CRYPTOGRAPHY ASSIGNMENT - 3

In the partial fulfilment of B. tech -III-year course

requirement of

Subject: Cryptography

BML MUNJAL UNIVERSITY™

FROM HERE TO THE WORLD

Submitted To:

Dr. Purnendu Shekar Pandey

School of Engineering & Technology

BML Munjal University

Submitted by:

Name: P. Mohana uma sushmanth

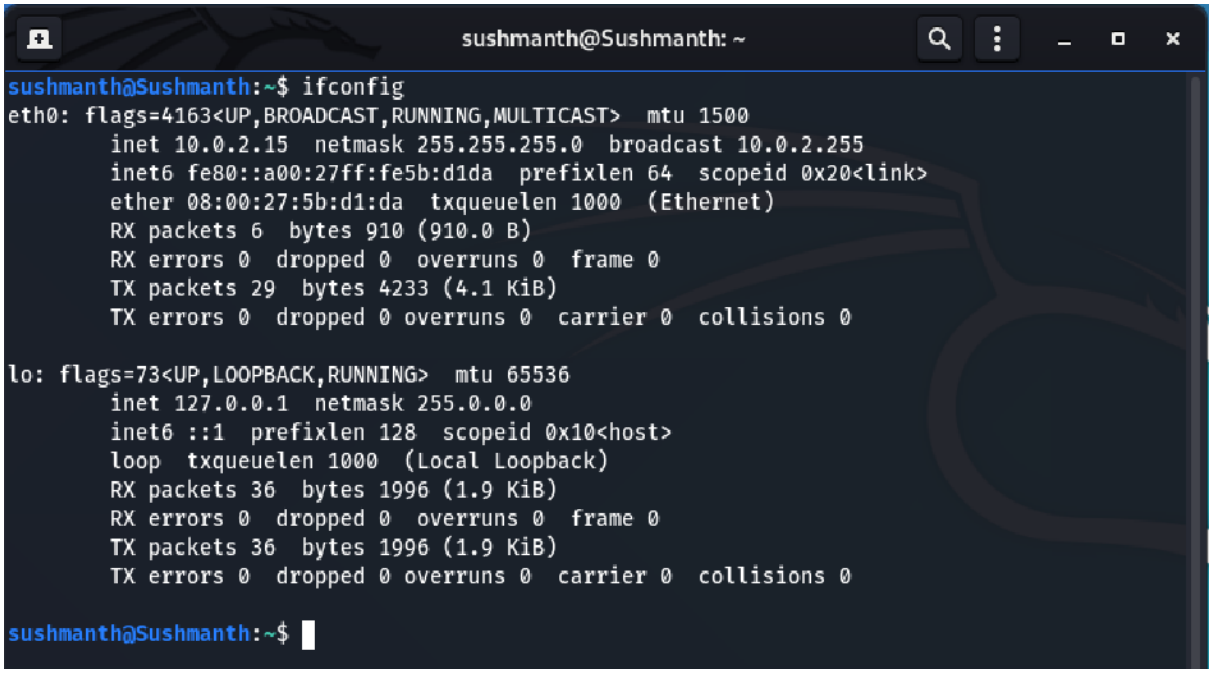
ID NO: 1800248C203

Section: CSE – 4

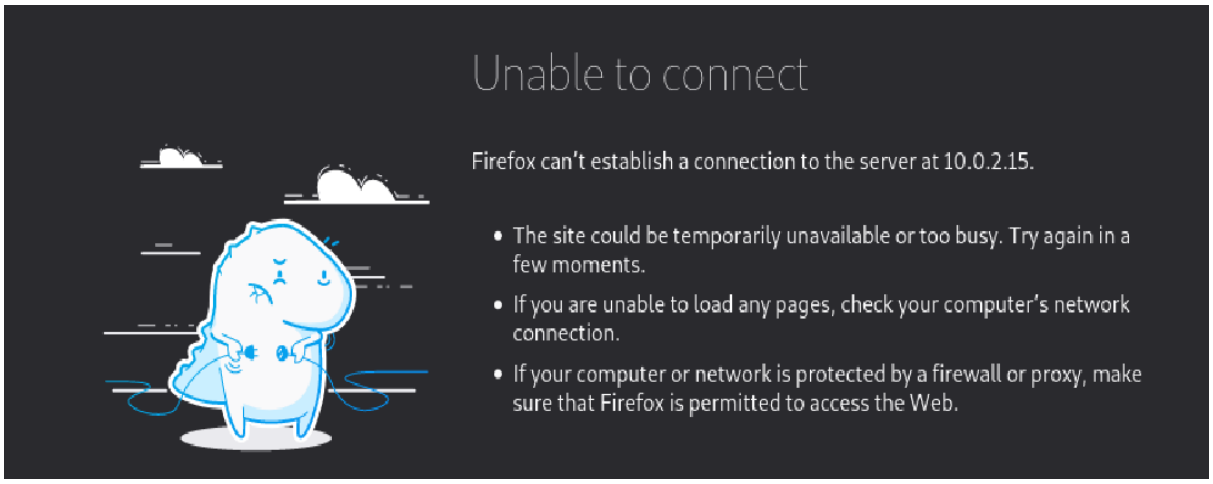
Part – 1 Web application firewalls

In this activity you will be using Kali Linux to set up a simple web and MySQL server. Also, will then undertake a web attack towards the web server and observe the outcomes.

Firstly, we have to know our IP address. We can be able to get to know the IP address by using **ifconfig** in the terminal. On executing the command, we will get the following outputs.



Now, open a web browser within the virtual machine and type the IP address as noted in Step 2. At this point, the connection should fail with an “Unable to connect” message. This is indicative that there is no web server for your browser to connect to.

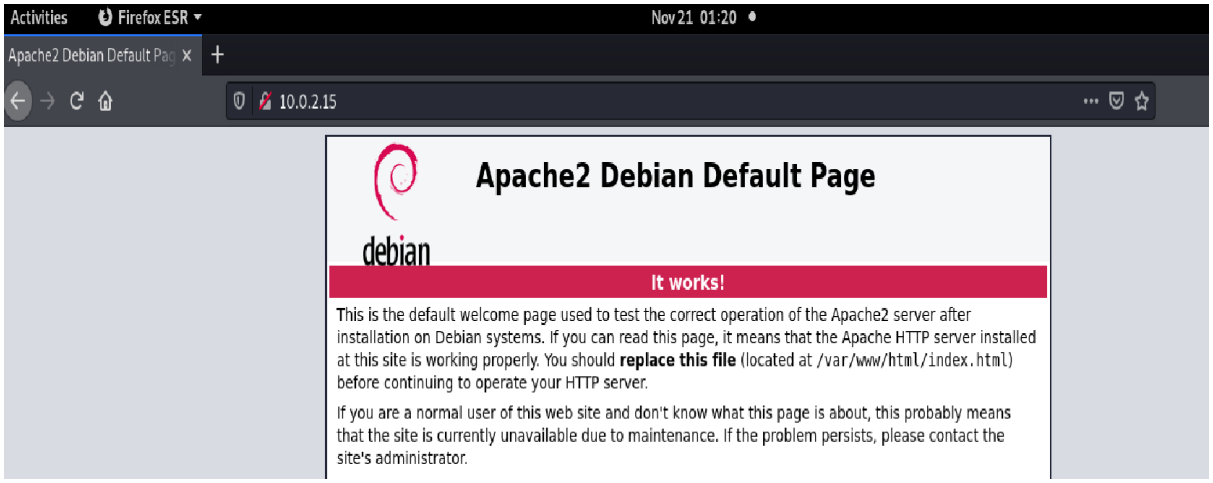


Next you will start the web server service within Kali Linux. The web server service will permit the virtual machine to host web pages and be accessible through a web browser. To start the web server, open your terminal and type the following command given bellow.

