Assignment 3 - A

SQL Injection

Atharva Vaidya
121942024

Contents

What is difference between DBMS and RDBMS?

Parameter	DBMS	RDBMS
Storage	DBMS Stores data as a file.	Data is stored in the form of tables.
Database structure	DBMS system, stores data in either a navigational or hierarchical form.	RDBMS uses a tabular structure where the headers are the column names, and the rows contain corresponding values
Number of Users	DBMS supports single user only.	It supports multiple users.
ACID	In a regular database, the data may not be stored following the ACID model. This can develop inconsistencies in the database.	Relational databases are harder to construct, but they are consistent and well structured. They obey ACID (Atomicity, Consistency, Isolation, Durability).
Type of program	It is the program for managing the databases on the computer networks and the system hard disks.	It is the database systems which are used for maintaining the relationships among the tables.
Integrity constraints	DBMS does not support the integrity constants. The integrity constants are not imposed at the file level.	RDBMS supports the integrity constraints at the schema level. Values beyond a defined range cannot be stored into the particular RDMS column.
Normalization	DBMS does not support Normalization	RDBMS can be Normalized.
Distributed Databases	DBMS does not support distributed database.	RBMS offers support for distributed databases.
Ideally suited for	DBMS system mainly deals with small quantity of data.	RDMS is designed to handle a large amount of data.
Data Redundancy	Data redundancy is common in this model.	Keys and indexes do not allow Data redundancy.
Examples	Examples of DBMS are a file system, XML, Windows Registry, etc.	Example of RDBMS is MySQL, Oracle, SQL Server, etc.

Before breaking in to the site for access, an attacker needs to know which SQL engine is at the back end, list out what are the ways to know which SQL server is on work?

If your site has web page extensions like .asp, .aspx etc. then there is a high chance of SQL Server or MS Access.

If the pages end as .jsp it could be an Oracle system at your end.

If the pages are ending as .php, it will probably be MySQL.

To identify the back end SQL server the minimum qualification required is beginner level experience with SQL. The general tool used to find out the SQL engine at the back end are the string concatenating characters and the comment characters.
Using the character used to end a SQL statement is also a good candidate for finding out the SQL engine at the backend.
We can not stop this kind of attacks because even showing an error page back to an attacker is like letting him know that the entered string was wrong. However the golden thumb of rule is in the error page don't show the full detailed error message back, instead just say some error has occurred, and you can thus be saved from further damage the attacker is going to cause. Because, SQL injection attacks generally use the returned error messages to go a bit deep into the attacking process.

What are some examples of web applications that connect to a database server to access data?

- Pandora
- Hulu
- Meebo
- Cooliris
- Facebook
- Youconvertit
- Peepel
- Jott
- Netvibes
- TiddlyWiki

What is the impact of SQL injection vulnerabilities?

With no mitigating controls, SQL injection can leave the application at a high-risk of compromise resulting in an impact to the confidentiality, and integrity of data as well as authentication and authorization aspects of the application.
An adversary can steal sensitive information stored in databases used by vulnerable programs or applications such as user credentials, trade secrets, or transaction records.
SQL injection vulnerabilities should never be left open; they must be fixed in all circumstances. If the authentication or authorization aspects of an application is affected an attacker may be able login as any other user, such as an administrator which elevates their privileges.

What is the main reason for the existence of SQL injection vulnerability?

- **Design of SQL** : It is designed to allow people to access information and is therefore vulnerable, so every developer should know about SQL Injection.

- **Every single Input Counts** : A single input validation, if remained, may lead to loss of data.

- **Inexperienced Developers** : New Developers lack proper training for designing secure Applications.

- **Priority** : Organizations do not give priority for securing such vulnerabilities.

- **Time** : A lot of time is needed to search vulnerabilities like SQL Injection. Most organizations do not invest in such required amount of time.

- **Money** : A lot of capital is needed to invest in Pen-Testers for finding out vulnerabilities.

