For finding flag 3 we saw that port 3306 with sql was open.
We first tried to login with mysql 10.162.2.83 root@localhost but this did not work.

We then found the tool sqlmap and played around a bit with that. We tried to do some simple tests on the login page and signup page but found no result. Then we found that we could specify a cookie so we could test forms when you are logged in as needed. We tried that on the search and post but that did not work either. Sqlmap was not very effective here in the beginning as it found nothing.

We then assumed the webpage was vulnerable on the same page in which we found flag 2 since flag 3 relied on flag 2 being completed so we assumed it was found on the recherche_old.php page. This flag was also targeted towards database hacking and we then assumed it had to do with SQLi(sql injection) on the form which we found there to search for users.

Then we used some basic sql injections, and when we tried ‘ or ‘1’=’1– we got an error message saying we found the matrix. After that we knew that the form was vulnerable to sql injections because of the error message from the page where we could see the entire sql query and what went wrong with executing it.

After that we tried a bunch of combinations, but the most we were able to do was dump all the users with only the username, mentions, and such. I.e instead of listing one user when searching we could list all users. This happened when we did the ‘ or ‘1’=’1– and it works because we have a query which looks like this:
SELECT DISTINCT usID, usPseudo, usNom , usAvecPhoto FROM users WHERE usNom LIKE '%’
When we then inject the sqli into the ‘%’ it instead looks like this
SELECT DISTINCT usID, usPseudo, usNom , usAvecPhoto FROM users WHERE usNom LIKE '%’ or ‘1’ =’1%’--
With this injected we will get all usID, usPseudo, usNom, usAvecPhoto since the ‘1’=’1’ will always be true and since we have an or operator there the query will get all entries in the table. The – in sql specifies a comment which means we comment out the sql which is after it.

We then tried to understand deeper what sqlmap could be used for and explored what different flags could be used. We found that we could test the form when logged in by specifying –cookie=“PHPSESSID=<our token>” which is used for specifying the token and the account you are logged in as.
After a while we found the –dump-all flag which seemed to dump most of the database's content, but this took a while. After a long while we found that it had enumerated the databases and found couteurs database which contained the flag table which we now had a csv for. When we opened that file we found the flag!

Dump the database:
sqlmap -u http://10.162.2.83/php/recherche_old.php -dbs --cookie="PHPSESSID=evfehqjr07kart84mdotpdh1e0" --dump-all --forms

Filter the files with the command:

find /home/kali/.local/share/sqlmap/output/10.162.2.83/dump/ | xargs grep flag
or just go into /home/kali/.local/share/sqlmap/output/10.162.2.83/dump/ and see that we have the cuiteur database and that it contains a flags table.