service apache2 start

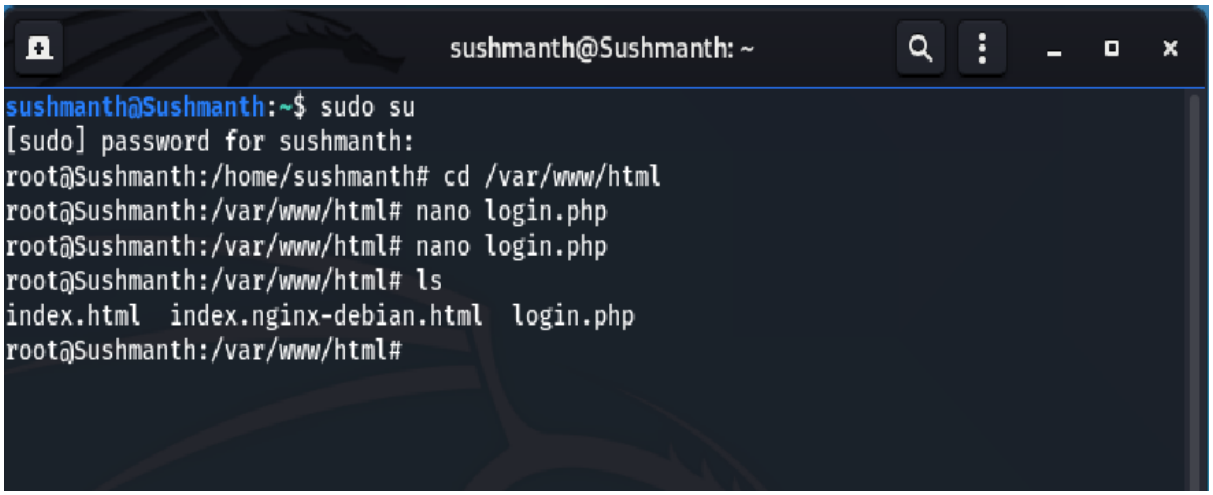
```
sushmanth@Sushmanth: ~
sushmanth@Sushmanth:~$ service apache2 start
sushmanth@Sushmanth:~$ service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor pres>
     Active: active (running) since Sat 2020-11-21 01:18:23 CST; 39s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 67465 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/S>
   Main PID: 67476 (apache2)
      Tasks: 7 (limit: 3462)
     Memory: 17.3M
     CGroup: /system.slice/apache2.service
             ├─67476 /usr/sbin/apache2 -k start
             ├─67477 /usr/sbin/apache2 -k start
             ├─67478 /usr/sbin/apache2 -k start
             ├─67479 /usr/sbin/apache2 -k start
             ├─67480 /usr/sbin/apache2 -k start
             ├─67481 /usr/sbin/apache2 -k start
             └─67482 /usr/sbin/apache2 -k start
lines 1-16/16 (END)
```

Assuming the apache2 service started with no errors. Open your web browser again and refresh the web page. At this point you should see the message “Apache2 Debian Default Page”. This informs us that the web server is running correctly.

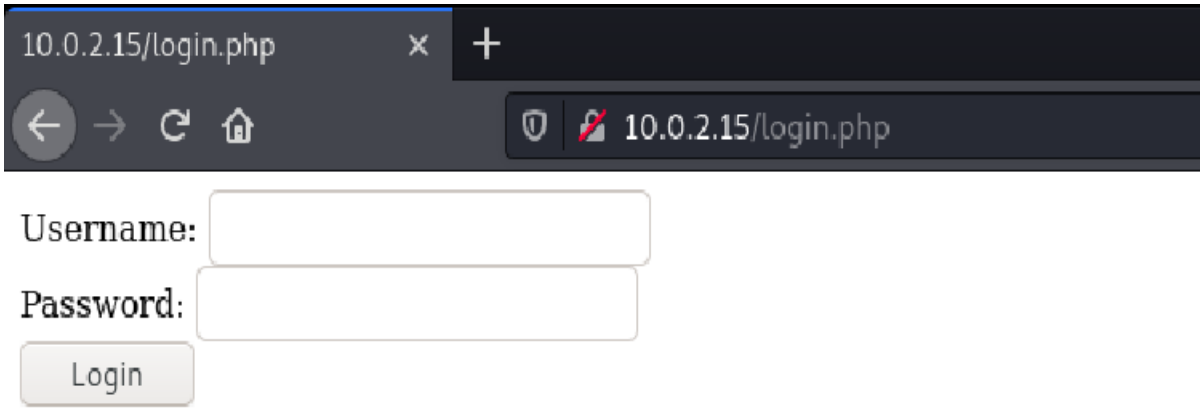


Next, create a vulnerable web page using a PHP script. Using terminal navigate to the /var/WWW/HTML directory by using **cd** command and create a new file in that location using the following command.

nano login.php



Write the php code which was given in resources folder in the login.php file and save it. Now, open your web browser again. At the end of the entered IP address add **/login.php** and enter it. It will give the page as shown in the bellow.



At this point you have created a web server and created a PHP script to host the authentication page. Next you need to create a MySQL database to securely store the authentication credentials that the website will use.

