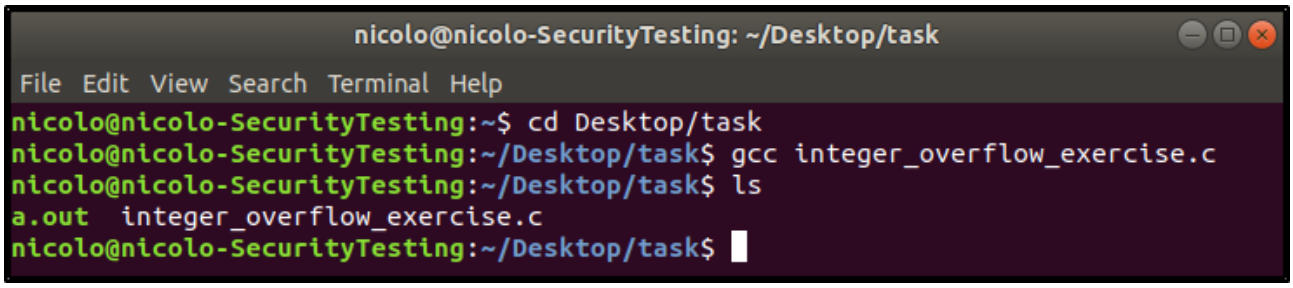
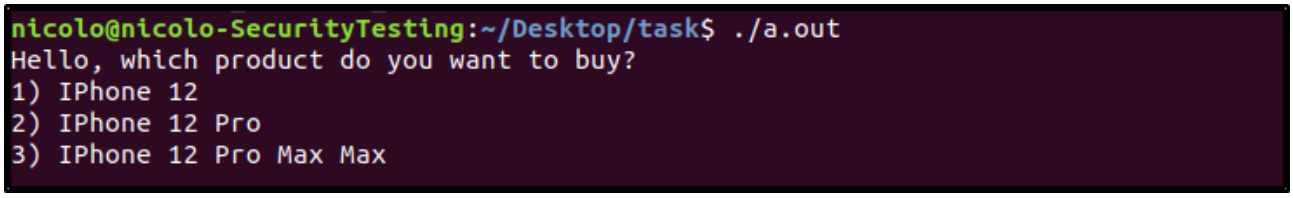
Vinci Nicolò 220229

INTEGER OVERFLOW

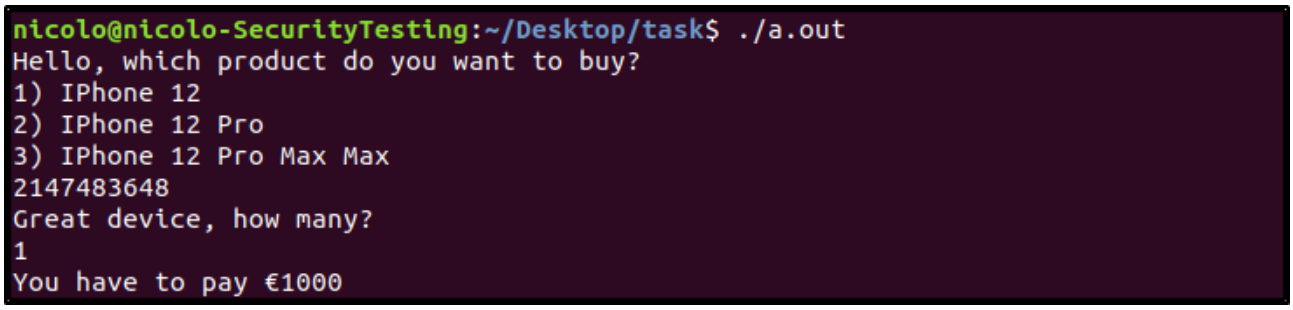
First of all, I download the script and I compile it with gcc.