SQL Injection Demo using WebGoat

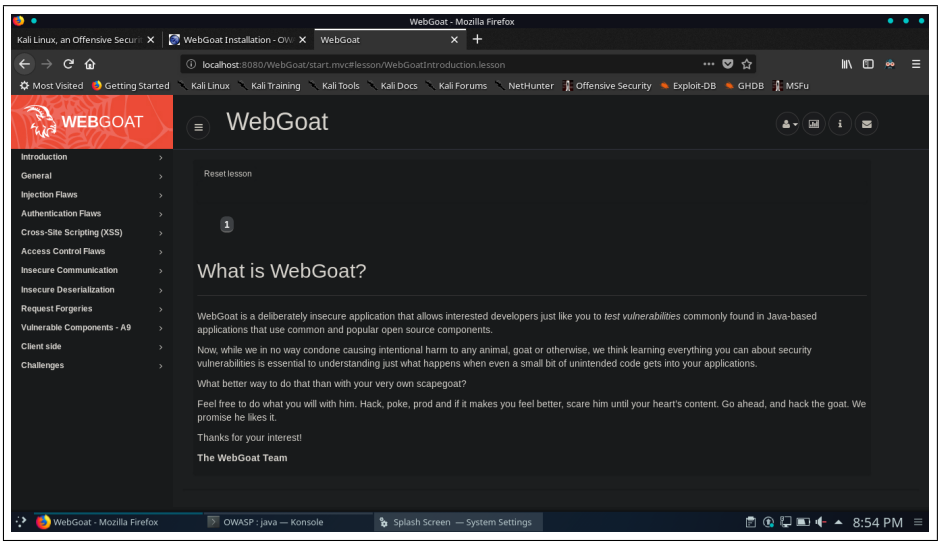

Figure 1: Introduction to WebGoat

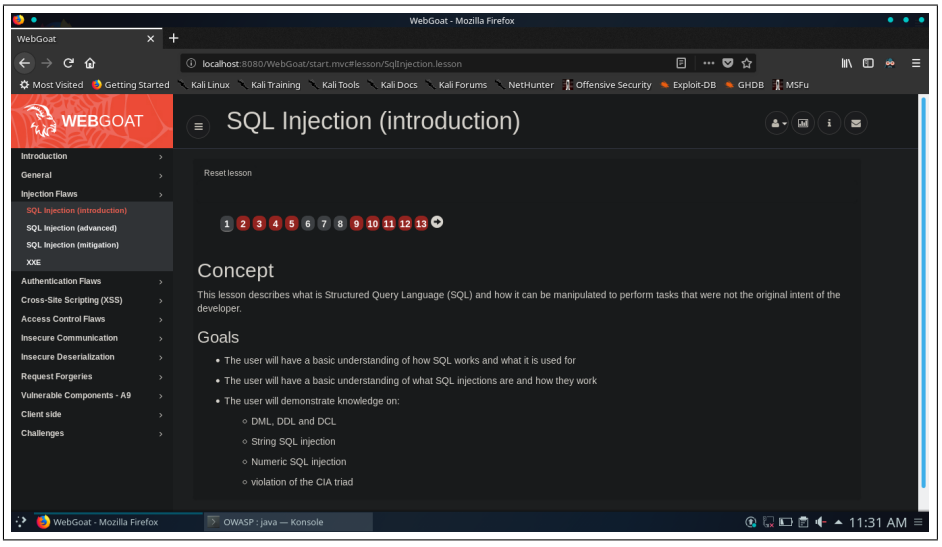

Figure 2: SQL Injection Introduction, Concepts and Goals

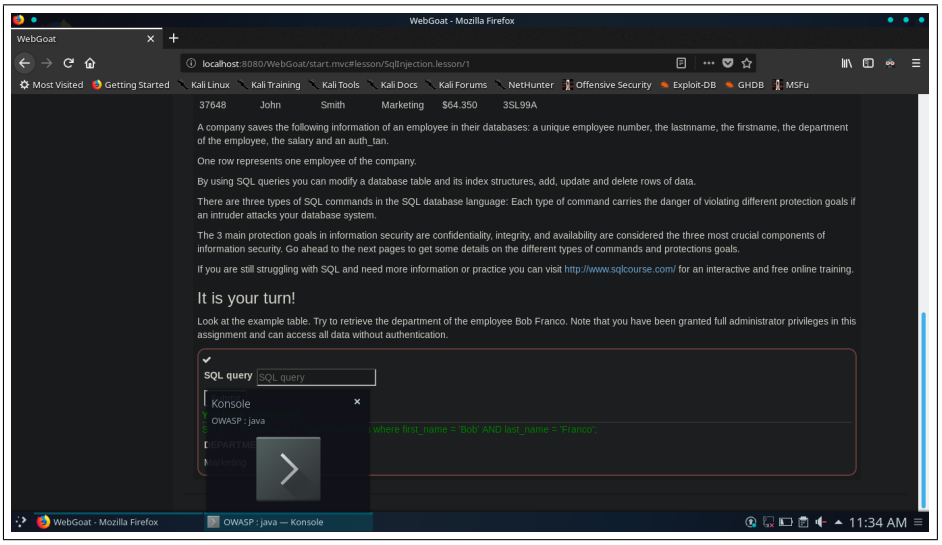

Figure 3: Basic SQL Select Query

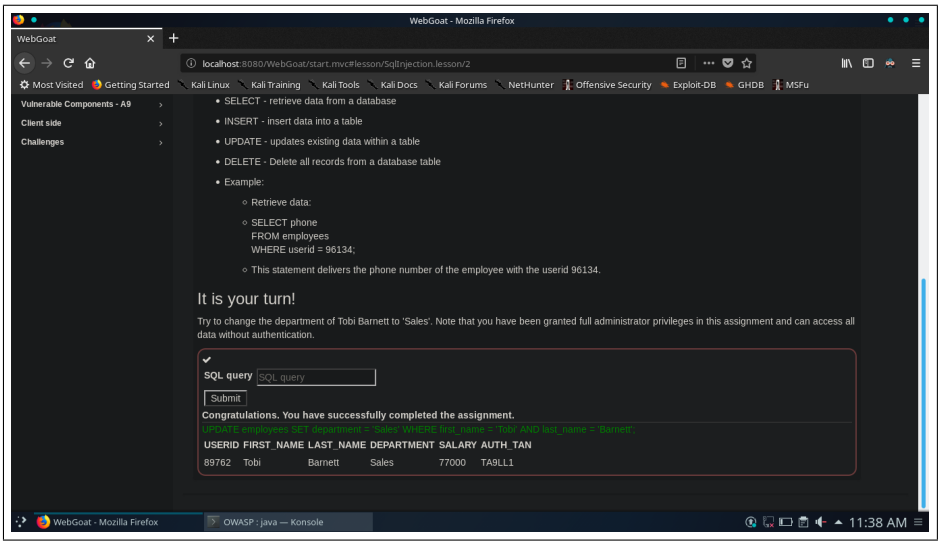

