Hacking Log Substitution Week 1

This hacking log wasn’t submitted because I had not actually completed any flags up to the end of week one, and therefore forgot I had to submit a hacking log. Because of that, I will talk about an exploit I made a few weeks later. This exploit was the SQL Injection into the Cuiteur old research page.

For an SQL injection to take place, first and foremost an injectable field must be found. Since I had some experience with SQL injections in the past, but no experience with SQL map, it took me a while before I realized how powerful the tool could be and I did this research, and caught the subsequent flag, manually. Normally, to manually test for an input that is vulnerable to an SQL injection, that requires manually inputting several things into each input in the web page. In this course, however, it was a little easier given that we had a flag at the source code of the page with the injectable input. Because of this, when I saw the flag I knew this would be the input I would have to inject. Another, less obvious, way to figure it out is the fact that the page is called “research old”. One is left wondering, why is it the old page? What’s wrong with it? Which may lead us to realize that it has a vulnerable input.

Once I found the input, the way to test if it was vulnerable to an SQL injection was to try a few different inputs. An SQL injection works the following way: let’s say there is a search input that looks for users with a certain username. When you search in this input, it sends an SQL query to the back-end of a form like:

SELECT * FROM users WHERE username=”**[username]**”

Where [username] is your input. If you input something like: **“ OR 1=1 – ddd**, what the server will receive in the back-end is:

SELECT * FROM users WHERE username=”“ OR 1=1 – ddd”.

This means that the server will interpret this as “Oh, we’re looking for everyone who either has a blank username, or where 1=1”. Of course, 1 always equals 1. This means that the server now returns the entirety of the users collection.

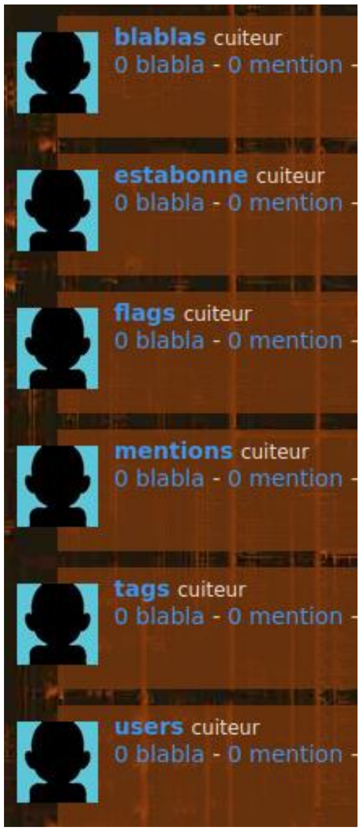
Of course this isn’t always so cut and dry. The back-end could, for example, use single quotes instead of double quotes, so you’d have to try multiple queries until you were absolutely certain that an input is safe. Normally SQL map does most of this hard work for you so you don’t have to worry much, but since again, I am stubborn and didn’t look early enough into it, I ended up typing most of it by hand. Thankfully, also, the error message given by simply outputting **‘ OR 1=1** told me all I needed to know for my SQL injection. My process to capture the flag went as follows:

1. Attempted to run other queries after the SELECT one and figured out we couldn’t run multiple queries.
2. Found out about the UNION keyword through the SQL cheat sheet and tried to use built in system functions to no avail
3. Found out about the INFORMATION_SCHEMA database which gave me access to all the available tables and columns
4. Found the “flags” table
5. Printed out all the columns in the flags table (once again using INFORMATION_SCHEMA)
6. Printed out the flag and found it :)

I am not going too much into the details of how I actually printed it out because they were unnecessarily complicated, but as an example, this is how I printed out all of the table names in the database:

```
ddd' UNION SELECT 0 AS usID, TABLE_NAME AS usPseudo, TABLE_SCHEMA AS usNom, 0 AS usAvecPhoto FROM INFORMATION_SCHEMA.TABLES; -- ddd
```

Essentially I kept 0 as the user id (would get errors otherwise), 0 as the profile picture (later figured out I didn't even need to specify AS usAvecPhoto, could have just had 0), and then I replaced the pseudonym of the user and their real name with the relevant data I was looking for, so I got it printed out nicely in the search results page like so:



```
blablas cuiteur
0 blabla - 0 mention -

estabonne cuiteur
0 blabla - 0 mention -

flags cuiteur
0 blabla - 0 mention -

mentions cuiteur
0 blabla - 0 mention -

tags cuiteur
0 blabla - 0 mention -

users cuiteur
0 blabla - 0 mention -
```

This is the process I ran for every search on the database.