Before creating a MySQL database, you must start the MySQL services by entering the following command **service mysql start** in terminal. Now, type **mysql -u root -p** command and click enter. It will ask you the password, just press enter again as there is no default password.

```
sushmanth@Sushmanth: ~
root@Sushmanth:/var/www/html# service mysql start
root@Sushmanth:/var/www/html# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 51
Server version: 10.3.24-MariaDB-2 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

MariaDB is the open source database that will be used to create and manage the authentication credentials. There are two things that will need to be done hereafter – create a new user to login to the database and create a database to populate the users for the website. From terminal run the following commands, one line at a time, you may either copy and paste the commands or write them exactly as shown here. The first set of commands creates a table and creates two users that will stored in the table.

create database mydb1;

use mydb1;

create table users(username VARCHAR(100),password VARCHAR(100));

insert into users values(‘sushmanth’, ’penumarthi’);

insert into users values(‘Mohana’, ’uma’);

quit;

```
MariaDB [(none)]> create database mydb1;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> use mydb1;
Database changed
MariaDB [mydb1]> create table users(username VARCHAR(100),password VARCHAR(100));
Query OK, 0 rows affected (0.123 sec)

MariaDB [mydb1]> insert into users values('sushmanth','penumarthi');
Query OK, 1 row affected (0.015 sec)

MariaDB [mydb1]> insert into users values('Mohana','uma');
Query OK, 1 row affected (0.022 sec)

MariaDB [mydb1]> quit;
Bye
root@Sushmanth:/var/www/html#
```

To see the users, type the following queries.

select * from users;

show columns from users;

```
MariaDB [(none)]> use mydb1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mydb1]> select * from users;
+-----------+------------+
| username  | password   |
+-----------+------------+
| sushmanth | penumarthi |
| Mohana    | uma        |
+-----------+------------+
2 rows in set (0.000 sec)

MariaDB [mydb1]> show columns from users;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(100) | YES  |     | NULL    |       |
| password | varchar(100) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
2 rows in set (0.103 sec)

