ASSIGNMENT 3

Komal Joshi (014682551)

Ruchita Dinesh Entoliya (015331966)

Setting security level: We can set security level as shown in below image.

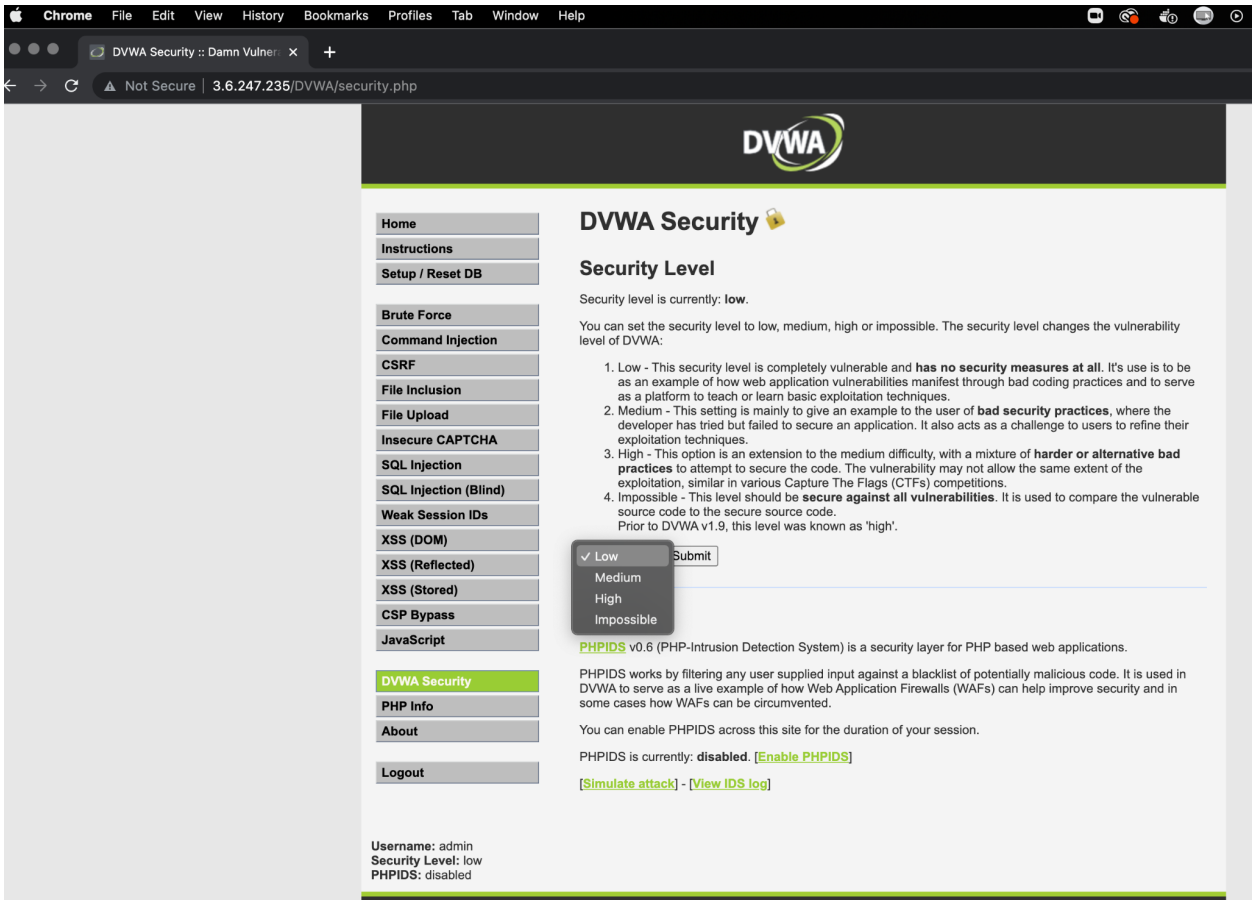


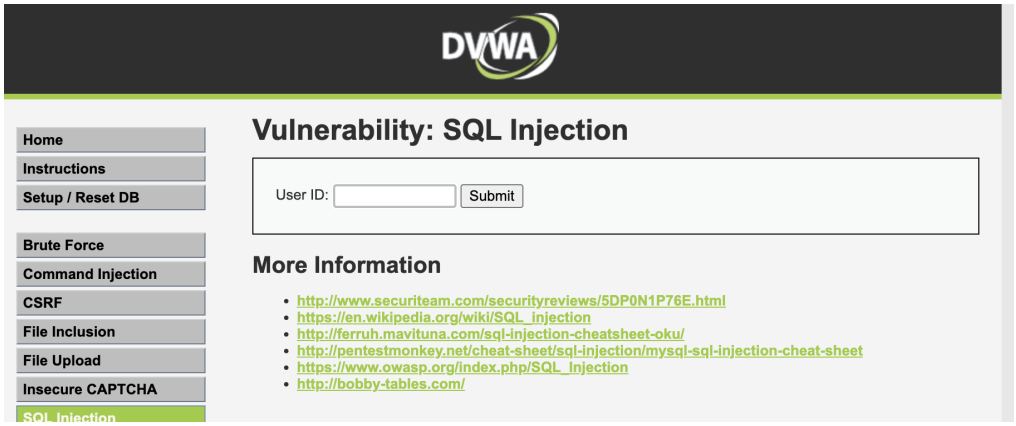
Fig 1. Security level

Que 1. Describe the SQLi attack you used, how did you cause the user table to be dumped? What was the input string you used?

Ans.

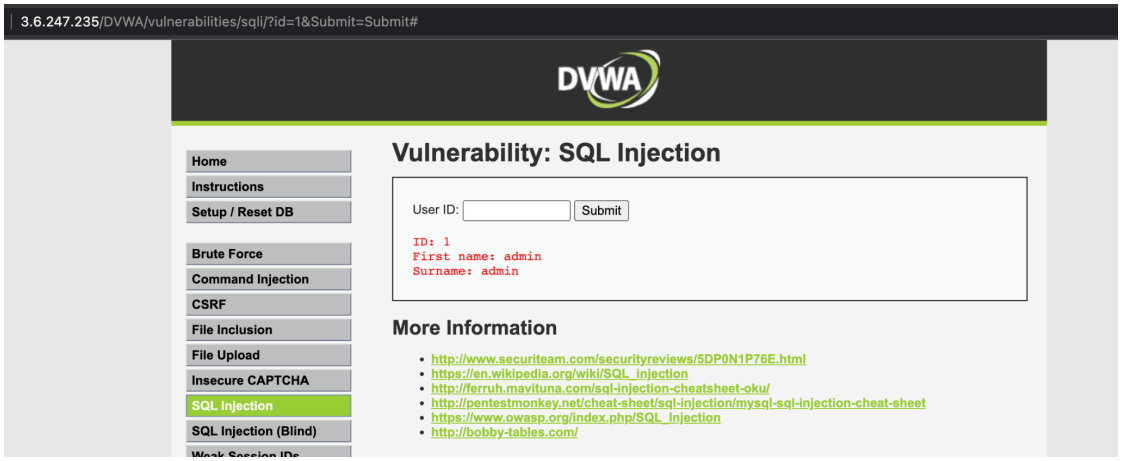
Step 1: Go to SQL Injection

As can be seen in Fig. 1, the security level is set to “low”. On the left panel menu options, we need to select SQL Injection. We can see following image:

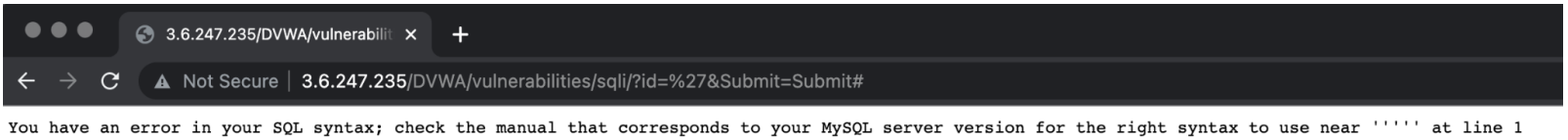


Step 2: Identify database

When I entered a user id, I could see details of the user correctly as follows for ID 1:



I was not getting any data when I entered an alphabet or string. This led me to the assumption that there is a “where” type of clause that accepts string as an id. I tried to enter ‘ as user id to mimic erroneous syntax and got following error:



Now, I know that the database is MySQL.

