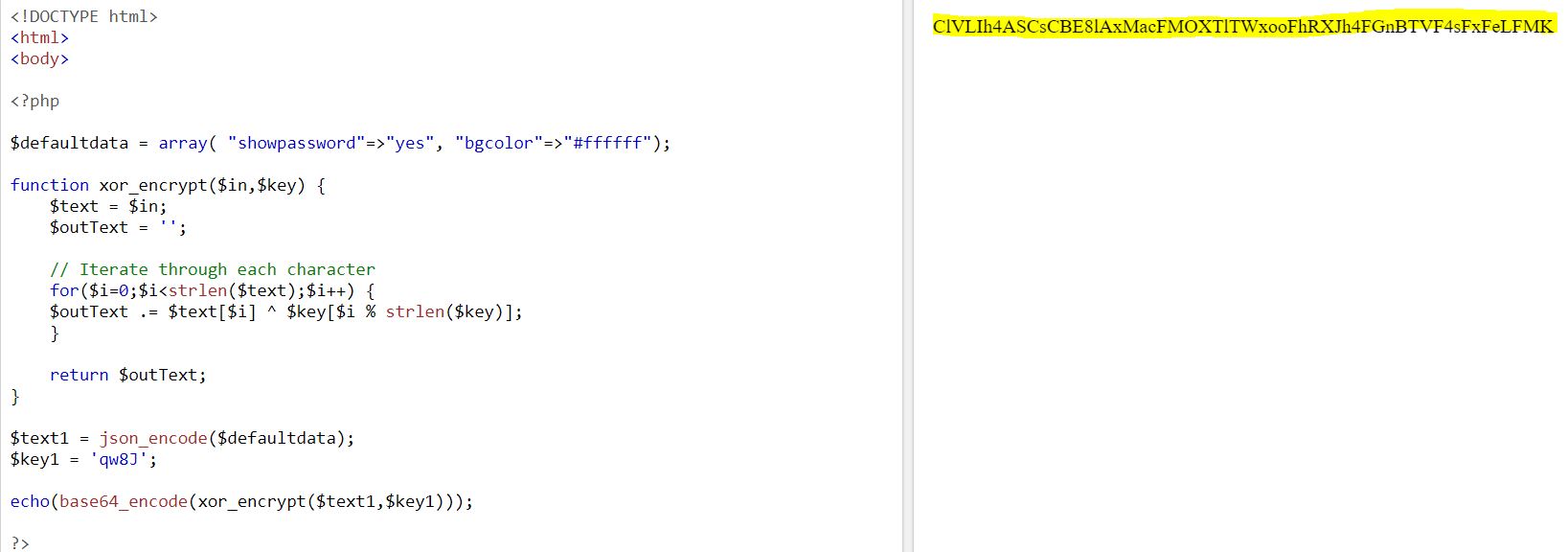
Continuing from Natas 12:

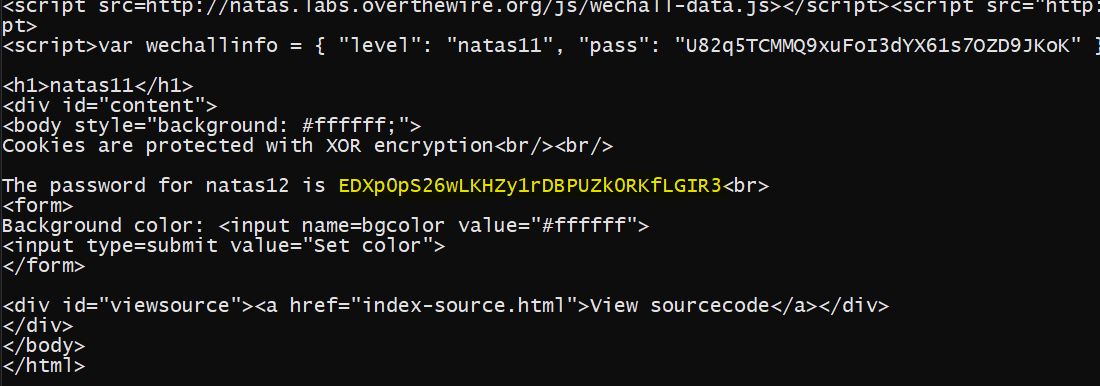
For Natas 12, this was a really tough one for me looking at the source code, it was clear that the cookies get encrypted multiple time first as the base64, then xor and then json\_decoding, so to get the password to the next level I need to set the show password as yes and then decode it to JSON then xor it and then base64 to send the cookie, so for finding the key of xor I used the property of xor which is A ^ B = C and A ^ C = B so if I xor the encrypted text with the plain JSON text then it will spill out the key, and doing that I got the key. and just encrypted the cookie of show password = yes and got the pass for the next level.



