简单的计时方法是在执行指定指令前获取一下当前时间，然后执行这几条指令，之后再次获取当前时间，通过计算这两个时间的差值，即可判断出是否被跟踪。常用的计时程序的精度是55ms，若考虑一下干扰因素，则时间的差值为0或55ms就属于未被跟踪的正常执行状态。

**3、通过检查堆栈来抵制动态调试跟踪**

动态调试过程中会产生单步中断，该中断响应过程会使用被调试程序的堆栈，也就是说被调试程序栈顶以上几个字存储区的内容会被中断响应过程修改。因此，只要预先在程序中执行压栈和出栈操作，即可设定栈顶以上几个字的内容，再获取栈顶以上几个字的内容，判断它们是否是先前压栈时的内容，即可判断是否被调试（当然，为了避免硬件中断的干扰，这段程序段执行前需要关闭可屏蔽中断，也就是要执行CLI指令，等这段程序执行完之后再开中断，也即执行STI指令）。

**4、通过检查中断矢量表来抵制动态调试跟踪**

动态调试工具都会接管单步中断和断点中断的中断服务程序。如果能判断中断矢量表中1号和3号中断处理程序的入口地址被修改到了缺省地址以外的区域，即可判断出有调试工具在运行，则本程序退出。

如果无法判断是否有调试工具在运行，也可以在本程序中直接修改中断矢量表中1号和3号中断服务程序的入口地址，让它们都指向本程序中的一段代码（比如只是一条IRET指令）。有的调试工具会阻止你的程序修改中断矢量表，这时，你只需要在修改之后再去判断一下中断矢量表中对应位置是否改成了你的程序的入口地址，即可判断是否存在调试工具了。

**为了避免反跟踪程序段被跟踪者跳过，一般会通过增加大量的无关程序段来扰乱视线，也可以把有效工作程序段穿插到反跟踪程序中（比如，把密码串转换的程序段放到计时子程序中，这样，如果没有执行计时子程序的功能，也就少了密码串转换的功能）。**

1. **加密方法简介**

**1、采用不公开的算法加密**

当用户不用知道密码，但系统内的信息要做加密处理时，可以选用一种算法或函数表达式对数据进行编码，使得保存或传输的数据是编码后的密文数据。简单的编码方法包括对数据进行移位、求补、高低位交换等。

**2、算法公开，密钥不公开**

算法复杂度足够高，必须知道密码才能解密。例如，对称加密体系DES。

**3、算法和密钥根据需要隐藏与公开**

为了提高算法复杂性不高的加解密方法的安全性，可以不公开加解密算法和密钥。但对于高复杂性的加解密方法，例如公钥加密体系RSA，是可以公布算法及公钥，而只保留私钥的。

**本次实验不需要大家实现算法复杂度高的加密算法，只需要选用单纯的运算方法或简单的函数算法即可。**

1. **程序举例**

下面的程序简单地示意了加密和反跟踪方法的应用情况（其中小写的指令语句属于反跟踪的程序段，颜色相同的部分属于同一种反跟踪方法）。学生在编写本实验的程序时可以只用其中一部分方法，而把主要精力放在灵活地应用这些方法上（比如把程序的功能与反跟踪程序更好地糅合在一起）。

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

;

DATA SEGMENT USE16

D1 DB 0DH,0AH,'STUDENT NAME:$'

D2 DB 0DH,0AH,'CHN SCORE:$'

D3 DB 0DH,0AH,'MATH SCORE:$'

D4 DB 0DH,0AH,'ENG SCORE:$'

RADX DB 10

;以下对成绩表信息进行了简单的加密

BUF DB 'z' XOR 'B','h'XOR 'a','a'XOR 't', 7 DUP(0) ;学生名字为“zha”，采用与密码串

;依次异或的方式加密

DB 100 XOR 'B' ,85 XOR 'a', 80 XOR 't', ? ;学生的成绩也依次与密码串异或。

DB 'l' XOR 'B','i'XOR 'a','s'XOR 't','i' XOR 'B',6 DUP(0)

DB 80 XOR 'B', 98 XOR 'a', 70 XOR 't',?

;以下对密码进行了简单的加密

PWD DB 3 XOR 'C' ;密码串的长度为3，采用与常数43H异或的方式编码成密文

DB ('B' -29H)\*3 ;真实密码为Bat。采用函数(X-29H)\*3对保存的密码进行编码。

DB ('a' -29H)\*3

DB ('t' -29H)\*3

DB 0A1H,5FH,0D3H ;用随机数填充密码区到6个字符，防止破解者猜到密码长度

;

IN\_PWD DB 7 ;使用者输入的密码区，最大长度6个字符

DB ?

