SI485H Stack Based Binary Exploits and Defenses
FALL 2015

NAME:_____

**HW5**

COLLABORATOR(S):_____

5/3/1/0
1. What's the equivalet of **main** in assembly programming?

5/3/1/0
2. If you have written an assembly program called **myprog.asm**, write the series of compilation/linking commands to produce the executable **myprog**.

3. Consider the following assembly program:

```
SECTION .text
      global _start
_start:

      mov     eax,0x0a797661
      push    eax
      mov     eax,0x4e206f47
      push    eax             ;MARK 1

      mov     edx,0x8
      mov     ecx,esp
      mov     ebx,0x1         ;MARK 2
      mov     eax,0x4
      int     0x80            ;MARK 3

      mov     ebx,0           ;MARK 4
      mov     eax,1
      int     0x80
```

5/3/1/0
a) What is the output of this program? **Explain.**

5/3/1/0
b) How does the output of the program change if at **MARK 2** 0x1 were changed to 0x2?

5/3/1/0
c) What system call is being setup to execute at **MARK 4**?

___/25

NAME: _____

4. Consider the following assemly program:

```
SECTION .data
prompt:  db "(echo) ",0x0a ;MARK 1

SECTION .text
     global _start

_start:
     mov     edx,0x7
     mov     ecx,prompt
     mov     ebx,0x1
     mov     eax,0x4
     int     0x80          ;MARK 2

     sub     esp,0x10      ;MARK 3
     mov     edx,0x10
     mov     ecx,esp       ;MARK 4
     mov     ebx,0x0
     mov     eax,0x3
     int     0x80

     mov     edx,eax       ;MARK 5
     mov     ecx,esp
     mov     ebx,0x1
     mov     eax,0x4
     int     0x80          ;MARK 6


     mov     ebx,eax         ;MARK 7
     mov     eax,1
     int     0x80
```

a) Explain the command **db** as it is used at **MARK 1.** 5/3/1/0

b) What is the result of the interupt at **MARK 2**? 5/3/1/0

c) At **MARK 3** and **MARK 4** esp is both manipulated and used as a setting to a system call. Explain this setup. 5/3/1/0

d) At **MARK 5, eax** is used for the setting to the system call interupt at **MARK 6.** Expain how this relates to the previous system call interupt and the output of the program. 5/3/1/0

e) Assume that the system call at **MARK 6** might fail, explain how the code at **MARK 7** would allow the programmer to determine the kind of failure that occurred. 5/3/1/0



___/25

NAME: _____

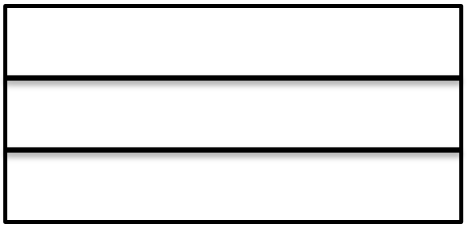
5. Consider the following compiled and assembeled shell code:

```
08048080 <_start>:
 8048080:    6a 00                   push   0x0
 8048082:    68 a8 90 04 08          push   0x80490a8
 8048087:    ba 00 00 00 00          mov    edx,0x0
 804808c:    89 e1                   mov    ecx,esp
 804808e:    bb a8 90 04 08          mov    ebx,0x80490a8
 8048093:    b8 0b 00 00 00          mov    eax,0xb
 8048098:    cd 80                   int    0x80

 804809a:    bb 00 00 00 00          mov    ebx,0x0
 804809f:    b8 01 00 00 00          mov    eax,0x1
 80480a4:    cd 80                   int    0x80
```

a) Complete the stack diagram prior to the first interupt. Assume that the string **/bin/sh** is at address 5/3/1/0

esp ->



b) If we were to package up the bytes of this shell code and call it, like so

```
int main(){

  char * code = "\x6a\x00\x68\xa8  (...)"
  ((void(*)(void)) code)();

}
```

Why would this fail?

5/3/1/0

6. Consider the following compiled and assmbeled shell code:

```
08048060 <_start>:
 8048060:    eb 20                   jmp    8048082 <callback>

08048062 <dowork>:
 8048062:    5e                      pop    esi
 8048063:    6a 00                   push   0x0
 8048065:    56                      push   esi
 8048066:    ba 00 00 00 00          mov    edx,0x0
 804806b:    89 e1                   mov    ecx,esp
 804806d:    89 f3                   mov    ebx,esi
 804806f:    b8 0b 00 00 00          mov    eax,0xb
 8048074:    cd 80                   int    0x80
 8048076:    bb 00 00 00 00          mov    ebx,0x0
 804807b:    b8 01 00 00 00          mov    eax,0x1
 8048080:    cd 80                   int    0x80

08048082 <callback>:
 8048082:    e8 db ff ff ff          call   8048062 <dowork>
 8048087:    2f                      das
 8048088:    62 69 6e                bound  ebp,QWORD PTR [ecx+0x6e]
 804808b:    2f                      das
 804808c:    73 68                   jae    80480f6 <callback+0x74>
```

a) At the **pop esi** instruction, what value will be in **esi**?

5/3/1/0

b) Explain how this structure avoids fixed references.

5/3/1/0

c) This shell code is still not complete. What's wrong?

5/3/1/0



___/25

NAME: _____

7. For each of the null containing instructions replace them with equivalent non-null containing instructions. (It may take more than one instruction.

5/3/1/0

a) push 0x0

b) mov ebx, 0x0

c) mov eax, 0xb

8. Below is the **asm** of the shell code presented in question 6. Rewrite this shell code so that it does not have any NULL bytes.

10/8/5/2/0

```
SECTION .text
            global _start

_start:
            jmp  callback

dowork:
            pop esi
            push    0
            push    esi

            mov     edx,0
            mov     ecx,esp
            mov     ebx,esi
            mov     eax,0xb
            int     0x80

            mov     ebx,0
            mov     eax,1
            int     0x80

callback:
            call dowork
            db "/bin/sh",0
```

9. How many bytes is in the resulting shell code once you remove the NULL bytes. (*Hint: You will need to compile and extract the bytes using objdump to* count)

5/3/1/0

___/25