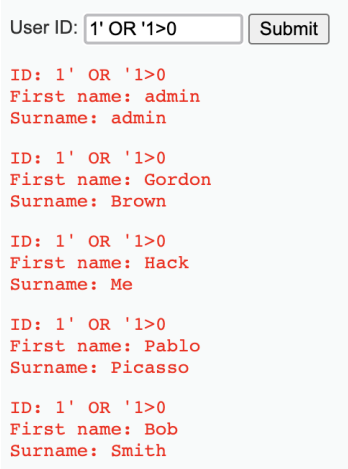
Step 3: Query format identification

From step 2, we are sure of the syntax of the SQL query to be somewhat as follows:
SELECT first_name, last_name FROM User WHERE ID=’$’

$ is any ID
I now entered the string as **1' OR '1>0**. Therefore, the select query will be like following:

SELECT first_name, last_name FROM User WHERE ID=’1' OR ‘1>0’

This results in fetching all the table details as follows:

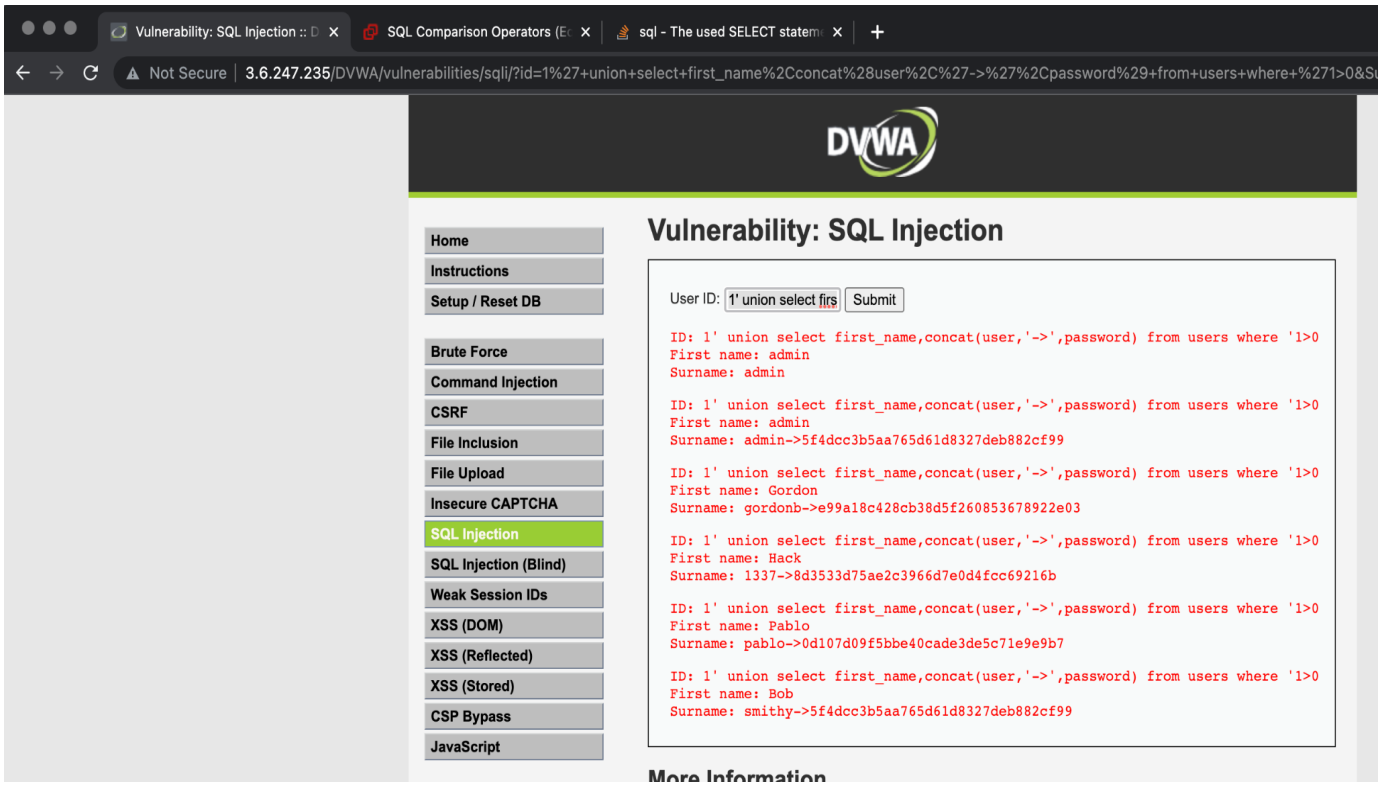


By hit and trial, we could determine that the field name for first name is first_name, last name is last_name and table name is users. From our findings up till now, we tried following text in the submit button and got the entire data dump:

1' union select first_name,concat(last_name,'->',user,'->',password) from users where '1>0

In the Surname field, we can see concatenated result containing surname, username and password separated by '->'.

