# IERG 4130 - Introduction to Cyber Security (2022 Fall)
# Assignment 1: Software Security

**Total: 50’ (+bonus 5’)**
**Due Date: Oct. 6, 11:59 pm**
Note:

- Please answer the following questions. Please write down your student id and answers in a single pdf file, then submit it to Blackboard.
- We assume that all the C code programs or snippets (with other omitted parts) in the following questions are compiled into an ELF file and run on a 32-bit x86 (i.e. IA-32) machine (32-bit aligned).

1. In which memory segments are the variables (including arguments *str* and *b*) in the following code located? (5’)

```
int a = 0;
void func(char *str, int b)
{
    int c;
    long d;
    char buf[64];
    char *ptr = malloc(sizeof(int));
}
```

2. For the following C program, assume the program is compiled, loaded, and run until line 6 (i.e. a breakpoint was set line 6). Please draw the function stack frame for the following C function. (10’)
Note that:

- As aforementioned, the program is 32-bit aligned. A char is 1 byte, and an int is 4 bytes.
- To simplify your thinking, putting down the symbol name of each element is fine.
- The frame pointer and saved eip of the caller of bof() are old_ebp and ret_addr respectively.
- No other general-purpose registers that need to be saved except ebp.
- Besides the variables (including arguments) and return address, please also indicate the values of ebp and esp at this point.

```
1 int bof(char *str, int i, int j, int k)
2 {
3     char buffer[16];
4     int temp = i + j + k;
5     strcpy(buffer, str);
6     return temp;
7 }
```

3.Is this function safe?  Justify your answer and propose possible mitigation by modifying the source code with the necessary explanation in your answer. (10’)
Hint: void *memcpy (void *destination, const void *source, size_t num);

```
1 void func(char *data, int len)
2 {
3     char buf[64];
```

```
    if (len > 64)
        return;
    memcpy(buf, data, len);
}
```

4. The following function is called in a privileged program. The argument str points to a string that is entirely provided by users (the size of the string is up to 300 bytes). When this function is invoked, the address of the buffer array is 0x66AA0020, while the return address is stored in 0x66AA0050. Please write down the string that you would feed into the program, so when this string is copied to *buf* and when the bof() function returns, the privileged program will run your code. In your answer, you don't need to write down the injected code, but the offsets of the key elements in your string need to be correct. (10')

```
int bof(char *str)
{
    char buf[16];
    strcpy(buf,str);
    return 1;
}
```

5. Please answer the following short questions about software security.

1) What are the common grounds between stack overflow and heap overflow? (3')
2) Which kinds of variables can be overwritten by stack overflow attack? (3')
3) Pls discuss the limitations of the following control-flow hijack defense techniques respectively.
   Hint: How to defeat each kind of mitigation technique?
   a. StackGuard(Canary) (3')
   b. No-Execution-Bit (3')
   c. Address Space Layout Randomization (ASLR) (3')

6. A student proposes to change how the stack grows. Instead of growing from high address to low address, the student proposes to let the stack grow from low address to high address. This way, the buffer will be allocated above the return address, so overflowing the buffer will not be able to affect the return address. Please comment on this proposal by combining the following program. (bonus 5')
(Hint: what will happen if a too large string is passed to the bar() function.)

```
void bar(char *str)
{
    char c[7];
    strcpy(c, str);
}
void foo()
{
    bar("overlflow");
}
```