MariaDB [mydb1]>
```

The second set of commands creates a ‘user’, which is required to login and access the table/database that was created in the preceding step.

mysql –u root –p

CREATE USER ‘compsec’@’localhost’ IDENTIFIED by ‘compsec’;

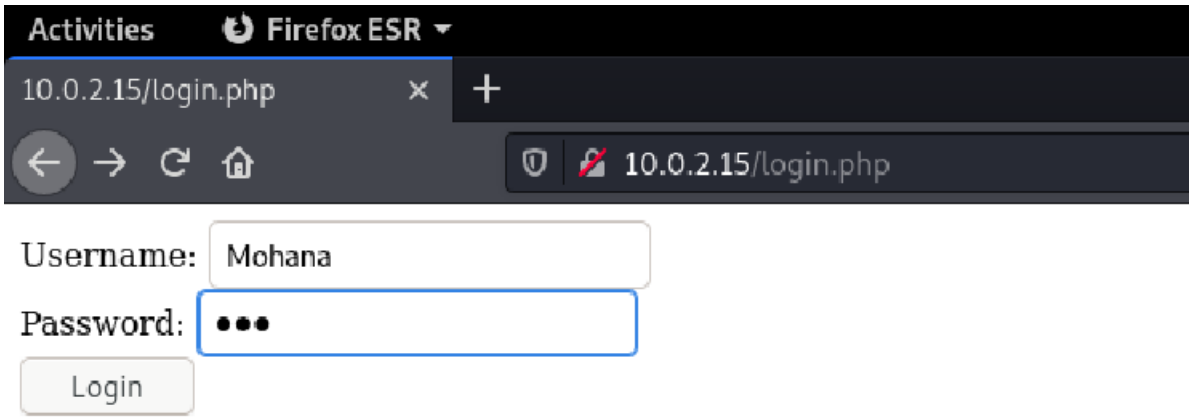
GRANT ALL PRIVILEGES ON *.* TO ‘compsec’@’localhost’;

FLUSH PRIVILEGES;

exit

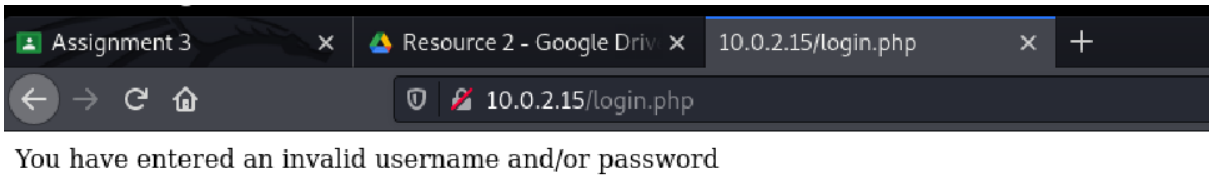
```
sushmanth@Sushmanth: /var/www/html
root@Sushmanth:/var/www/html# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 59
Server version: 10.3.24-MariaDB-2 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE USER 'compsec'@'localhost' IDENTIFIED by 'compsec';
Query OK, 0 rows affected (0.012 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'compsec'@'localhost';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> exit
Bye
root@Sushmanth:/var/www/html#
```

Open your web browser again and refresh the web page that is connected to our web server and enter the login details which are given in database as shown in below.