Figure 4: Basic SQL Update Query

Figure 5: Basic DDL Query

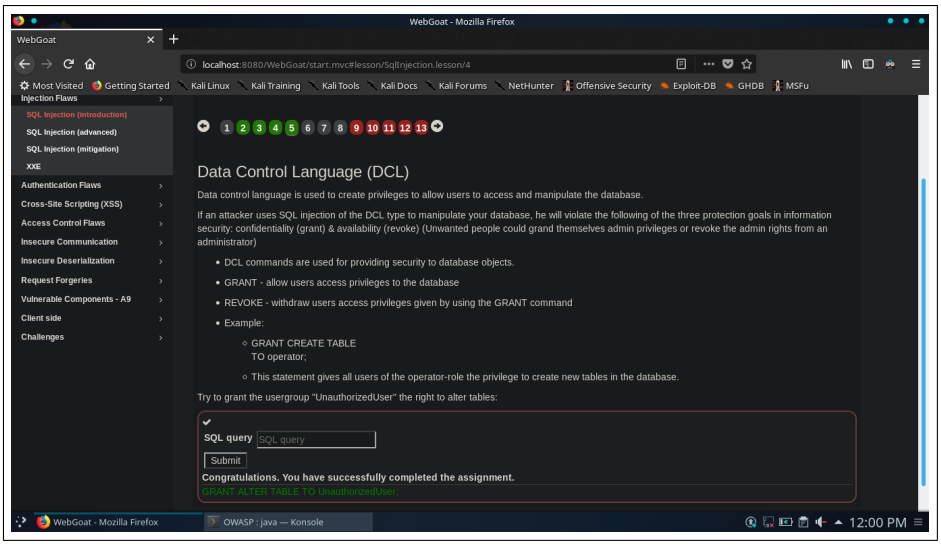

Figure 6: Basic DCL Query

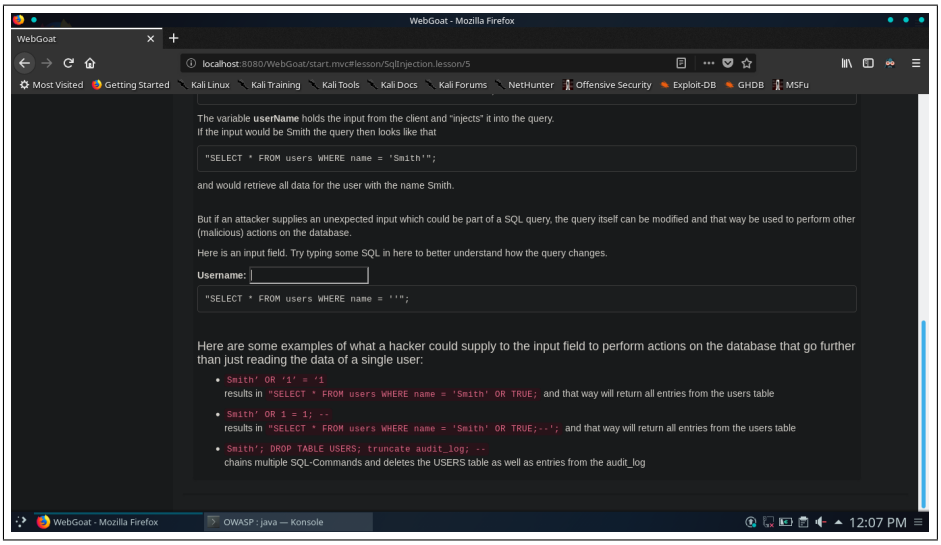

Figure 7: Some Examples of SQL Injection Queries

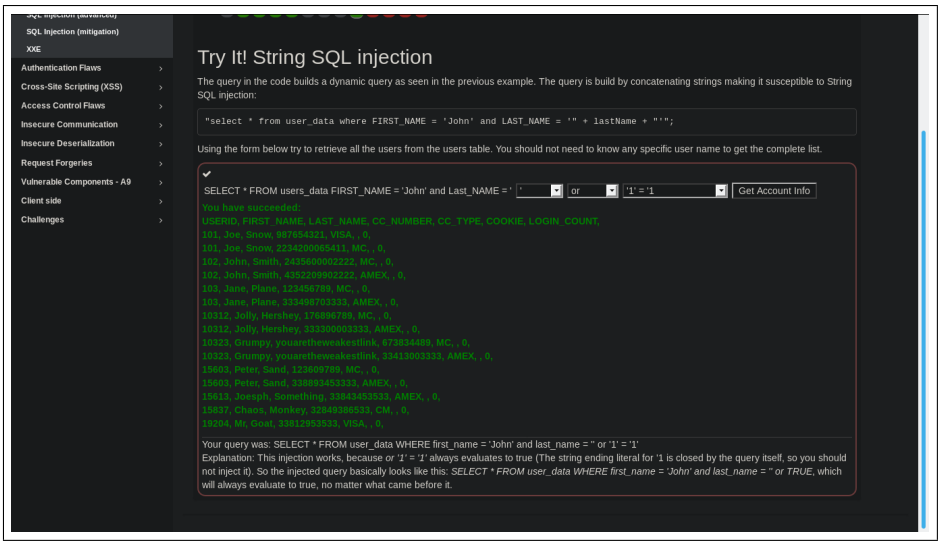

