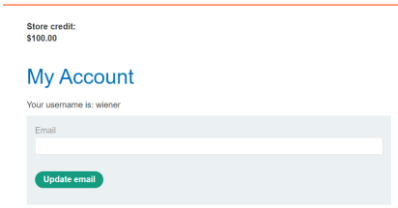
Application logic vulnerabilities

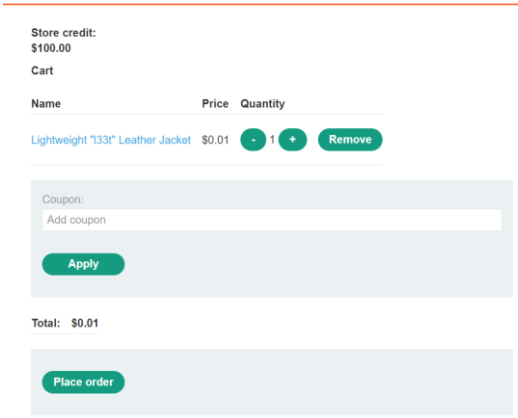
LAB 45 Excessive trust in client-side controls

I was provided with an account of user `wiener:peter` that has $100 on his account:

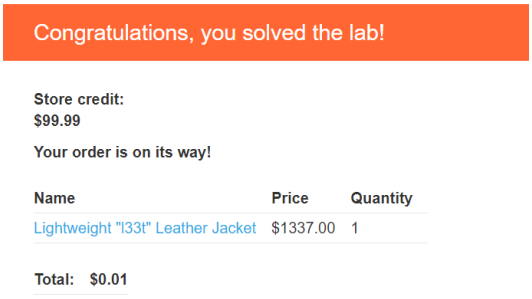


I was asked to buy “Lightweight "l33t" Leather Jacket” that costs $1337.
To do so, I tested the order making process by adding goods to the cart and trying to spend money. I have noticed, that, when adding an item to the cart, it contains price parameter:

I changed the value of this parameter to “1” and observed that the price in the cart has changed:

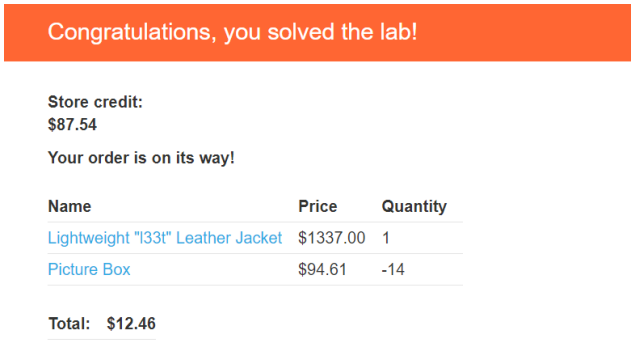


What is left is to place the order and demonstrate, how order has been made with exploiting site’s purchase making logic:



LAB 46 High-level logic vulnerability

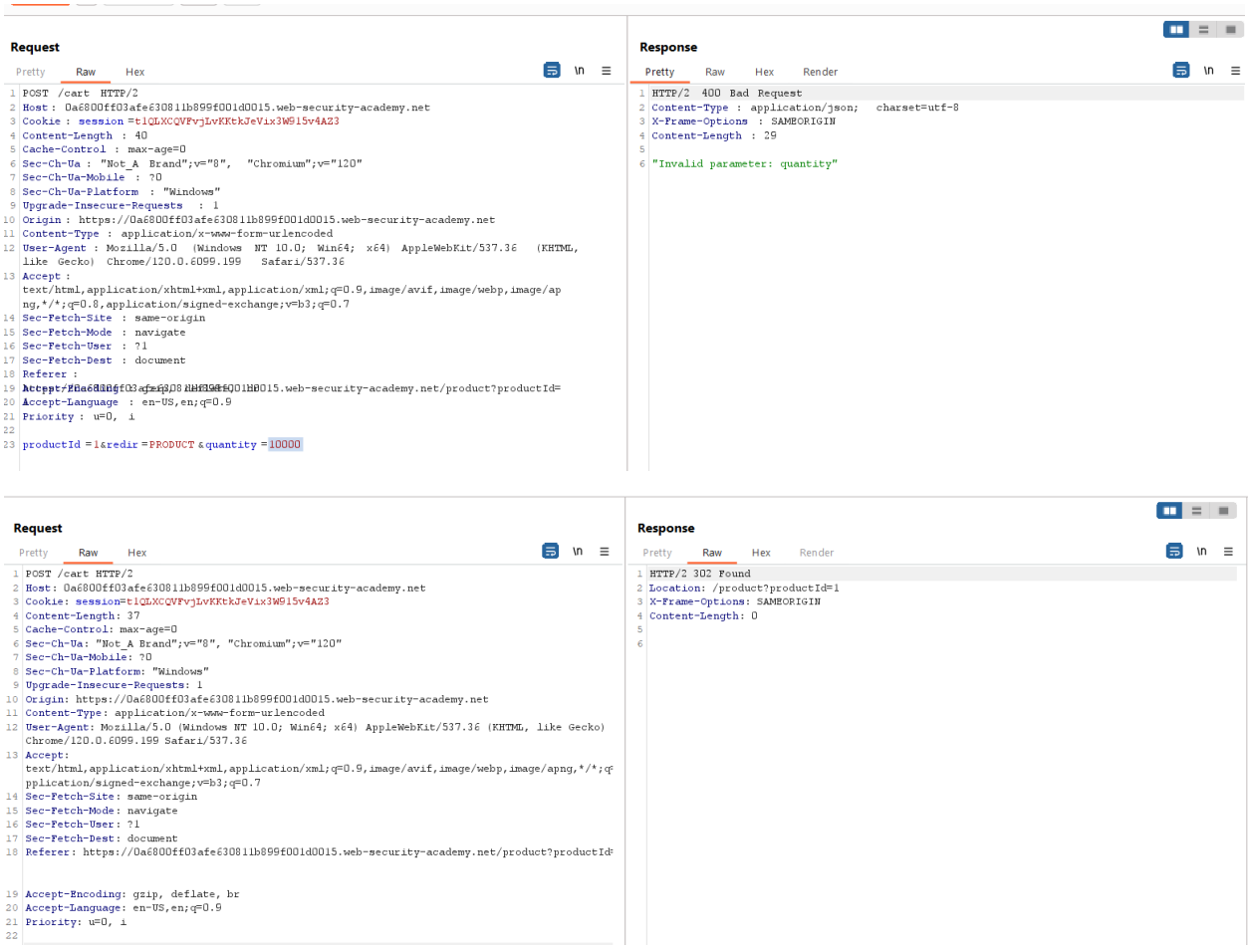
This lab is similar to the previous one, however, this time it does not contain price parameter. However, there is 'quantity' parameter left when adding an item to a cart. Changing its value does change the value of the item in the cart, and there was no handling of passing negative quantities that could lead to decreasing of the price of items in the cart. In such a way, I was able to decrease the amount of leather jacket and purchase it for the $12 (it's possible to make it even cheaper):



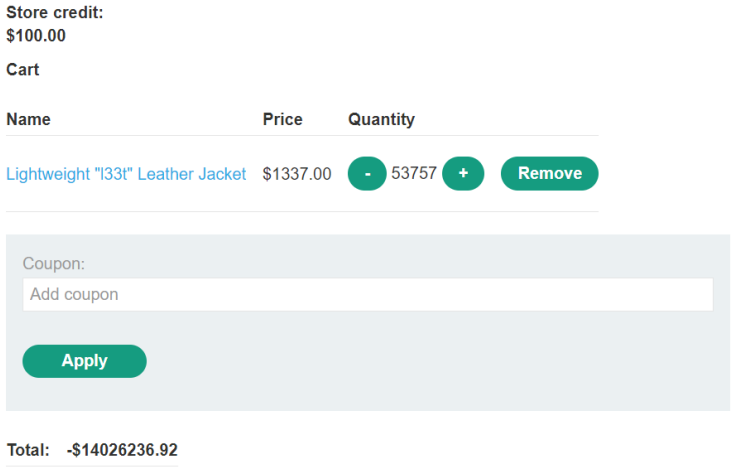
However, there is a correct assumption tat price of cart cannot be lower than 0, so I couldn't add myself extra money in such a way.