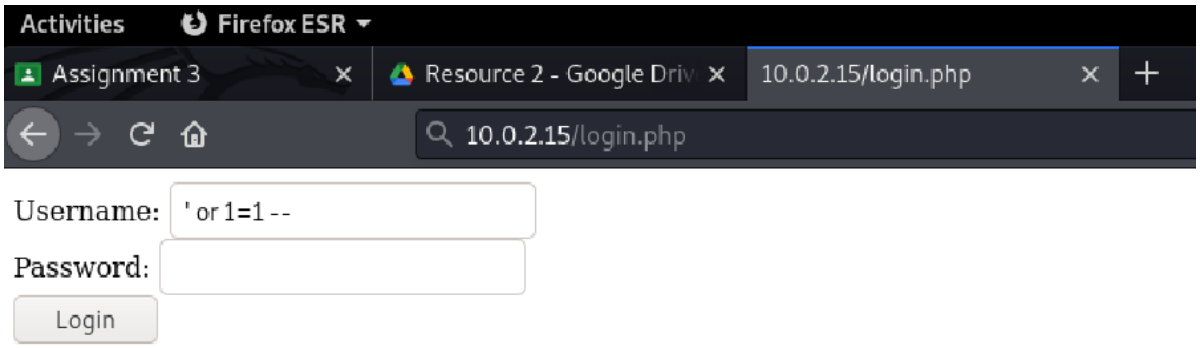


Assume you are a legitimate user. When you correctly entered the authentication credentials, you should be presented with a web page as shown below.

Now, enter an incorrect set of authentication credentials. Assuming the incorrect credentials were entered, you should see a web page as shown below. This tells you that the connection to the MySQL database is functioning correctly and each user lookup is working as intended.



Next you are going to perform a very simple SQL injection attack. Enter the following text into the username field ‘or 1=1 --. Leave the password field empty and proceed with the login.



But it gives the “Successfully login page” as shown below.

Successfully logged in

This text will only be displayed when you have logged in with valid credentials.

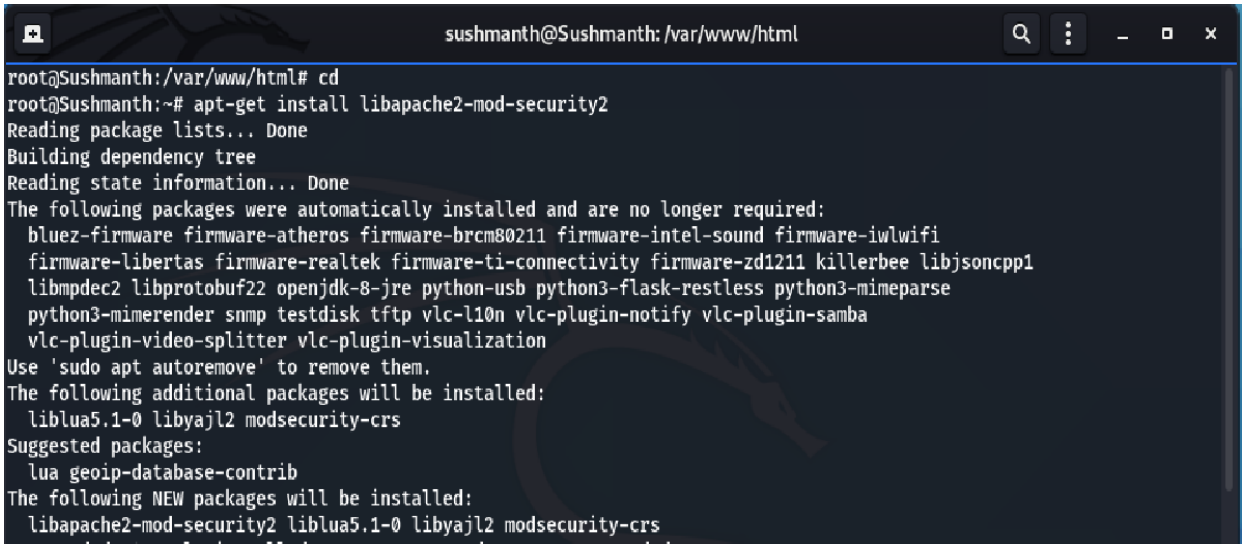
Given the successful login following the preceding SQL injection attack, it is evident that the services have not been secured correctly. As a result, you are going to install the open source, web application firewall – ModSecurity.