We got the following table data with user critical details.

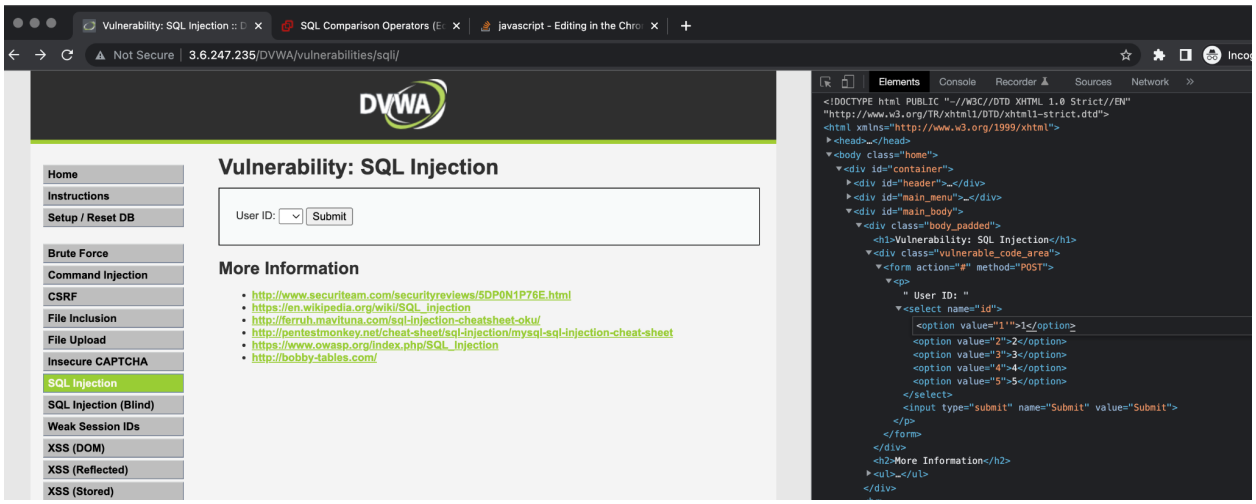


We reached a conclusion that low level of security permitted users to send any data from UI and simply appended to the query in the backend. This resulted in an extremely vulnerable app prone to sql injection attacks.

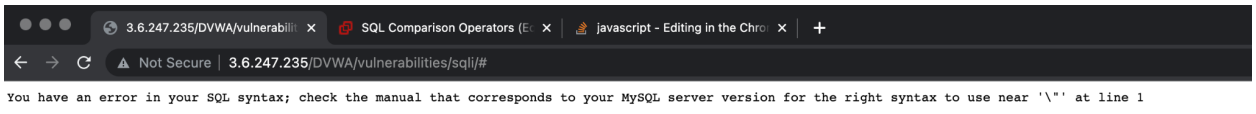
Que 2. If you switch the security level in DVWA to “Medium”, does the SQLi attack still work?

Ans. After changing the security level to “Medium”, I noticed that I was unable to perform SQLi attack. Since the first level of security is drop down instead of text box, it limits a user:

Then, we tried to update the HTML page and change <select> value to a simple **1”** as follows:



However, doing so resulted in the following error from the backend. This leads us to a conclusion that the query no longer used simple appending.