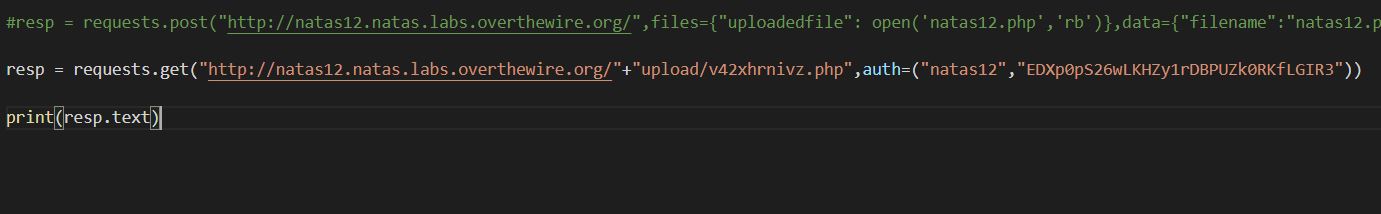
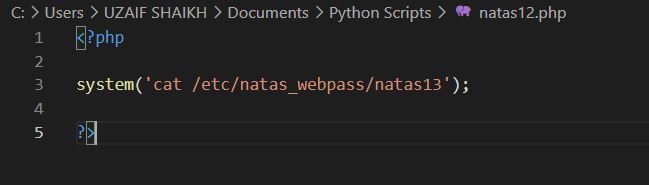
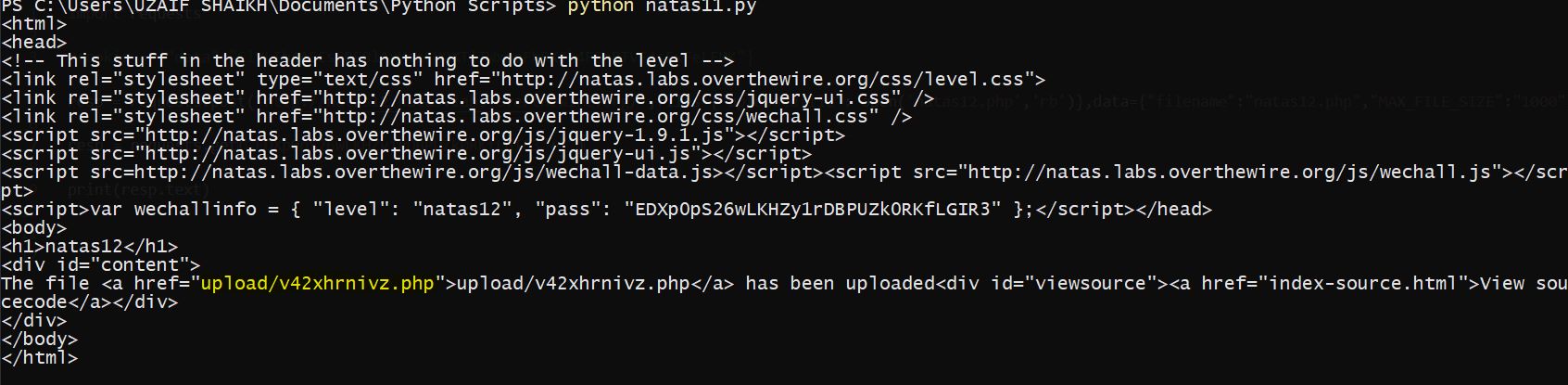




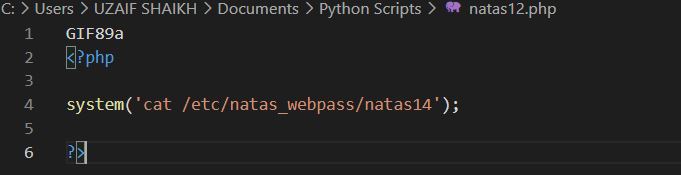
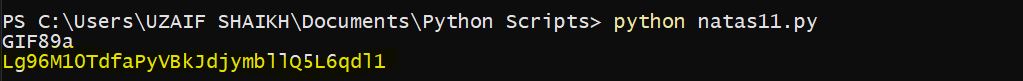




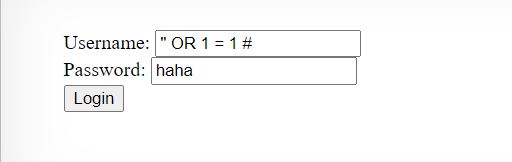
For this level, I had to upload an image datatype Jpg so using the data arg of the post function from the requests library I change the filename to a PHP and then uploaded the PHP file which had the command for cat /etc/natas\_webpass/natas13 and after requesting the get for that uploaded file I got the password for the next level.



