**Practical 16 Integer Overflow**

**Exercise Inspecting sizes of data types**

In Kali

1. In Kali, create the following C program “datatype\_size.c”.

#include <stdio.h>

int main() {

printf("The size of int data type is \t\t%d bytes\n", sizeof(int));

printf("The size of unsigned int data type is \t\t%d bytes\n", sizeof(unsigned int));

printf("The size of short int data type is \t\t%d bytes\n", sizeof(short int));

printf("The size of long int data type is \t\t%d bytes\n", sizeof(long int));

printf("The size of float data type is \t\t%d bytes\n", sizeof(float));

printf("The size of double data type is \t\t%d bytes\n", sizeof(double));

printf("The size of char data type is \t\t%d bytes\n", sizeof(char));

}

1. Compile and run the program. Take note of the various sizes of the different data types.

gcc –o datatype\_size datatype\_size.c

./datatype\_size

1. Create the same program in Windows under Cygwin, compile and run it. Is there any difference in the sizes of the different data types?

When writing programs, it is important to be aware of the maximum and minimum values that can be stored for numeric data types.

**Exercise Two’s Complement**

In Kali or Windows:

1. Create the following C program “twocomplement.c”.

#include <stdio.h>

int main() {

int num;

num = 0x7FFFFFFF;

printf("Current value of num: %d (dec) %x (hex)\n", num, num);

num++;

printf("Current value of num: %d (dec) %x (hex)\n", num, num);

}

1. Compile and run the program. When the integer reaches the value 0x80000000, it becomes a negative value.
2. Change num into an unsigned int. Make the following changes in your program (highlighted in red and bold)

#include <stdio.h>

int main() {

**unsigned** int num;

num = 0x7FFFFFFF;

printf("Current value of num: %**u** (dec) %x (hex)\n", num, num);

num++;

printf("Current value of num: %**u** (dec) %x (hex)\n", num, num);

}

1. Compile and run the program. This time, when the integer reaches the value 0x80000000, it remains positive.

*End of Practical Setup*