LAB 47 Low-level logic flaw

In this lab, everything remains the same, but this time negative quantities are handled nicely: if there is negative quantity, then it will remove the item from cart. I tried to go in opposite way and increase amount to huge numbers and discovered that maximum quantity that could be set is 99:



Automatically repeating this request in Burp Intruder and increasing the amount by 99, I noticed that at certain point the price becomes negative:

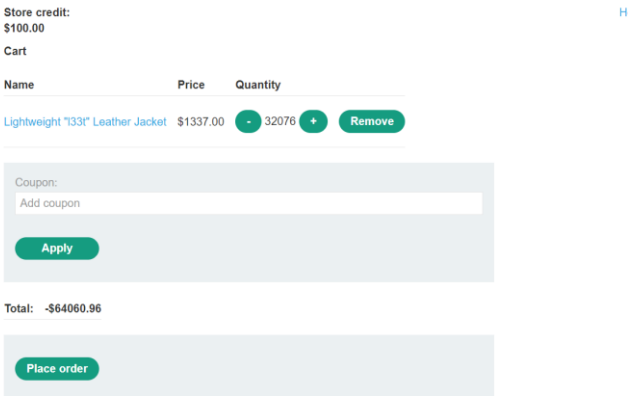


This could mean that price number becomes that large, so it exceeds its data type limit and loops back to negative values back to 0 and so on. Knowing that, typically, integer limit in most languages is normally [-2,147,483,647 ; 2,147,483,647] and for float it's 3.4E +/- 38, I can mathematically compute the quantity number to loop back to zero. Before I start with integers, I noticed that price is kept without floating points, so our jacket will cost $133700 instead of $1337.00:

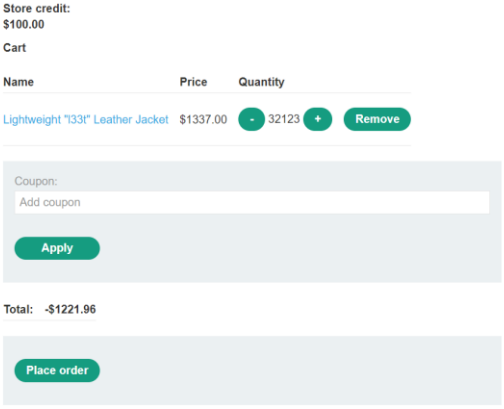
133700 x 99 = 13 236 300

2 x 2,147,483,647 / 13 236 300 = 324,5

Therefore, I will add order of 99 jackets exactly 323 times to be sure that price remains negative:



I need to add 6406096/13700 = 47 jackets to cart to be maximally close to $0:



Now, I will simply select any other product (obviously, not too expensive, easy scalable) from the catalogue to reach $0:

Store credit:
$100.00

Cart

Name	Price	Quantity	
Lightweight "l33t" Leather Jacket	$1337.00	- 32123 +	Remove
Safety First	$46.36	- 27 +	Remove

Coupon:
Add coupon

Apply

Total: $29.76

Place order

All is left is to place order:

Congratulations, you solved the lab!

Store credit:
$70.24

Your order is on its way!

Name	Price	Quantity
Lightweight "l33t" Leather Jacket	$1337.00	32123
Safety First	$46.36	27

Total: $29.76

LAB 48 Inconsistent handling of exceptional input

Using Burp, I fuzzed the website on its contents and /admin was discovered. It contained the following message:

Admin interface only available if logged in as a DontWannaCry user

Since I was not provided with any account, I created it myself to test the website as registered user this time. Having multiple accounts registered, I found out that user's email (which can be a very long string) on dashboard is being truncated when displayed:

My Account

Your username is: hackr1

Your email is:
baaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

This account was registered with email (436 character long):

baaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaa@exploit-0a5b001a03850a7e83a6db06014f003b.exploit-server.net

The application truncated it up to 255 characters, so my idea was to try to register a user with @dontwannacry.com domain with last symbol 'm' on 255th position:

blablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablab
lablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablablabl
ablablablablablablablablablablablablla@dontwannacry.com.exploit-
0a5b001a03850a7e83a6db06014f003b.exploit-server.net

Having such a payload, I received a registration confirmation email:

Next, I checked the account and discovered that admin panel is now available to me:

Now I can conduct malicious actions, such as deletion of accounts: