Problem solving points.

1. Integer Overflow:
   * An integer overflow is a type of an arithmetic overflow error when the result of an integer operation does not fit within the allocated memory space.
   * Instead of an error in the program, it usually causes the result to be unexpected.
   * **Must read** - <https://www.welivesecurity.com/2022/02/21/integer-overflow-how-it-occur-can-be-prevented/>
   * An integer overflow or wraparound happens when an attempt is made to store a value that is too large for an integer type. The range of values that can be stored in an integer type is better represented as a circular number line that wraps around. The circular number line for a signed char can be represented as the following:
   * A picture containing shape

     Description automatically generated
   * If an attempt is made to store a number greater than 127 in a signed char, the count wraps around to −128 and continues upwards, toward zero, from there. Thus, instead of what should be 128 the value −128 is stored, instead of 129 the value −127, and so on.
2. Buffer Overflow:
   * A buffer overflow vulnerability occurs when you give a program too much data. The excess data corrupts nearby space in memory and may alter other data. As a result, the program might report an error or behave differently. Such vulnerabilities are also called buffer overrun.
   * Some programming languages are more susceptible to buffer overflow issues, such as C and C++. This is because these are low-level languages that rely on the developer to allocate memory.
   * Most common languages used on the web such as PHP, Java, JavaScript or Python, are much less prone to buffer overflow exploits because they manage memory allocation on behalf of the developer.
   * There are two primary types of buffer overflow vulnerabilities: stack overflow and heap overflow.
     1. In the case of stack buffer overflows, the issue applies to the stack, which is the memory space used by the operating system primarily to store local variables and function return addresses. The data on the stack is stored and retrieved in an organized fashion (last-in-first-out), the stack allocation is managed by the operating system, and access to the stack is fast.
     2. In the case of heap buffer overflows, the issue applies to the heap, which is the memory space used to store dynamic data. The amount of memory that needs to be reserved is decided at runtime and it is managed by the program, not the operating system. Access to the heap is slower but the space on the heap is only limited by the size of virtual memory.