Stack Buffer Overflow:

Graphical user interface, text, application, email

Description automatically generated

From the above code, we can see a stack buffer overflow caused by the statement “memmove(dest, data, wcslen(data)\*sizeof(wchar\_t)) “ at line 24. Specifically, the buffer “dest” has a size of 50 “wchar\_t” elements. However, the statement tries to copy the string carried in the buffer named “data” to “dest”. As initialized by the statements of “wmemset(data, L'A', 100-1)” at line 18 and “data[100-1] = ‘\0’” at line 19, the string in “data” will have a size of 99 “wchar\_t” elements. Thus, the memmove will overflow “dest” and thus, a stack overflow happens.

Heap Buffer Overflow:

Graphical user interface, text, application

Description automatically generated

From the above code, we can see a heap buffer overflow caused by the statement “SNPRINTF(data, 100, L"%s", source)” at line 29. Specifically, “data” is a buffer with 50 “wchar\_t” elements, allocated at line 21. However, the statement at line 29 tries to move 100 “wchar\_t” elements from “source” to “data”. Thus, the buffer “data” will be overflowed and a heap overflow happens.

Use After Free:

A picture containing graphical user interface

Description automatically generated

From the above code, we can see a use-after-free caused by the statement at line 32. Specifically, the code declares a pointer “data” at line 12, and makes “data” point to a buffer allocated on the heap at line 17. The buffer is initialized at line 23 and then freed at line 27. However, the buffer, after being freed, is used again at line 32. Thus, a use-after-free happens.

Integer Overflow:

Graphical user interface, text, application

Description automatically generated

From the code at line 10, we can see that the variable “data” has a type of “int64\_t” (also known as “long long int”), whose value ranges from -2^31 to 2^31-1. The code at line 15 assigns LLONG\_MAX (i.e., 2^31-1 ) to “data” and then adds 1 to “data” at line 26, which goes beyond the maximal value of a “long long int” and overflows the value in “data”. Thus, an integer overflow happens.

Use of Uninitialized Variable:

Graphical user interface, text, application

Description automatically generated

From the above code, we can see the use of uninitialized variables caused by the statements at line 21 and line 22. Specifically, the code first declared a variable “data” at line 12. However, the code never initializes the data in that piece of memory before using the data at line 21 and line 22. Thus, use of uninitialized variables happens.