

In order to exploit the exercise, we first take a look of the source code.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello, which product do you want to buy?\n");
    printf("1) iPhone 12\n");
    printf("2) iPhone 12 Pro\n");
    printf("3) iPhone 12 Pro Max\n");

    // Get item
    int item_choice;
    scanf("%d", &item_choice);

    printf("Great device, how many?\n");
    int item_quantity;
    scanf("%d", &item_quantity);

    if (item_quantity ≤ 0) {
        printf("You should buy at least one iPhone!\n");
        return -1;
    }

    int insurance = 1200;
    if (item_choice == 3)
    {
        int price = 1500*item_quantity + insurance;
        if (price == 0) {
            printf("You solved the problem!\n");
            printf("The iPhone Max Max is yours!\n");
            return 1;
        }
        printf("You have to pay €%d\n", price);
    }
    else
    {
        if (item_quantity > 3) {
            printf("You can buy maximum 3\n");
            return -1;
        }
        int price = 1000*item_quantity;
        printf("You have to pay €%d\n", price);
    }
    return 0;
}
```

integer variable
to be exploited

guard for negative inputs

exploitable code

more guards...

This code has an evident flaw in the `int price = 1500 * ...` line. An arbitrarily large `item_quantity` variable, which is also an `int`, can cause the `price` variable to overflow easily due to the multiplication with 1500.

In order to exploit the code, we can follow two paths. Both are equally easy. The first would not need even the knowledge of the source code: we can just brute force our way by repeatedly running the program and trying increasing values of `item_quantity` until the exploit is performed. The second way is described here.

```
int i = 0;
int tmp;
while(1) {
    tmp = 1500 * i + 1200;
    if (tmp == 0) {
        printf("%d\n", i);
        return 12;
    }
    i++;
}
```

This short code snippet brute forces the formula used in the source code of the program, in order to find a suitable i that when substituted makes price equal to 0. Running it eventually finds an i value of 214748364.

```
Hello, which product do you want to buy? -9000 = 1500 * 214748358 + 1200
1) iPhone 12 -7500 = 1500 * 214748359 + 1200
2) iPhone 12 Pro -6000 = 1500 * 214748360 + 1200
3) iPhone 12 Pro Max Max -4500 = 1500 * 214748361 + 1200
3 -3000 = 1500 * 214748362 + 1200
0 = 1500 * 214748363 + 1200
0 = 1500 * 214748364 + 1200
Great device, how many? 1500 = 1500 * 214748365 + 1200
214748364 3000 = 1500 * 214748366 + 1200
You solved the problem 4500 = 1500 * 214748367 + 1200
The Iphone Max Max is yours 6000 = 1500 * 214748368 + 1200
9000 = 1500 * 214748369 + 1200
9000 = 1500 * 214748370 + 1200
```

Plugging this value in the program completes the exploit and the exercise.