A PROGRAMMER'S PERSPECTIVE

MINSEOK SONG

Code Security

- In the beginning of the chapter, B & H introduces two scenarios regarding vulnerability.
- 1) Unlike Java, C doesn't feature a garbage collector. However, when declaring an array such as char kbuf[KSIZE];, the array kbuf can be initialized with garbage values. In two's complement representation, there's a significant difference between -MSIZE and MSIZE due to the most significant bit being set to 1 for negative numbers. This discrepancy can introduce vulnerabilities and potentially lead to memory leaks.
- 2)If we do malloc(ele_cnt * ele_size) but if each argumnent is very large, the program might not allocate the desired memory size due to integer overflow.

Two's Complement

arithmetic and Underflow/Overflow. When adding two numbers in two's complement representation, there's a potential for underflow and overflow.
Let's denote the result of addition as $TAdd_w(u, v)$
where w is the bit width of the numbers, and u and v are the numbers being added. Then, we can define $TAdd_w(u, v)$ as:

$$TAdd_w(u,v) = \begin{cases} u+v+2^w & \text{if } u+v < TMin_w \\ u+v & \text{if } TMin_w \leqslant u+v \leqslant TMax_w \\ u+v-2^w & \text{if } u+v > TMax_w \end{cases}$$

Here, $TMin_w$ and $TMax_w$ are the minimum and maximum values representable with w bits in two's complement, respectively.

For instance:

- When we add two values (i.e., $0\dots$ and $0\dots$) and get a result starting with $01\dots$, it indicates an overflow (desired: $a-2^{w-1}+b-2^{w-1} = a+b-2^w$, but what we get: $a+b+2^w-2^w = a+b$, where a and b are bits after the most significant bit, so subtract 2^w).
- When we add two values (i.e., $1\dots$ and $1\dots$), and the result is $1\dots$, it indicates an overflow (desired: y, but what we get: $-(2^w - y) = y - 2^w$, so add back 2^w).

Remark 1.
- Multiplication and division are a bit more involved; but essentially, we use addition and bit-shifting operations to accomplish tasks (we need to be careful when dividend is negative number and introduce the concept of "bias").
- When sign-extending an integer using the >> (right shift) operator, the most significant bit (often called the sign bit) is replicated to fill in the shifted positions.