Figure 8: Successful Execution of basic SQL Injection Select Query

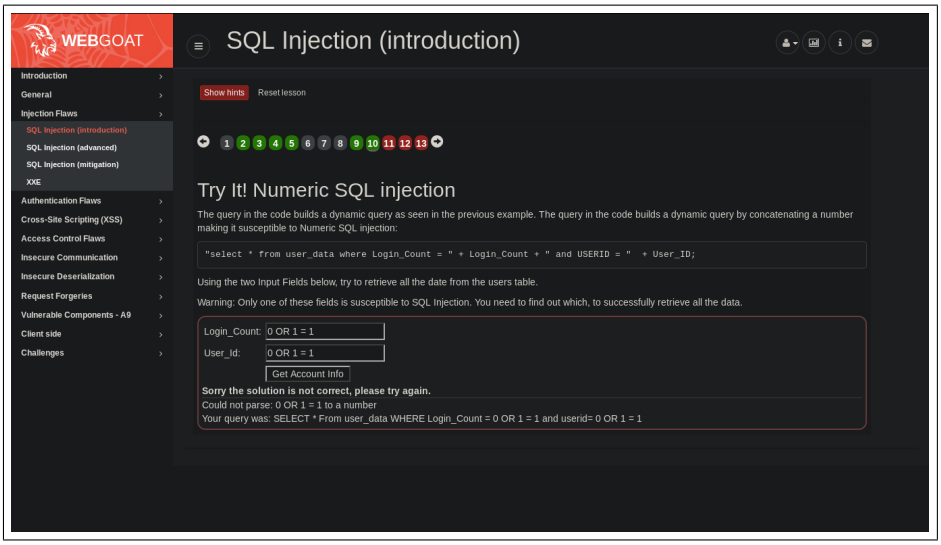

Figure 9: The Login-Count can only accept Numeric Values.

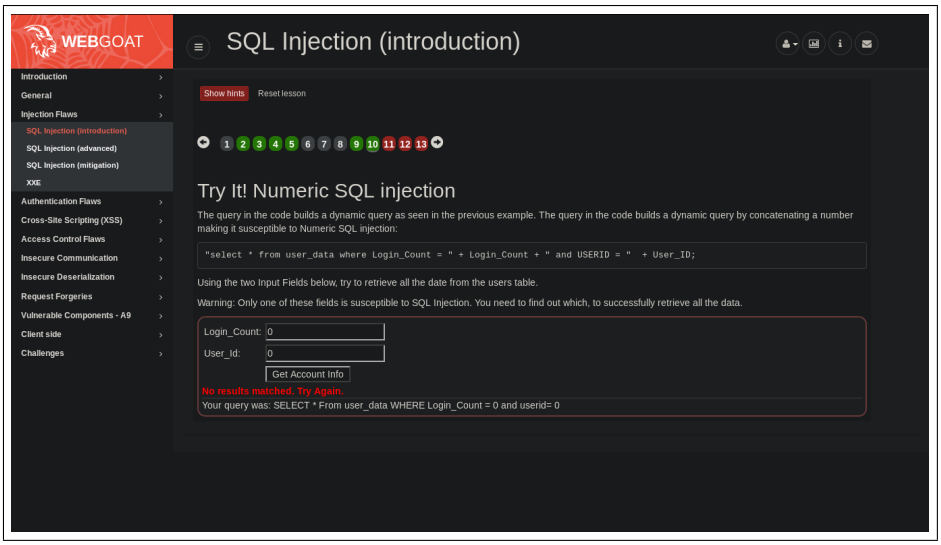

Figure 10: Checking the error message.

Figure 11: Identified the vulnerable field and applied the appropriate SQL Injection Query.