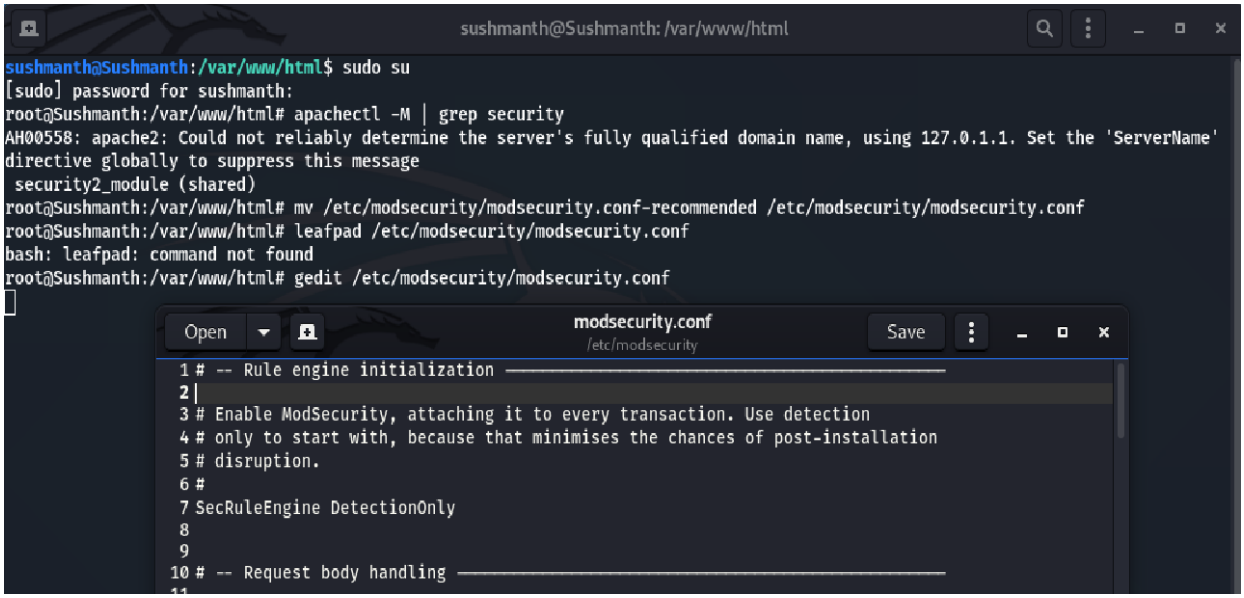
apt-get install libapache2-mod-security2

apachectl -M | grep security

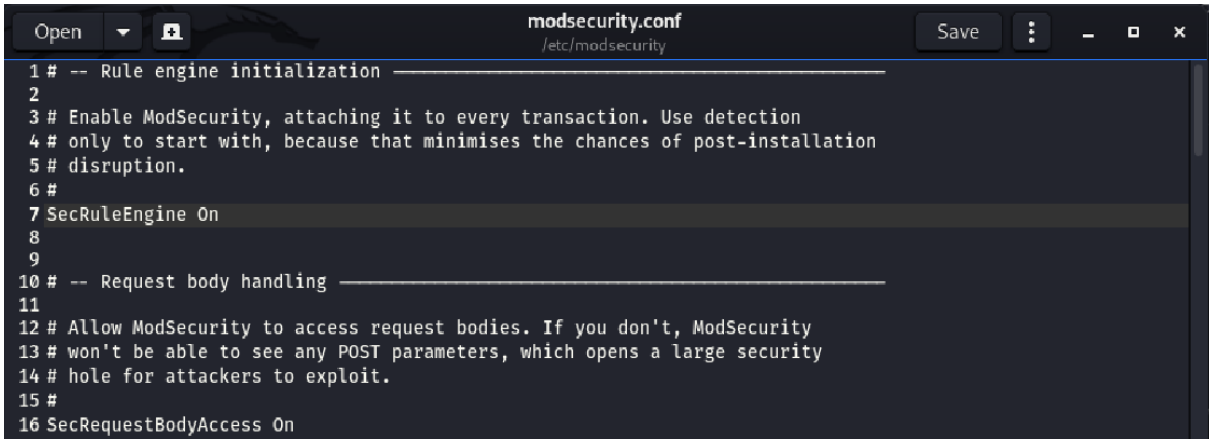
mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf

gedit /etc/modsecurity/modsecurity.conf

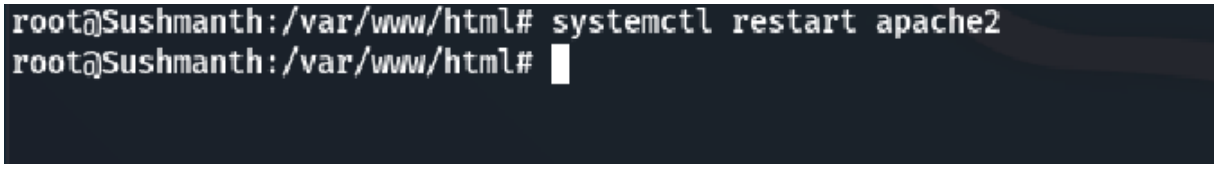




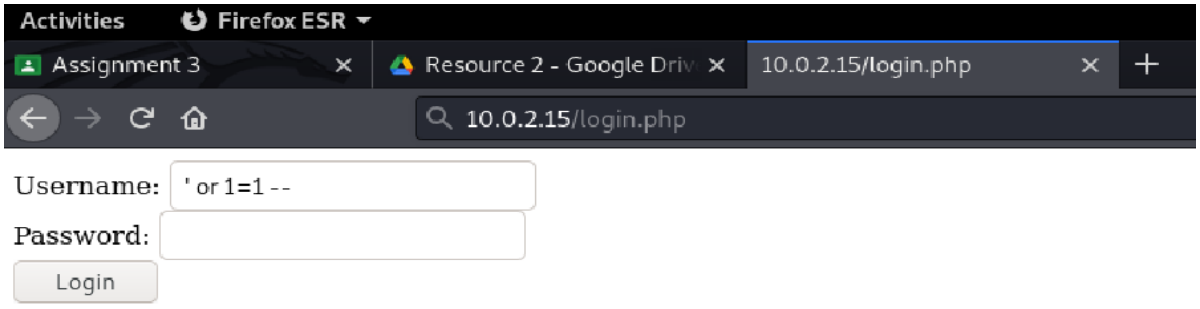
The **apachectl -M | grep security** command shown above will result in an output of security2_module (shared) if ModSecurity was successfully installed. Once the modsecurity.conf file is open within the gedit editor, find the line SecRuleEngine DetectionOnly and change it to “SecRuleEngine On” and save that file.



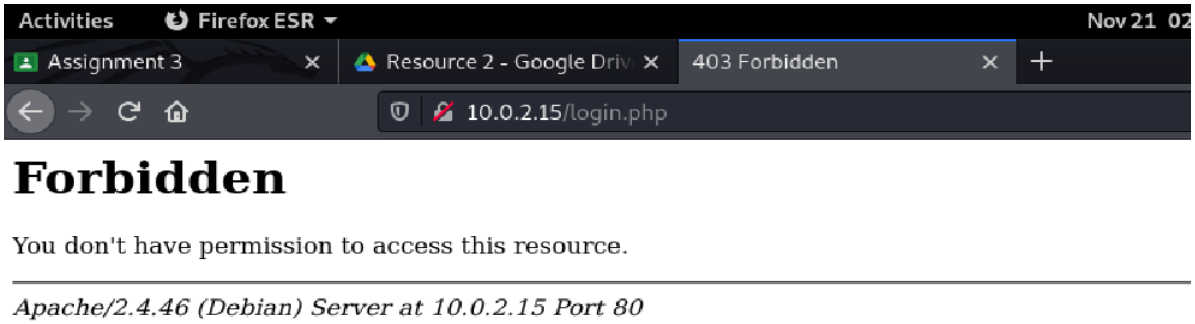
Now type the command **systemctl restart apache2** to enable the modsecurity rules set.



Now it is time to test the web server, MySQL server and the web application firewall. Load your web browser and navigate to the login page previously created. Give the text ‘or 1=1 – in username and leave password empty and check.

