What are some of the formatting vulnerabilities that can be encountered in using the iostream library in C++?  What tips can be utilized to identify these vulnerabilities?  Be sure to provide an appropriate source code example to illustrate your points.

Formatting vulnerabilities can exist when an input to format functions is executed as a command. Format string function usually require a format specifies, which specifies how the output will appear. When a format specifier requires an argument, it expects to find it on the stack. A vulnerability can occur if attackers can specify a format string to a format function, granting them control to pop from the stack. One way to mitigate this vulnerability is to compare the type of arguments passed to a function and the type of arguments a function expects.

Vulnerabilities are also caused by integer errors. There are 2 types of exploitable integer errors; integer overflow and integer signedness. Integer overflows occurs when an integer is too large for its data type. This could cause a program to fail or act unexpectedly but is generally not exploitable. Integer signedness can occur if a declared signed integer was passed to function expecting an unsigned integer. In a scenario where a negative number is passed as a maximum size argument and later used as an unsigned value, a heap overflow may ensue.

Another issue that I came across by chance is data type. In the following code, x is declared as an integer. The next line prompts the user to enter x. If you run this code yourself, you can see that x accepts strings as an input. Although in this scenario mismatched data types don’t have consequences, they can probably cause program termination or failure in a different scenario.

The iostream library can be used to format output.

The iostream library provides basic input/output operations for C++ programs. Data type inconsistency results in truncation errors and round off error.