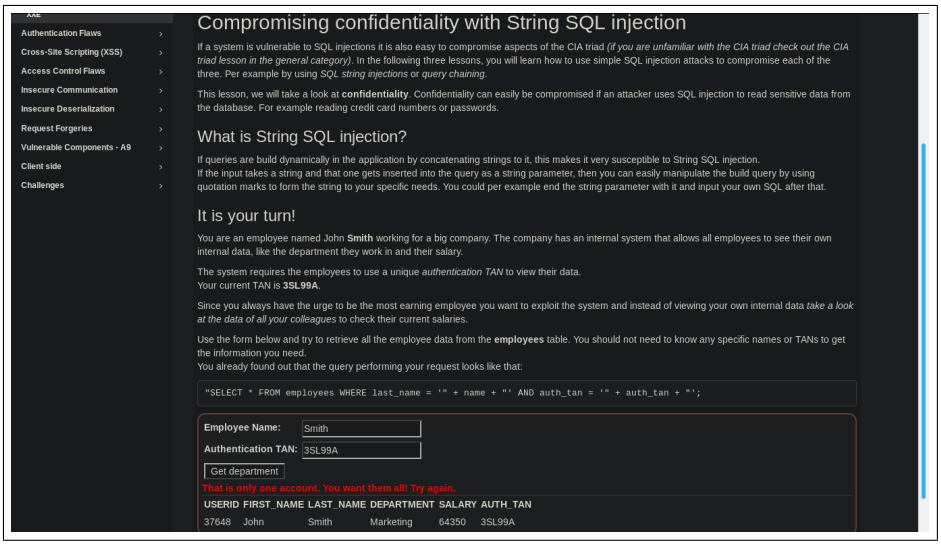

Figure 12: Applied the Query with known values.

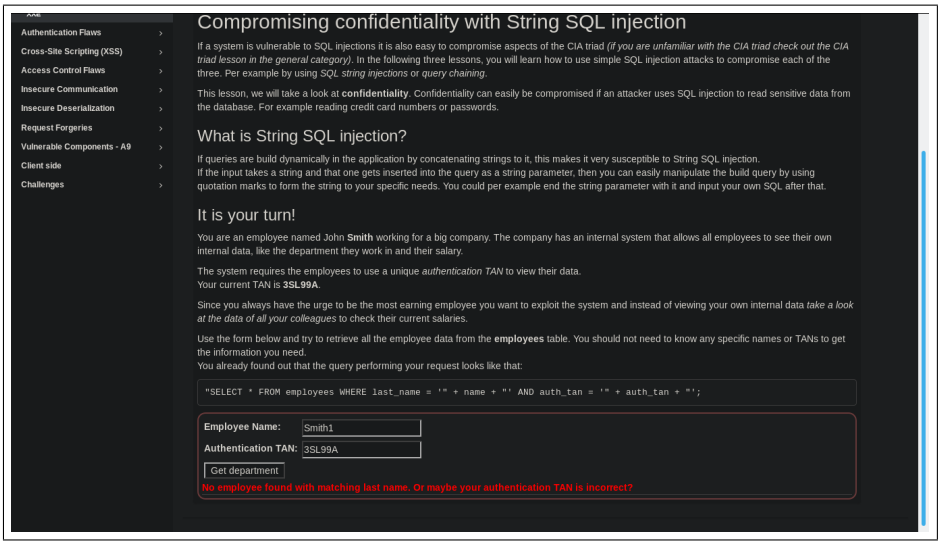

Figure 13: Checking the error message by feeding wrong usename and password.

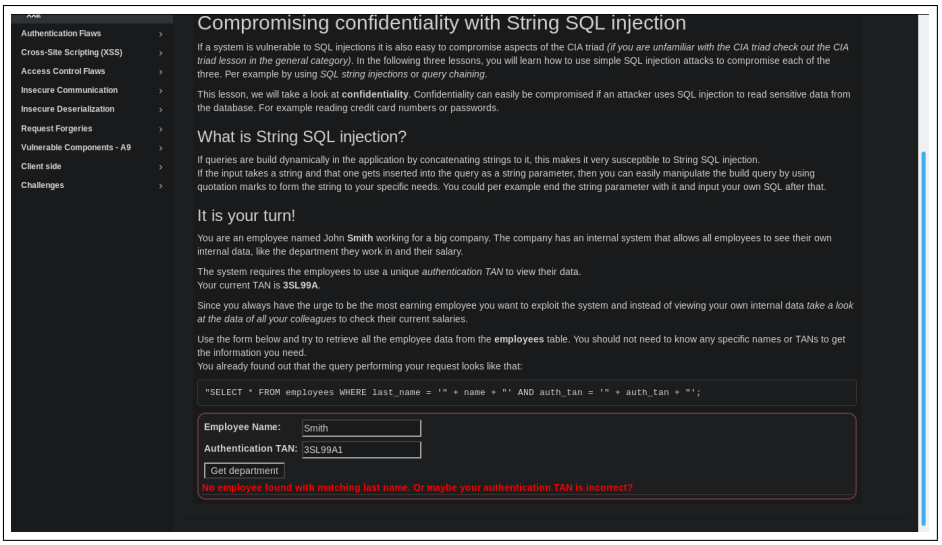

Figure 14: Checking the error message by feeding wrong password only.