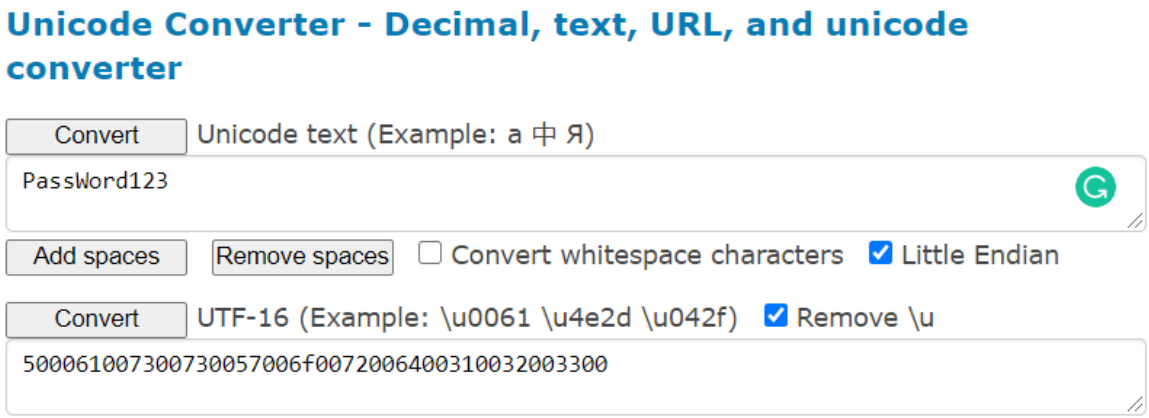


The web application firewall has detected the presence of an attack and based on its ruleset it has reacted accordingly. ModSecurity can be configured to detect/log or detect/react to an attack as it occurs.



Part – 2 Cracking a password

Here I had kept the password as ‘PassWord123’. After that I have converted into Unicode using the branch Unicode converter. After that to know the raw value here I have selected the Little Endian and remove \u from the output.



Raw value: **500061007300730057006f0072006400310032003300**

To know the MD4 value I took the raw value obtained previously and placed it in data section and kept data format as hex string in HachCalc.

MD4 value: **67a54e1c9058fca16498061b96863248.**

With the MD4 value obtained previously now using the crack station website, the password is cracked. Hence, we got the output as original password that we have entered initially.

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
67a54e1c9058fca16498061b96863248
```

