Assignment 10 – Stack smashing and feedback

Question 1: Stack smashing (8 points)

Read the following code and answer questions below.

```c
#include "stdio.h"
#include "string.h"

void buggy_print(char *msg, int len) {
    char buf[16];
    memcpy(buf, msg, len);
    printf("msg is %s!\n", buf);
}

void wrap(char *msg, int len) {
    buggy_print(msg, len);
}

void attack() {
    printf("you've been attacked!\n");
}

int main() {
    buggy_print("hello", sizeof("hello"));
    int num = /* TODO: fill in an integer */;
    void *ptrs[num];
    for (int i=0; i<num; i++) {
        ptrs[i] = attack;
    }

    wrap((char*) ptrs, sizeof(void *)*num);
}
```

1.a (3 points)
Save the above code to a file, “demo.c”.
Fill in “TODO” with an integer 1.
Compile and run the code:

```
$ gcc -o demo demo.c -fno-stack-protector -z execstack
$ ./demo
```

What do you see as the result and why?
Write down the output of the program and explain why this happens in 2-3 sentences.

1.b (3 points)

Now, fill in TODO with an integer 10.
Compile and run the code:

```
$ gcc -o demo demo.c -fno-stack-protector -z execstack
$ ./demo
```

Do you see the string "you've been attacked"? How does this happen?
Write down the output of the program and explain **from a high level** why you see the **being-attacked string**.

notes:
1. if you're using x86 CPUs (like Intel and AMD), you should be able to fully understand what has happened.
- recall x86 instruction "call" and "ret"; what do they do?
- in particular, where the "$rip" (instruction pointer) will point to after "ret".
2. if you're using ARM CPUs (like Apple silicon), the results would be slightly different from what you learned because ARM uses a different way for function call, but the main idea stays the same.

Question 2: Feedback (4 points)

This is to gather feedback. Any answer, except a blank one, will get points.

2.a (2 point)
Please state the topic or topics in this class that have been least clear to you.

2.b (2 point)
Please state the topic or topics in this class that have been most clear to you.

2.c (optional) anything you'd like us to know?