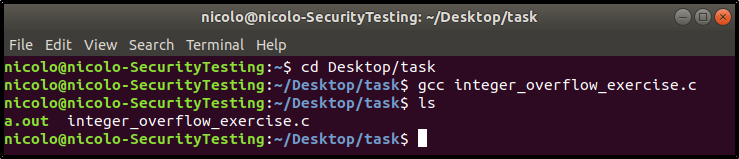
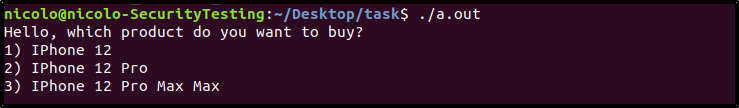
Vinci Nicolò 220229

INTEGER OVERFLOW

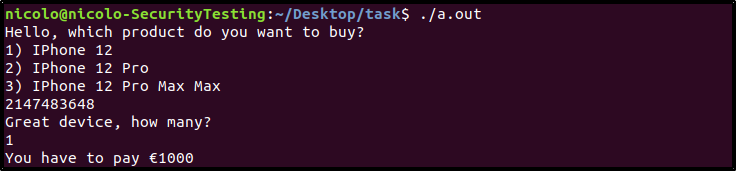
First of all, I download the script and I compile it with gcc.



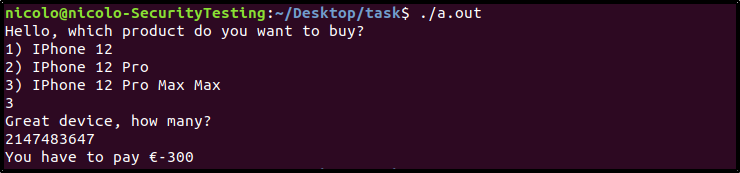
Then, the file a.out will be created and I can execute it.



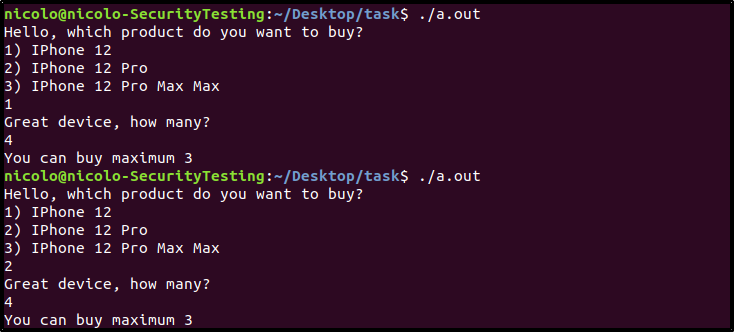
I observe from the source code that there is an operation to calculate the phone price. Moreover, the first input is not checked, so I can put a large number to obtain an integer overflow. However, it is useless for having the IPhone 12 Pro Max Max for free.



Instead, the second input is checked so that I can not put a negative integer number. The second input represents the number of devices that I want to buy and it is used to calculate the final price. So, I should overflow that integer for buying the IPhone 12 Pro Max Max. Indeed, if I insert 2147483647 as input, I will get a negative total price. It means that the integer overflows.



If I choose the other two phones, I can not overflow the integer, because I can insert at most 3 as input for the quantity.



From the source code I notice that the total price is calculated as:

Where insurance is equal to 1200 and item\_quantity is my second input. I solve the following equation for buying for free the phone:

Where x is the number that I will insert as item\_quantity and 232 is the all possible number that an integer can represent in C. So, it is a modular equation and I rewrite it as:

First of all, I check if this equation has solutions. To verify it, I have to find the MCD between 1500 and 232:

Then, I observe that 4 divides -1200, indeed:

So, the equation has solutions. I notice that 4 divides 1500, -1200 and 232, thus I rewrite the equation as:

Now, I have to find the multiplicative inverse to solve it in order to isolate the x on the left. For this, I have to solve:

Where c is the multiplicative inverse. The congruence means that 1073741824 divides (375\*c -1). I apply the Bezout identity between 375 and 1073741824 to solve it. Thanks to an online tool, I find:

Now, I can multiply both sides for c in the initial equation:

375\*c is equal to 1 for what I have just done before:

I observe that 42090679500 > 1073741824, so I can reduce it exploiting the module operation:

Because:

At the end, the all possible solutions are:

If I take k = 0, I test as input integer 214748364.

