CSE 406
Online -1 (B - 2)

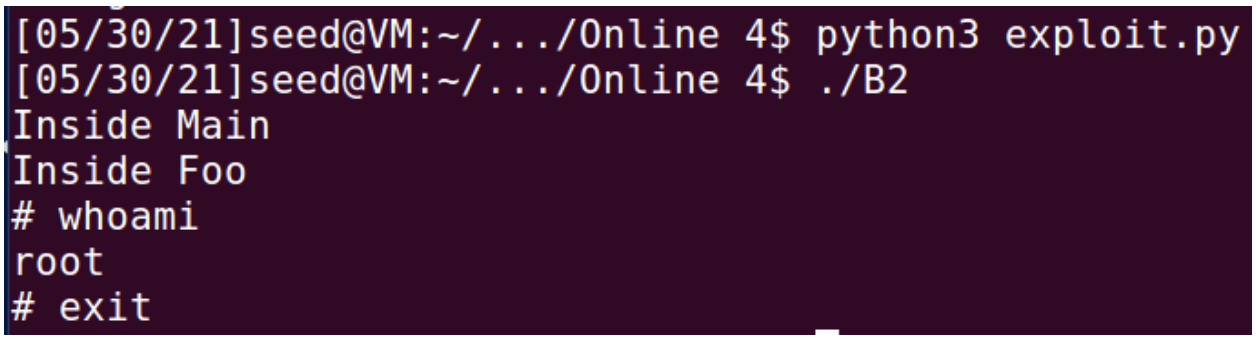
You are given the following vulnerable C program *B2.c.* Replace <param_1> and <param_2> in the source code with the corresponding values of Table-1.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char* foo();
int bar(int x);
int execute(char* m_code);
char code[] =
  "\x31\xc0"              /* xorl    %eax,%eax              */
  "\x50"                  /* pushl   %eax                   */
  "\x68""//sh"            /* pushl   $0x68732f2f            */
  "\x68""/bin"            /* pushl   $0x6e69622f            */
  "\x89\xe3"              /* movl    %esp,%ebx              */
  "\x50"                  /* pushl   %eax                   */
  "\x53"                  /* pushl   %ebx                   */
  "\x89\xe1"              /* movl    %esp,%ecx              */
  "\x99"                  /* cdq                            */
  "\xb0\x0b"              /* movb    $0x0b,%al              */
  "\xcd\x80"              /* int     $0x80                  */
;

int bof(char *str){
    char buffer[<param_1>];
    strcpy(buffer, str);
    return 1;
}

int execute(char* m_code){
   ((void(*)( ))m_code)( );
}

char* foo(){
    printf("Inside Foo\n");
    return code;
}

int bar(int x){
    printf("Input Parameter %d\n",x);
    return x+1;
}
```

```
int main(int argc, char **argv){
    char str[<param_2> + 1];
    FILE *badfile;
    badfile = fopen("badfile", "r");
    printf("Inside Main\n");
    fread(str, sizeof(char), <param_2>, badfile);
    bof(str);
    printf("Returned Properly\n");
    return 1;
}
```

Tasks:

1. First, compile the program with root privilege as shown in the lab. Do not forget to turn off address space randomization and stack protection. Also, make sure that the stack is executable while compiling the program.
2. Prepare a payload (e.g. badfile) which will cause the program to produce the following output



3. First you have to call the function `foo`, then using its return value call function `execute`.
4. Produce similar output when <param_2> = <param_1>+100. *[Hint: Directly point to the address of the **variable code** instead of calling the **function execute**]*
5. Rename your ***exploit.py*** file with ***16050xx.py*** and submit in moodle.

Mark Distribution

Item	Marks
Call function `foo` (i.e. Print Inside Foo)	5
Call function `foo` and.`execute`	12
Task 4	3
Total	**20**

Hint

You already have the payload to open a shell. Steps to cause the overflow is similar to opening a shell. You just need to replace the payload of opening a shell with your own payload which calls **`foo`** and **`execute.`**

For example, the following payload calls the function **`bar`** twice.

```
shellcode_3 = (
"\x6a\x04"              #push   4             (Setting Parameter x1 = 4)
"\xbb\x3e\x85\x04\x08" #mov    ebx,0x804853e (Move entry point of bar to edx)
"\xff\xd3"              #call   ebx           (Call the function bar)

"\x50"                  #push   eax           (Return value is stored at eax.
                        #                      Setting Paratemer x1 = last return value)
"\xff\xd3"              #call   ebx           (Calling bar again)
).encode('latin-1')
```

Output of the above payload:

```
Inside Main
Input Parameter 4
Input Parameter 5
Segmentation fault
```

Table - 1

Student ID	Param_1	Param_2
1605091	324	2030
1605092	810	1807
1605093	783	2012
1605094	829	2220
1605095	802	1642
1605096	741	1940
1605097	403	1961
1605098	770	2253
1605099	824	1708
1605100	320	2174
1605101	452	2252
1605102	917	2025
1605103	686	2331
1605104	835	1926

1605105	496	1612
1605106	758	1674
1605107	390	1881
1605108	988	2252
1605109	530	2289
1605110	908	1982
1605111	649	2389
1605112	515	1617
1605113	649	1731
1605114	933	1622
1605115	391	2333
1605116	995	2079
1605117	930	2032
1605118	812	1648
1605119	503	2168
1605120	820	1809
0905081	457	2110
1405059	619	1710
1405081	545	1684
1605040	393	2156
1605021	916	1876
1605065	597	2320
1605067	441	2245
1605076	691	1986
1605080	721	2129
1605085	675	2239
1605089	785	2392
1605090	461	2098