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
67a54e1c9058fca16498061b96863248	NTLM	PassWord123

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

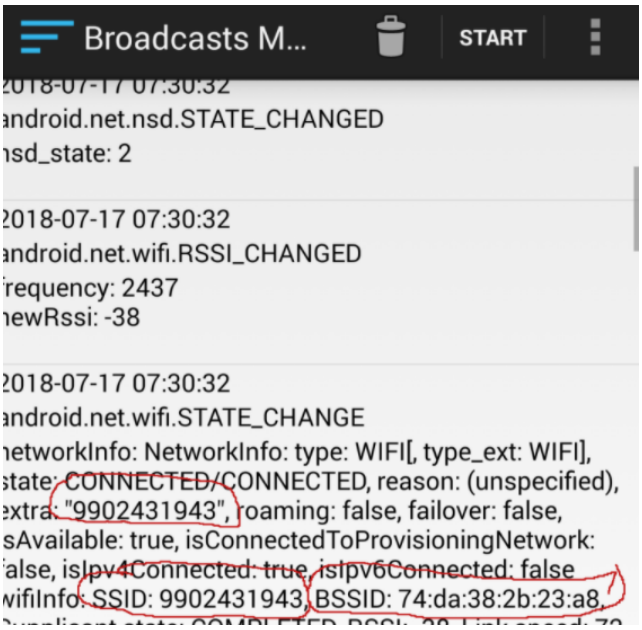
Part – 3

Sensitive Data Exposure:

System broadcasts by Android OS expose information about the user's device to all applications running on the device. This includes the WIFI network name, BSSID, local IP addresses, DNS server information and the MAC address. Some of this information (MAC address) is no longer available via APIs on Android 6 and higher, and extra permissions are normally required to access the rest of this information. However, by listening to these broadcasts, any application on the device can capture this information thus bypassing any permission checks and existing mitigations.

Because MAC addresses do not change and are tied to hardware, this can be used to uniquely identify and track any Android device even when MAC address randomization is used. The network name and BSSID can be used to geolocate users via a look up against a database of BSSID such as WiGLE or SkyHook. Other networking information can be used by rogue apps to further explore and attack the local WIFI network.

Reproduce with a application:



References:

https://securityboulevard.com/2018/08/mobile-misery-android-ios-data-leakage/

https://www.bleepingcomputer.com/news/security/unsophisticated-android-spyware-monitors-device-sensors/

https://macviruscom.wordpress.com/2018/08/31/mobile-misery-android-ios-data-leakage/