Buffer Overflow

What is it?

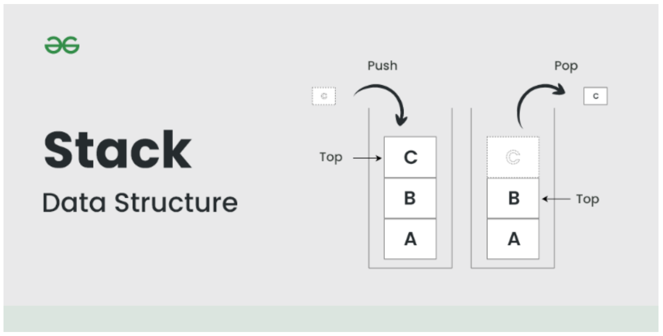
In C, when you initialise a variable, you must explicitly state its size and the memory to allocate it. This is known as a buffer. If you attempt to write to memory outside of the allocated buffer, this is not intended behaviour and is known as buffer overflow.

How can this be exploited?

There can be several consequences of buffer overflows, ranging from overwriting important data, to crashing the program, or allowing an attacker to run their own code.

Whenever a C program is run, it is assigned a block of memory for it to store any local variables. This block of memory operates as a stack. The name “stack” describes how variables are stored in this block of memory; they are stacked next to one another. Therefore, if we were to accidentally (or intentionally…) try to write outside the bounds of a variable, we might end up writing data to another variable stored next to it instead.

Additionally, a common way of describing how variables are stored in a stack is “Last In, First Out”, meaning that the last variable added is the first to be removed.



What we can understand from this principle is that the order in which these variables are stored is determined by the order they are initialised. Hence, we are able to deliberately target a variable to attack, and this can range from innocuously changing a number within a program, to changing the return address of a function so that you can gain unwanted access to other areas of the device or execute your own malicious code. However, in modern systems, there are often preventative measures such as randomising the order that these variables are stored.

Additionally, when you carry out a buffer overflow attack, you might be attempting to write to an address in the kernel space. Generally, we interact with the user space, while the kernel space is reserved for the core of the operating system, thus protecting the computer’s essential functions and hardware. If you attempt to write to an address in the kernel space, most often you will crash the program.