DB 7 DUP(0)

STR1 DB 0DH,0AH,'PLEASE ENTER PASSWORD:$'

;

P1 DW PASS1 ;地址表（用于间接转移反跟踪）

E1 DW OVER

P2 DW PASS2

OLDINT1 DW 0,0 ;1号中断的原中断矢量（用于中断矢量表反跟踪）

OLDINT3 DW 0,0 ;3号中断的原中断矢量

DATA ENDS

;

CODE SEGMENT USE16

ASSUME CS:CODE,DS:DATA,SS:STACK

START: MOV AX,DATA

MOV DS,AX

xor ax,ax ;接管调试用中断，中断矢量表反跟踪

mov es,ax

mov ax,es:[1\*4] ;保存原1号和3号中断矢量

mov OLDINT1,ax

mov ax,es:[1\*4+2]

mov OLDINT1+2,ax

mov ax,es:[3\*4]

mov OLDINT3,ax

mov ax,es:[3\*4+2]

mov OLDINT3+2,ax

cli ;设置新的中断矢量

mov ax,OFFSET NEWINT

mov es:[1\*4],ax

mov es:[1\*4+2],cs

mov es:[3\*4],ax

mov es:[3\*4+2],cs

sti

LEA DX,STR1

MOV AH,9

INT 21H

LEA DX,IN\_PWD ;输入密码字符串

MOV AH,10

INT 21H

cli ;计时反跟踪开始

mov ah,2ch

int 21h

push dx ;保存获取的秒和百分秒

MOV CL,IN\_PWD+1 ;比较输入的串长与密码长度是否一样

XOR CL,'C'

SUB CL,PWD

MOVSX BX,CL

ADD BX,OFFSET P1

mov ah,2ch ;获取第二次秒与百分秒

int 21h

sti

cmp dx,[esp] ;计时是否相同

pop dx

jz OK1 ;如果计时相同，通过本次计时反跟踪

mov bx,offset E1 ;如果计时不同，则把转移地址偏离P1

OK1: mov bx,[bx]

cmp word ptr cs:[bx],0B60FH ;是否是PASS1处的指令，其实是用于判断前面比较的

;串长是否相同

jz OK2

jmp E1

OK2: jmp bx

db 'How to go' ;定义的冗余信息，扰乱视线

PASS1: MOVZX CX,IN\_PWD+1

cli ;堆栈检查反跟踪

push P2 ;PASS2的地址压栈

MOV SI,0

MOV DL,3

pop ax

mov bx,[esp-2] ;把栈顶上面的字（PASS2的地址）取到

sti

jmp bx ;如果被跟踪，将不会转移到PASS2

db 'i do not know！'

PASS2: MOVZX AX,IN\_PWD+2[SI] ;比较密码是否相同。把输入的串变成密文，与保存的密文比较

SUB AX,29H

MUL DL

CMP AL,PWD+1[SI]

JNZ ERR2

INC SI

LOOP PASS2

JMP PASS3

ERR2: mov ebx,OFFSET P1 ;当密码不等时，通过地址表计算出OVER（退出）的位置

mov edx,1

jmp word ptr [ebx+edx\*2] ;指向OVER

db 'YES,get it'

;

PASS3: mov bx,es:[1\*4] ;检查中断矢量表是否被调试工具阻止修改或恢复

inc bx

jmp bx ;正常修改了的化，这里将转移到TESTINT，否则就不知道转到哪了

db 'Now,you see.'

;

PASS4: LEA DX,D1 ;进入实际功能区。这里仅实现显示BUF区第2个学生的姓名和成绩

MOV AH,9

INT 21H

MOV SI,0

NEXT: MOV DL,BUF+14[SI]

CMP DL,0

JE SCORE

MOV AX,SI

DIV IN\_PWD+1 ;求密码长度的模数（因为名字长度可能超过密码长度，

;取模之后可以保证循环使用密码串）

MOVZX DI,AH

XOR DL,IN\_PWD+2[DI] ;注意，使用用户输入的明文密码串进行解密，不仅不用去解密

;程序中的密码，而且可以抵抗跟踪者跳过密码判断过程直接

;转移到实际功能区后获取密文信息的做法

MOV AH,2

INT 21H

INC SI

JMP NEXT

SCORE: LEA DX,D2 ;显示三科成绩

MOV AH,9

INT 21H

MOV SI,0

MOV AL,BUF+14+10[SI]

XOR AL,IN\_PWD+2[SI]

MOV AH,0

DIV RADX ;这里假设成绩最大为99，所以只除一次10

ADD AX,3030H

PUSH AX

MOV DL,AL

MOV AH,2

INT 21H

POP AX

MOV DL,AH

MOV AH,2

INT 21H

INC SI

;

LEA DX,D3

MOV AH,9

INT 21H

MOV AL,BUF+14+10[SI]

XOR AL,IN\_PWD+2[SI]

MOV AH,0

DIV RADX

ADD AX,3030H

PUSH AX

MOV DL,AL

MOV AH,2

INT 21H

POP AX

MOV DL,AH

MOV AH,2

INT 21H

INC SI

;

LEA DX,D4

MOV AH,9

INT 21H

MOV AL,BUF+14+10[SI]

XOR AL,IN\_PWD+2[SI]

MOV AH,0

DIV RADX

ADD AX,3030H

PUSH AX

MOV DL,AL

MOV AH,2

INT 21H

POP AX

MOV DL,AH

MOV AH,2

INT 21H

;.........注意，本实验中是要实现查找学生姓名的，在查找学生姓名时，应将学生姓名串

; .........转换成密文后再去成绩表中搜索，也就是用密文比较。

;..........计算平均成绩时需要解密各科成绩，但不要把成绩表中的密文数据给覆盖掉了。

JMP OVER

;

NEWINT: iret

TESTINT: jmp PASS4

;

OVER:

cli ;还原中断矢量

mov ax,OLDINT1

mov es:[1\*4],ax

mov ax,OLDINT1+2

mov es:[1\*4+2],ax

mov ax,OLDINT3

mov es:[3\*4],ax

mov ax,OLDINT3+2

mov es:[3\*4+2],ax

sti

MOV AH,4CH

INT 21H

CODE ENDS

END START