Learning Outcome: Students will have fun with stack buffer overflow.

For the first two problems, write the following program in a file named `warmup.c`.

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

int main(){
        char str1[] = "Hello";
        char str2[] = "World!";
        //printf("Enter a string: ");
        //scanf("%s", str2);
        printf("str1 = %s at %p\n", str1, str1);
        printf("str2 = %s at %p\n", str2, str2);
}
```

a. Run the program on Kali Linux. Make sure that you understand where `str1` and `str2` are located in memory, i.e. the `main` function's stack frame.
b. Now, uncomment the `printf` and `scanf` calls and run the program again. It will prompt the user to enter a string that will be stored in `str2`. Run it several times using strings of different lengths 4, 5, 6, 7, 8, and 9. Observe output of the program for each run.

1. (25 points) Run `warmup.c` and enter a string so that it prints `str1` is "`Hi`". It doesn't matter what `str2` prints.

```
str1 = Hi at 0x7ff…
str2 = … at 0x7ff…
```

2. (25 points) Run `wamup.c` again using strings of different lengths. As you increase the length, it will eventually crash the program. This time, your task is to find a longest string that does not cause segmentation fault. What's the length? Include the string you found as a comment in the source code. You will use the string when you receive signoffs.

3. (25 points) Open the `linux_overflow.c` file. Take a moment to read through the code. Notice that the C library function named `system` is used in the `display_time` function. The `system` function executes a command and returns after the command has been completed, e.g., `system("date"), system("ls -al"), etc;`

   Execute the program and enter your name. Your name and current time will be displayed.

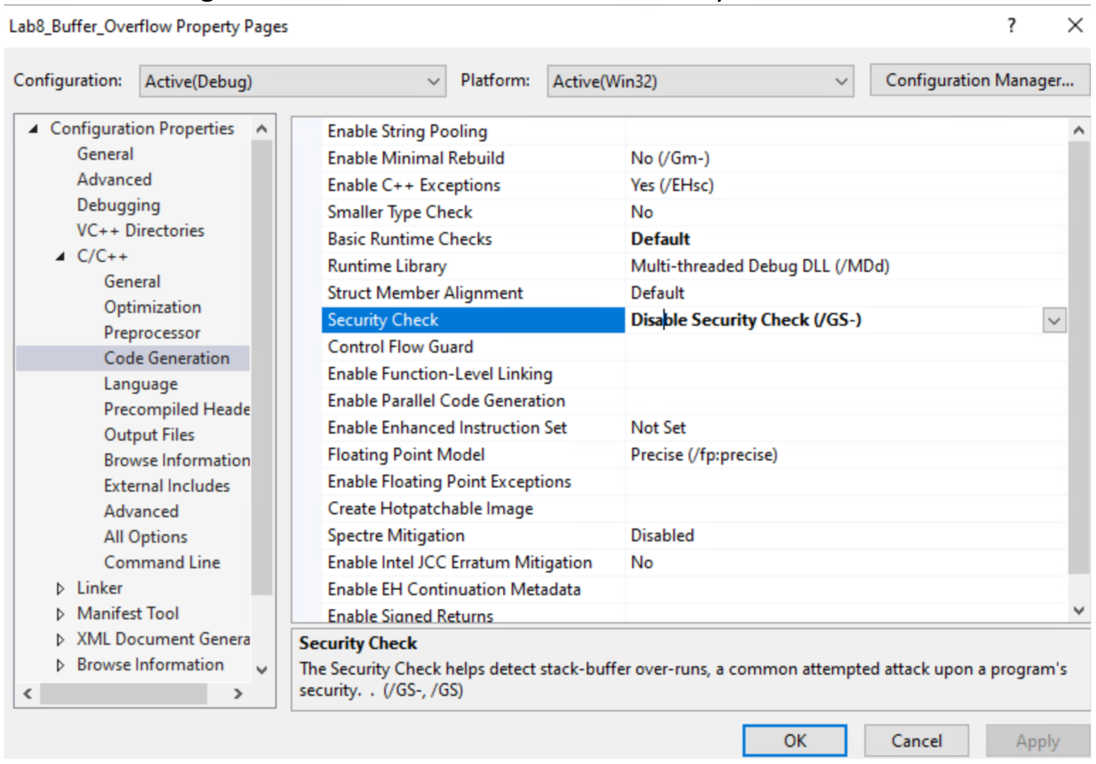
The program should run fine in normal circumstances. However, it has vulnerability that can be exploited by a bad guy. Did you notice that? No? Let's use the vulnerability to make the program execute "`ls -al`" instead of "`date`". To do this, you are not modifying the provided program. Instead, you will feed a "bad" string into the program to manipulate its behavior.

Note:

- Even though you are not allowed to delete or change given code, feel free to insert `printf` statements for debugging.
- The blank space in your string between ls and -al will break the `scanf` function. You may use the metacharacter `$IFS`.
- Don't forget to include the string you found in your source code as a comment.

4. (25 points) Copy and paste `linux_overflow.c` and rename it to `windows_overflow.c`. Use Visual Studio to run `windows_overflow.c.` Try to run the program. What happens?

   VS does not allow you to use unsafe functions such as `scanf` or `strcpy` by default. If you change them to `scanf_s` and `strcpy_s` as it suggests, your exploitation won't work. So, turn off its security features. To do this, click Project → Properties → C/C++ → Code Generation. Change "Basic Runtime Checks" and "Security Checks" as follows:



Lab8_Buffer_Overflow Property Pages

Configuration: Active(Debug)   Platform: Active(Win32)   Configuration Manager...

Enable String Pooling	
Enable Minimal Rebuild	No (/Gm-)
Enable C++ Exceptions	Yes (/EHsc)
Smaller Type Check	No
Basic Runtime Checks	**Default**
Runtime Library	Multi-threaded Debug DLL (/MDd)
Struct Member Alignment	Default
Security Check	**Disable Security Check (/GS-)**
Control Flow Guard	
Enable Function-Level Linking	
Enable Parallel Code Generation	
Enable Enhanced Instruction Set	Not Set
Floating Point Model	Precise (/fp:precise)
Enable Floating Point Exceptions	
Create Hotpatchable Image	
Spectre Mitigation	Disabled
Enable Intel JCC Erratum Mitigation	No
Enable EH Continuation Metadata	
Enable Signed Returns	

Security Check
The Security Check helps detect stack-buffer over-runs, a common attempted attack upon a program's security. . (/GS-, /GS)

Now run the program and enter a bad string so that "dir" is executed instead of "date".

Deliverables:

- Push source code (**warmup.c**, **linux_overflow.c,** and **windows_overflow.c**) to your Assignment08 repository before 7 pm, Wednesday 10/20/2021.
- Each of the source files must contain the string you found (Q2 – Q4). During signoffs, you will copy and paste the string into the console when the program runs.