Beirut Arab University	**Course: CMPS455-Computer Security**
Faculty of Science	**Semester: Spring 2024-2025**
Mathematics and Computer Science Department	**Instructor:**
Lab Sheet # 11	**TA:**

Part 1 – General questions

1. What is the difference between malicious code and non-malicious error?
2. What non malicious program error can be used by an attacker to get rid of normal authentication? How do they work and how can they be exploited?
3. What security issue is indicated in the URL:
   http://www.securitylab.com/query.php?date=2025April9
   Which systems are vulnerable to this type of error?

Part 2 – A non-malicious program error

```
#include<stdio.h>
#include<sys/stat.h>
int main()
{
        char tmpname [20] = "/tmp/not_so_random";
        char command [100];

        struct stat buf;
        int ok = stat(tmpname, &buf);

        sleep(5);

        if(ok != 0) { // 3 Only continues if it doesn't
                printf("Temporary file already exists, we need a new name..
        } else {
                printf("File does not exist, writing...");
                sprintf(command, "echo Hello > %s", tmpname);
                system(command);
        }
}
```

1. What is this program supposed to do in Linux?
2. What type of vulnerability is exposed here?
3. When does this bug occur?
4. How can we avoid this vulnerability?

Part 3 – Buffer overflows

1. Define an array of size 10, and initialize it with any value.
   Print the 10 elements of the array.
   Try to print the 11^{th} element of the array.

Try to print the element of index -1 in both cases.
Compare the behavior of the program in C++ and Java. Conclude.

2. Given the following C code:

```c
#include <stdio.h>
#include <string.h>

#define BUFFERSIZE 10

char buffer[BUFFERSIZE] ="message";
char password [BUFFERSIZE] ="passw0rd";

int main(int argc, char** argv)
{
    if (argc != 3) {
        printf("Usage: %s <password> <string_to_print>\n", argv[0]);
        return 1;
    }
    strcpy(buffer,argv[2]);
    if (strcmp (argv[1], password)==0) {
        printf("password checks out\n");
        printf("message: %s\n", buffer);
    } else {
        printf("password error\n");

    }
    return 0;
}
```

What is this program supposed to do?
Try the code above with different arguments to test its functionality.
What vulnerability can the attacker exploit? What potential consequences could arise from it?
How would you modify the code to prevent this vulnerability while maintaining the functionality of the authentication mechanism? Give two methods.
Do you know other vulnerable functions? Mention two and give their safer version. Compare their operation in a program.