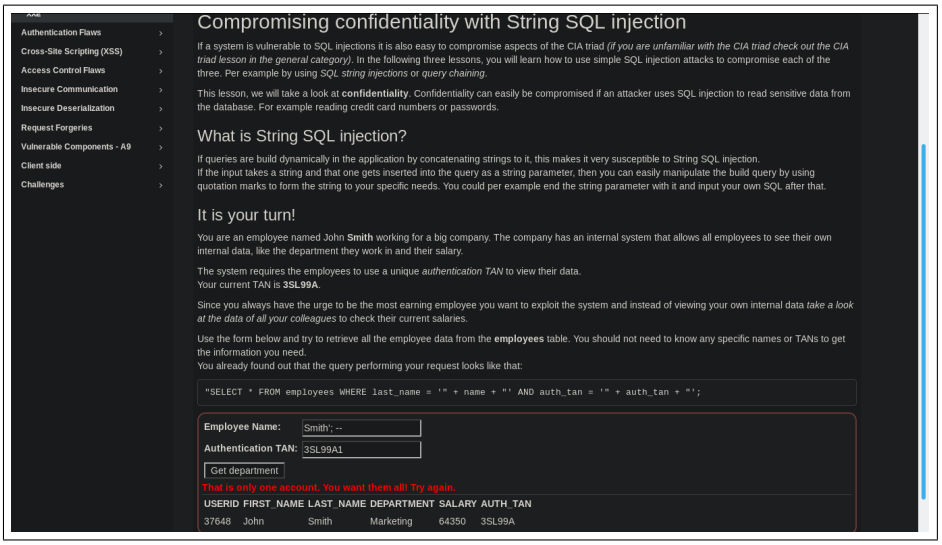

Figure 15: Trying SQLi by commenting the password field.

SELECT Query

Here the Query can be implemented in 2 Ways:

- Username - Smith
  Password - ' OR '1'='1
- Username - Smith'; SELECT * FROM employees; –
  Password - BLANK

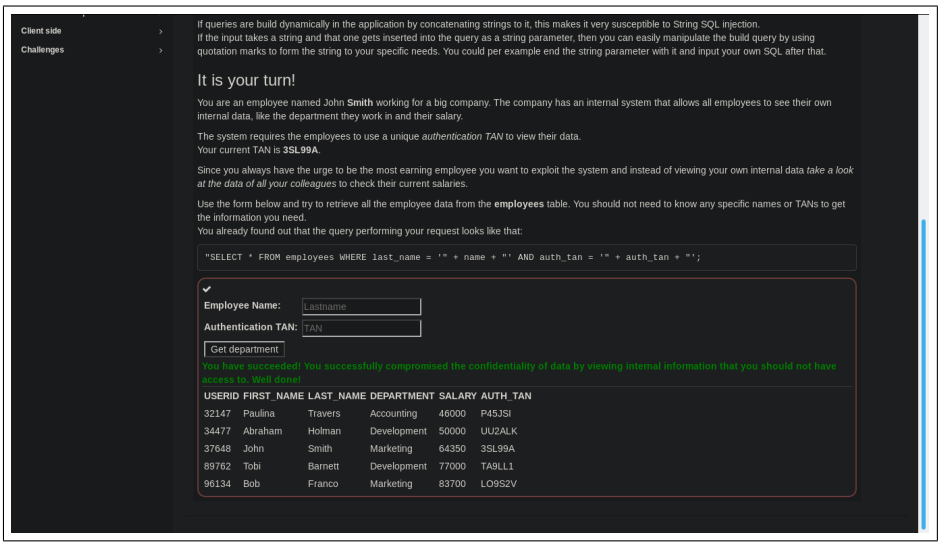

Figure 16: Successfully accessed the whole table

UPDATE Query

Employee Name - Smith'; UPDATE employees SET salary = '90000' WHERE last-name = 'Smith'; – –
Auth-TAN - BLANK