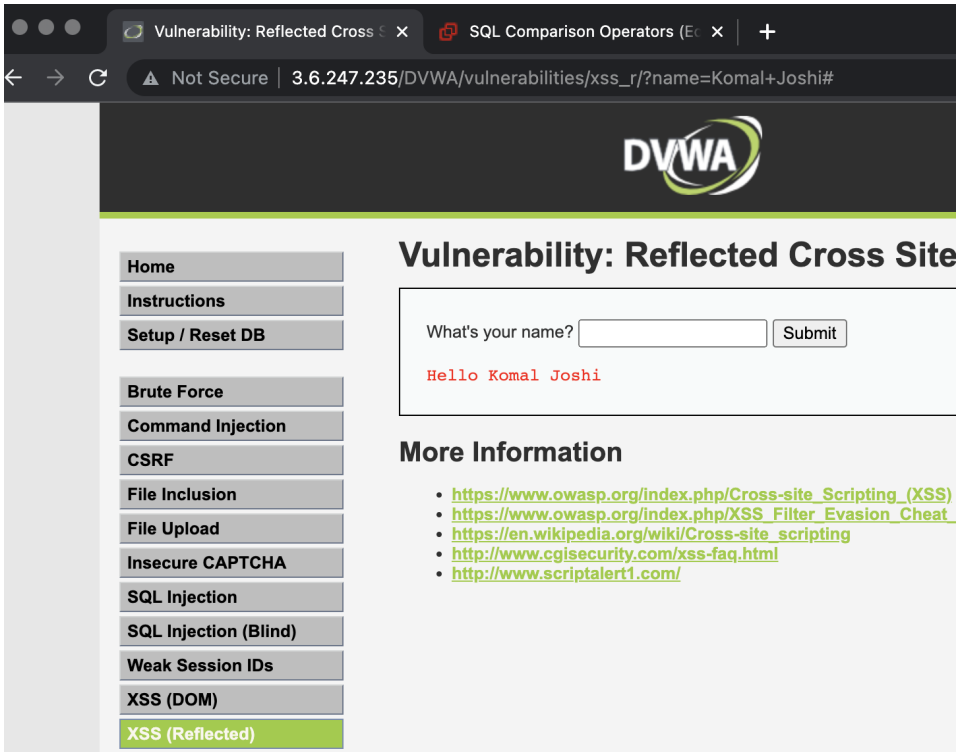


Que 3. Describe the reflected XSS attack you used, how did it work?

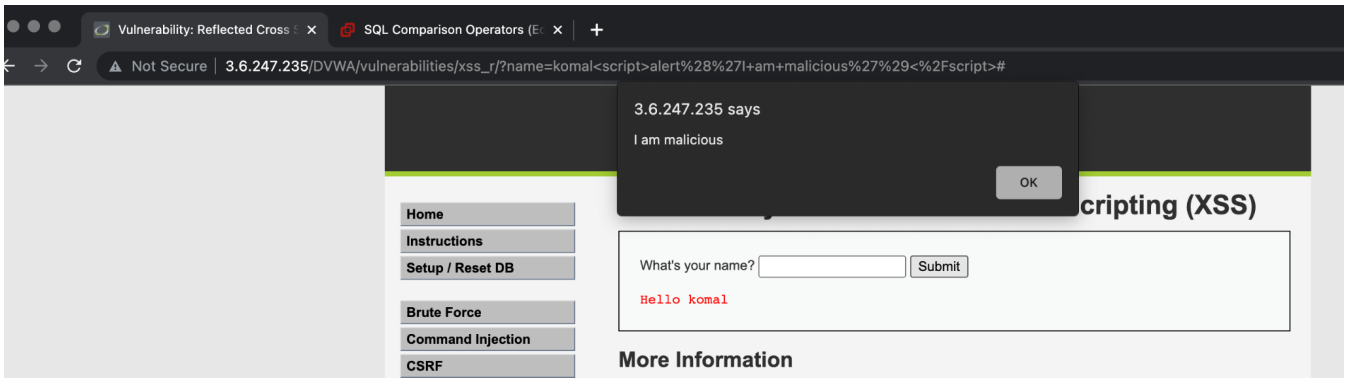
Ans.

Security Level: Low

In the below image, we get a Hello <user_input> message upon clicking on submit. <user_input> is the text data entered in the text area.



We can directly try using some text wrapped in <script> as the text data does not appear to be under any validation. Writing **komal<script>alert('I am malicious')</script>** gave us console output as follows:



Entering the following command in the text area, we can redirect to any malicious site, in the example we have redirected to stack overflow.
World <script>window.location.href='https://stackoverflow.com/'</script>

Similarly any XSS attacks can be performed.

Que 4. If you switch the security level in DVWA to "Medium", does the XSS attack still work?
Ans.
Security level: Medium

Upon adding same command as in Que 3., we can see in the right side of the below image that the script tag got removed. The updated security seems to omit the script tag to strengthen security by adding validations to data received from the text area.