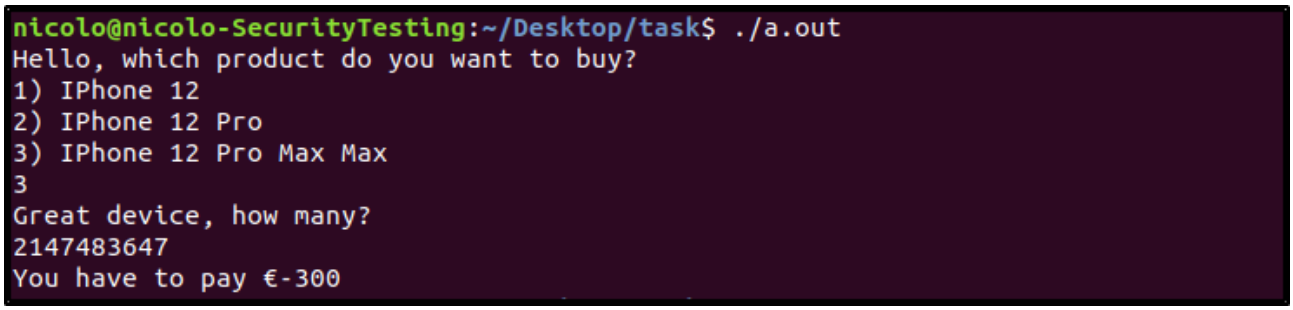
Then, the file a.out will be created and I can execute it.



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.

```
nicolo@nicolo-SecurityTesting:~/Desktop/task$ ./a.out
Hello, which product do you want to buy?
1) IPhone 12
2) IPhone 12 Pro
3) IPhone 12 Pro Max Max
1
Great device, how many?
4
You can buy maximum 3
nicolo@nicolo-SecurityTesting:~/Desktop/task$ ./a.out
Hello, which product do you want to buy?
1) IPhone 12
2) IPhone 12 Pro
3) IPhone 12 Pro Max Max
2
Great device, how many?
4
You can buy maximum 3
```

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

$$price = 1500 * item\_quantity + insurance$$

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

$$1500 * x + 1200 = 0 \; mod \; 2^{32}$$

Where x is the number that I will insert as item_quantity and 2^{32} is the all possible number that an integer can represent in C. So, it is a modular equation and I rewrite it as:

$$1500 * x = \; -1200 \; mod \; 2^{32}$$

First of all, I check if this equation has solutions. To verify it, I have to find the MCD between 1500 and 2^{32}:

$$MCD(1500, 2^{32}) = 4$$

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

$$-\frac{1200}{4} = \; -300$$

So, the equation has solutions. I notice that 4 divides 1500, -1200 and 2^{32}, thus I rewrite the equation as:

$$\frac{1500}{4} * x = \; -\frac{1200}{4} mod 2^{32}/4$$

$$375 * x = \; -300 \; mod \; 1073741824$$

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:

$$375 * c \; \equiv 1 \; mod \; 1073741824$$

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:

$$-140302265 * 375 + 49 * 1073741824 \; = 1$$

$$c = \; -140302265$$

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

$$(375 * c) * x = -300 * x \ mod \ 1073741824$$

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

$$x = -300 * c \ mod \ 1073741824$$

$$x = 42090679500 \ mod \ 1073741824$$

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

$$42090679500 \ mod \ 1073741824 = 214748364$$

$$x = 214748364 \ mod \ 1073741824$$

Because:

$$42090679500 \equiv 214748364 \ mod \ 1073741824$$

At the end, the all possible solutions are:

$$Solutions = \{ 214748364 + k * 1073741824 \}$$

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

```
nicolo@nicolo-SecurityTesting:~/Desktop/task$ ./a.out
Hello, which product do you want to buy?
1) IPhone 12
2) IPhone 12 Pro
3) IPhone 12 Pro Max Max
3
Great device, how many?
214748364
You solved the problem
The Iphone Max Max is yours
```