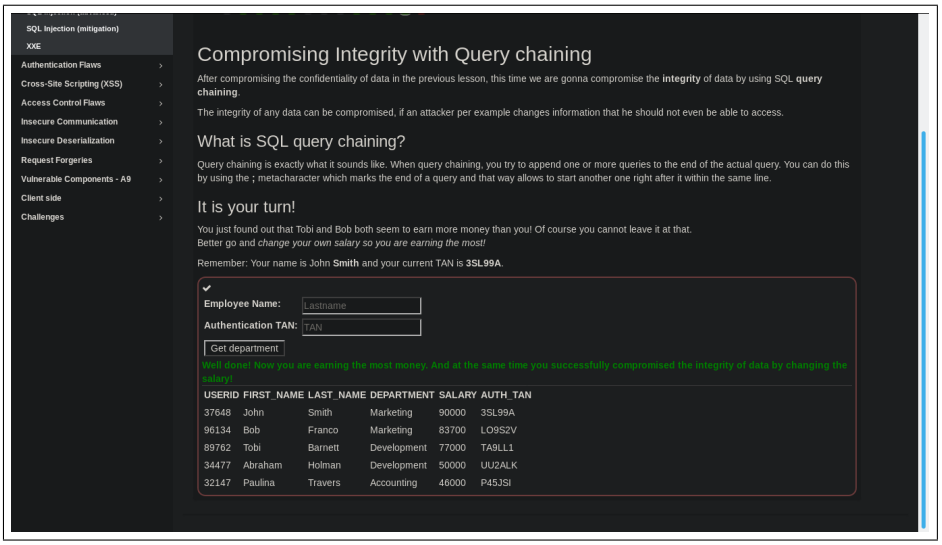

Figure 17: Successfully compromised the Integrity of the Table by chaining Update Query.

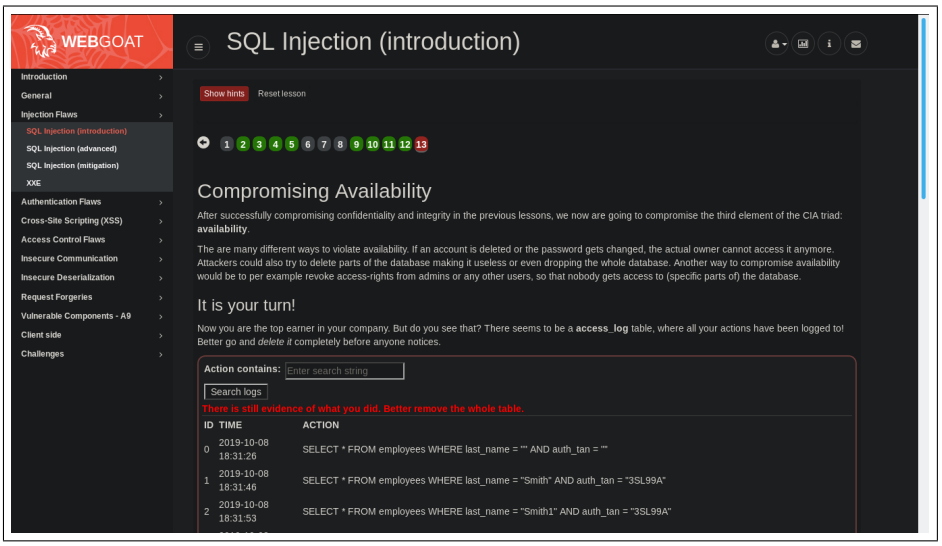

Figure 18: Searching with Empty field.

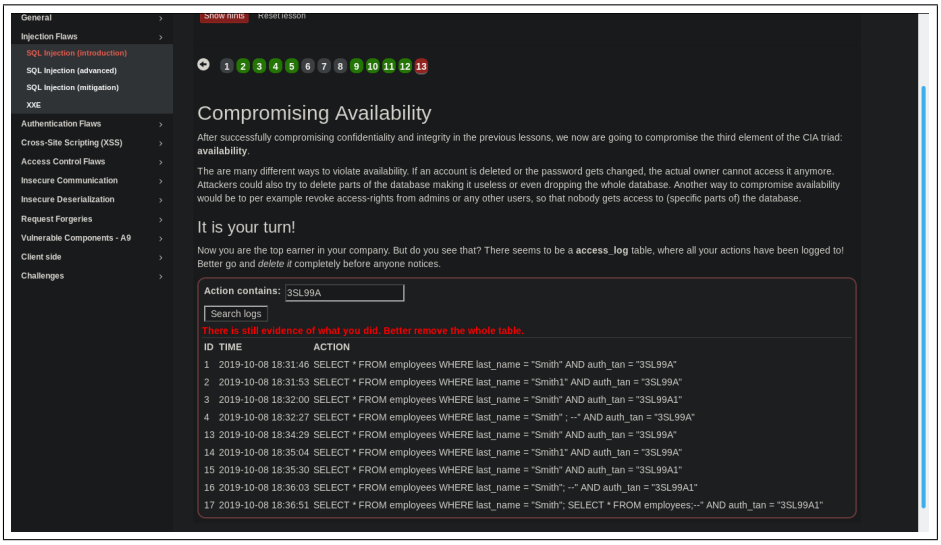

Figure 19: Searching the table with the Specified value.

