Solution to Integer Overflow Attack

Steps

1. Start the integer_overflow program with command *gcc integer_overflow_exercise.c*

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
```

2. Entering an id of the item we want to buy we will be prompted with question asking the number of items we want to buy. If we enter an arbitrary number, it will do an arithmetic calculation and show us the price.

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
2
Great device, how many?
3
You have to pay €3000
```

3. Now lets try to exploit this. If we try to buy a large number of items, we'll get a prompt saying we cannot buy more than three then the program exits. So here, we know there is a checker on the number entered by the user. So we clearly cannot exploit this

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
1
Great device, how many?
5
You can buy maximum 3
```

4. So we do same for the rest options that is, 2&3. We able to get a breakthrough on option 3 since there is no checker here.

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
3
Great device, how many?
5
You have to pay €8700
```

5. So entering an extremely large value, it will round up and turn into a negative value hence we will get a different error message. SO there is a checker that checks that

the number entered is a positive value

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
3
Great device, how many?
700000000000000000
You should buy at least one Iphone!
```

6. Now entering a value that is not too big or small, we will have the result of a negative value.

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
3
Great device, how many?
400000000
You have to pay €-1295420240
```

7. Let's take a look at the source code, exploit it and fix it. This is the logic behind the price calculation so if we need to make price to be equal to zero

```
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);
```

8. So we can try some arithmetic manipulations like having 0= 1500*x +1200 where x is the number we entered and to consider overflow we use the mod function and consider all the possible number of bit that's needed we'll get 0= 1500*x +1200 mod 2^32. Go to wolfram alpha to convert this calculation into something meaningful, we get x = 214748364. Now lets try it

9. Solved

```
homework-5$ ./a.out
Hello, which product do you want to buy?
1) IPhone 11
2) IPhone 11 Pro
3) IPhone 11 Pro Max Max
3
Great device, how many?
214748364
You solved the problem
The Iphone Max Max is yours
```

10. Mitigation: Adding the same checker like the other options have fixes the vulnerability.

```
int insurance = 1200;
if (item_choice == 3)
        if (item_quantity > 3) {
                printf("You can buy maximum 3\n");
                return -1;
        }
{
        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;
```