整数溢出

原理：

由于整数在内存里面保存在一个固定长度 (在本章中使用32位)的空间内，它能存储的最大值就是固定的，当尝试去存储一个数，而这个数又大于这个固定的最大值时，将会导致整数溢出。

举个例子，有两个无符号的整数，num1和num2，两个数都是32位长，首先赋值给num1 一个32位整数的最大值，num2被赋值为1。然后让num1和num2相加，然后存储结果到第3个无符号32位的整数num3，代码如下：

num1 = 0xFFFFFFFF;

num2 = 0x00000001;

num3 = num1 + num2;

很显然，num1的值是：11111111 11111111 11111111 11111111；

num2的值是：00000000 00000000 00000000 00000001；

两者相加，得到结果为：00000000 00000000 00000000 00000000。因此，num3中的值是0，发生了整数溢出。

此时，如果一个整数用来计算一些敏感数值，如缓冲区大小或数组索引，就会产生潜在的危险。

不过，并不是所有的整数溢出都可以被利用，毕竟，整数溢出并没有改写额外的内存；但是，在有些情况下，整数溢出将会导致"不能确定的行为"，由于整数溢出出现之后，很难被被立即察觉，比较难用一个有效的办法去判断是否出现或者可能出现整数溢出。

就发现的难度而言，和缓冲区溢出相比，整数溢出更加难被发现。因此，即使是审核过的代码，有时候也难以幸免。

综上所述，一言以蔽之，整数溢出是尝试存储一个很大的数到一个变量中，由于这个变量能够存储的数值范围太小，不足以存储那个很大的数，造成溢出。

漏洞程序：

int I;

unsigned short s;

s = I;

s为整数溢出点，当值大于65535时就会发生溢出，溢出过后就可以改变程序执行结果造成攻击。

攻击程序：

send(65536)

出发整数溢出漏洞。