DROP Query

'; DROP TABLE access-log; –

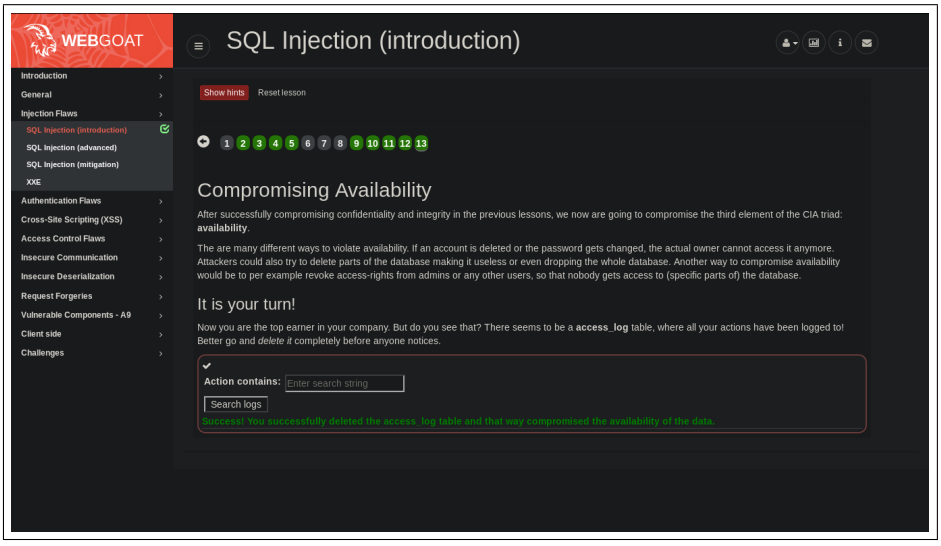

Figure 20: Successfully deleted the access-log table by chaining DROP Query.

Figure 21: Advanced SQL Injection

Figure 22: SELECT Query using Chaining.

Figure 23: Again applying SELECT Query using Chaining on another Table.

Figure 24: We have successfully found out the required password.

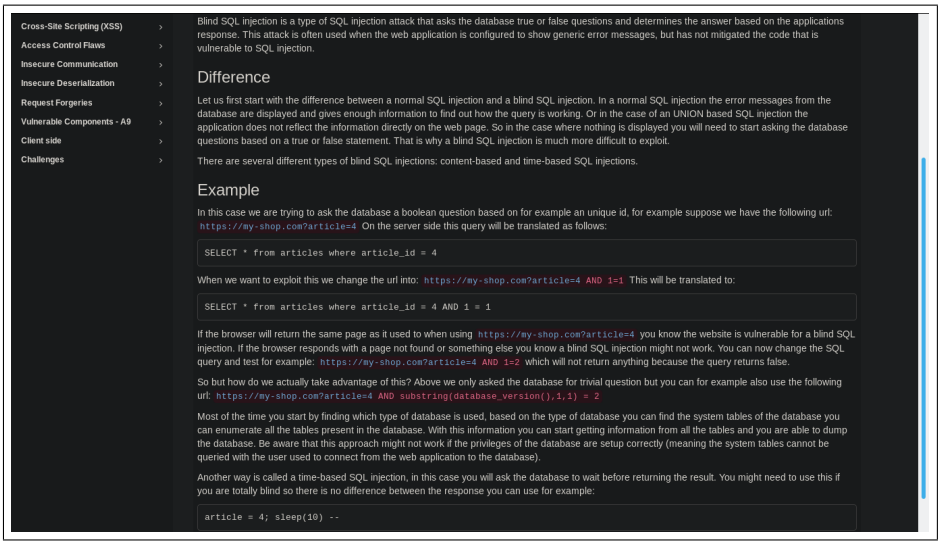

Figure 25: Blind SQL Injection Introduction.

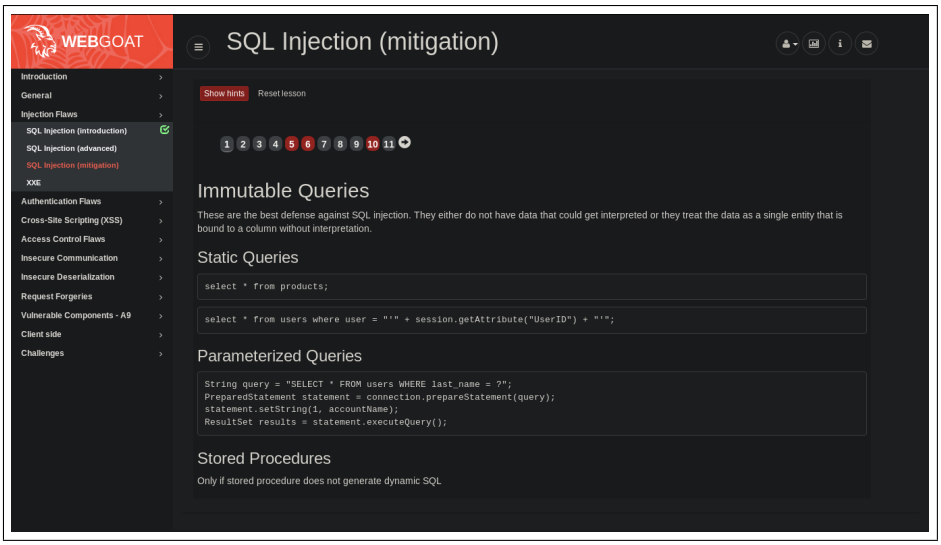

Figure 26: Some SQL Injection Mitigation Practices.