Vending Machine by Yekong

nail_

Preface

This CTF task isn't the most complicated CTF and has quite a unique way to be solved. We are given the tip: "*What is the num less than 0?*" which we should keep in mind through solving the task.

Required Tools:

- Ghidra (Not needed for the flag)

Recon

On executing the file, we are greeted with a menu:

```
[nail_@nailCPU 2-VendingMachine]$ ./vending_machine
You have 10 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             10
(2) Pepsi       5             10
(3) Flag        100           1
(4) Exit
```

Assuming we attempt to claim the flag instantly, we are greeted with:

```
3
Not enough coins
You have 10 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             10
(2) Pepsi       5             10
(3) Flag        100           1
(4) Exit
```

If we attempt to claim either Coca-Cola or Pepsi, we are prompted to input an amount. Assuming we love Coke and buy 2, we end up with 0 coins.

```
1
How many do you want?
2
You have 0 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             8
(2) Pepsi       5             10
(3) Flag        100           1
(4) Exit
```

Now, assuming we're indebted to Coke and it's army of sugar and coca leaves, we want to return a can of coke, inputting -1 cokes.

```
1
How many do you want?
-1
You have 15 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             11
(2) Pepsi       5             10
(3) Flag        100           1
(4) Exit
```

Alas! We have 15 coins and 11 drinks! But how is this?

This is easily explainable:
Assuming x is our input, I is our initial value and y is our final value,

$$I - x = y$$

If $x = -1$ and $I = 10$, this can be substituted as

$$10 - -1 = y$$

where $y = 11$

The same logic can be applied for the coins:
Assuming x is our requested items, I is our initial value and P is our price and y is our final value,

$$I - (Px) = y$$

If $x = -1$, $I = 10$, $P = 5$, this can be substituted as

$$10 - (5 \times -1) = y$$

where $y = 15$

Thus, we can substitute $y = 100$ for the coin equation to find x,

$$10 - (5 \times x) = 100$$

where $x = -18$

Inputting "-18" gives us 100 coins, allowing us to obtain the flag. `flag{v3nd1nd_m4ch1n3_1s_4w3s0m3}`

```
1
How many do you want?
-18
You have 100 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             28
(2) Pepsi       5             10
(3) Flag        100           1
(4) Exit

3
How many do you want?
1
flag{v3nd1ng_m4ch1n3_1s_4w3s0m3}
You have 0 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             28
(2) Pepsi       5             10
(3) Flag        100           0
(4) Exit
```

Further Analysis

Inputting "-18" is not the only way to form an overflow. The following is the overthinker's method:

Using Ghidra's Code browser, extracting the `vending_machine` file into a folder and placing the file in Ghidra the decompiled file in assembly is generated.

We see references to "Elf64" in the assembly code, suggesting that the code supports a 64-bit instruction set (which is exclusive to UNIX systems, but this is irrelevant). Keep this in mind for later.

Looking at the symbol tree, we see reference to Rust (based) and `rustc` compiler functions.

A simple scan through of *The Rust Programming Language* section on data types shows that Rust contains 6 separate lengths of integer data, either signed or unsigned.

Length	Signed	Unsigned
8-bit	`i8`	`u8`
16-bit	`i16`	`u16`
32-bit	`i32`	`u32`
64-bit	`i64`	`u64`
128-bit	`i128`	`u128`
arch	`isize`	`usize`

Recalling that the code supports a 64-bit instructions set and that it's possible to obtain unsigned integers we can use from signed 8 to 64 bit integers in order to find the most efficient way to cause an integer overflow.

The largest integer per bit set is given as:

Length	Amount of Values	Highest Integer
8-bit	2^8	255
16-bit	2^16	65,535
32-bit	2^32	4,294,967,295
64-bit	2^64	1.844674407x10^19

As 64 bit is way too inefficient to input so it's either 255, 65,535 or 4,294,967,295.

As 5 coins are removed from us each time we order something, to find the value we would need to divide the integer limit by 5.

8-bit

$$255 \div 5 = 51$$

Inputting this as the amount, we get

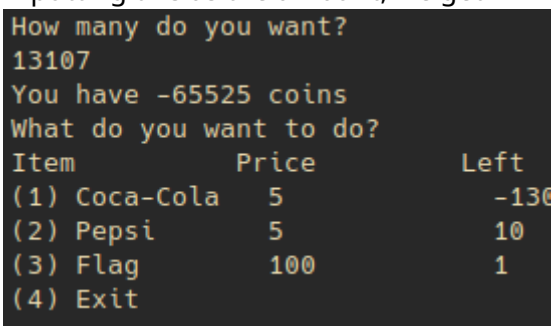
```
How many do you want?
51
You have -245 coins
What do you want to do?
Item          Price         Left
(1) Coca-Cola   5             -41
(2) Pepsi       5             10
(3) Flag        100           1
(4) Exit
```

Which doesn't hold up.

16-bit

$$65535 \div 5 = 13107$$

Inputting this as the amount, we get

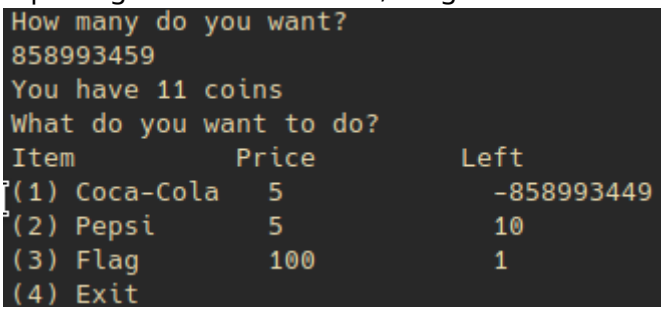


Which doesn't hold up.

32-bit

$$4294967295 \div 5 = 858993459$$

Inputting this as the amount, we get



Which doesn't hold up BUT it somehow goes to 11. This suggests the limit has been hit prior. Using the formula

$$I - (Px) = y$$

we substitute values to get

$$10 - (5x) = 0$$

where $x = 2$

Dividing 858993459 by 2, we get 429496729.5
Rounding 429496729.5 to the nearest integer, we get 429496730
Adding 1 due to finding the maximum value of 32 bit is $2^{32} - 1$ is 429496731
Adding 1 due to 2's complement, we get 429496732

If we input 429496732, we get 429,496,732 coins. Enough for a flag.