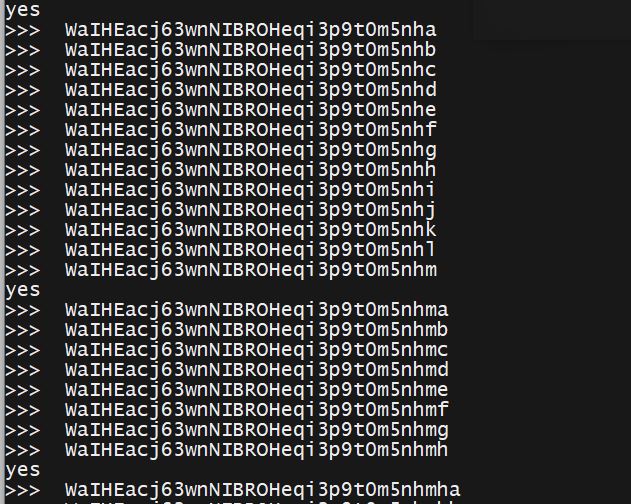
For this level, It was pretty similar to the previous one, here in the PHP source code the uploaded file is being checked for the signature of the file so for the PHP I just had to change the signature to an image. and running the same code as before I got the password for the next level.

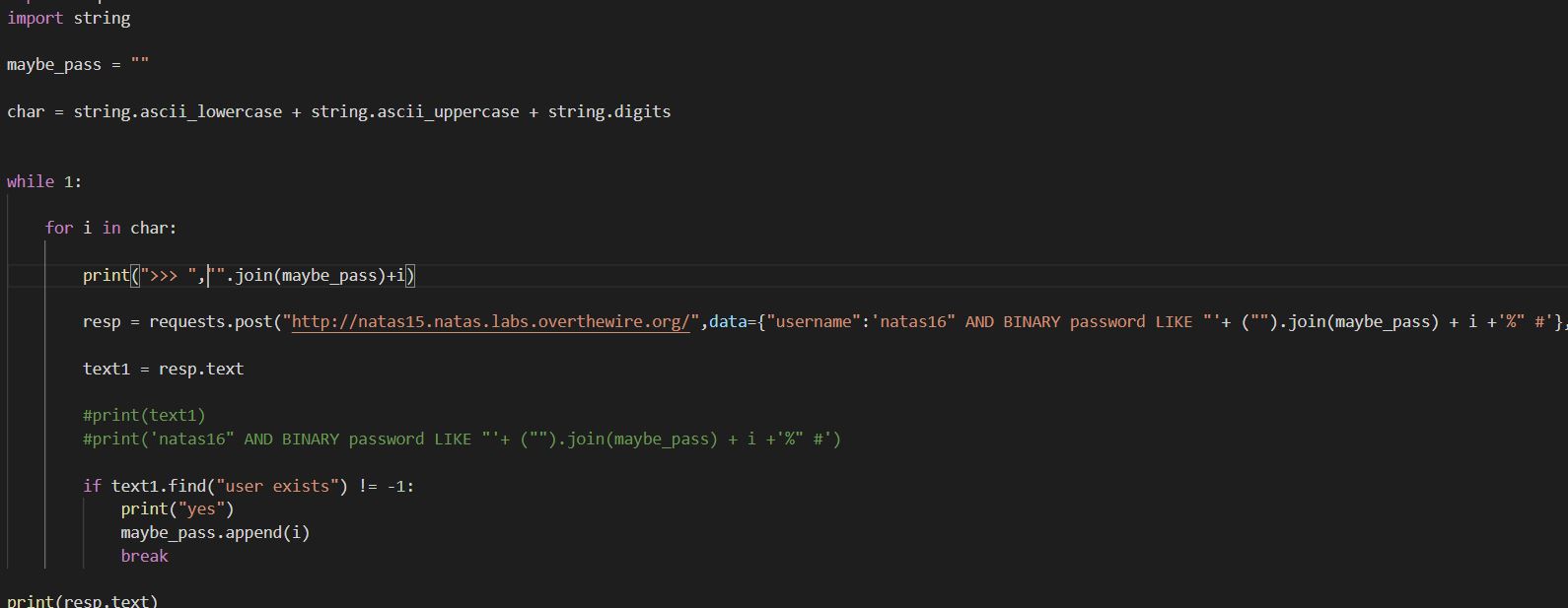


For Level 15, Looking at the source code tells that, the user input is directly used in the SQL query which makes this open to SQL injection so we can make the query valid by putting in " OR 1=1 # the # will comment out everything in the rest of the query and it will always be true and it will give out the password of the next level.

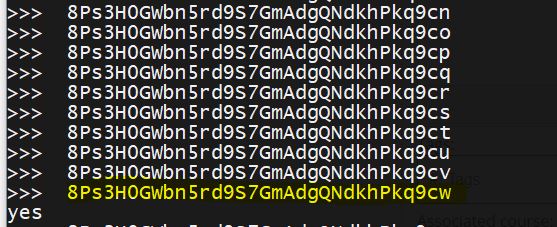


For Level 16, I had to do brute force to get the password, in the input first I checked if the username natas16 exist and it does, so I check for the password using the " AND Password LIKE  command of SQL which like give user exist if the password matches a string. so for this, I wrote a simple python script which just while infinitely till the character matches the password it does then check for the next string in the combination of the char which is lowercase\_asiic + uppercase\_asiic+digits.

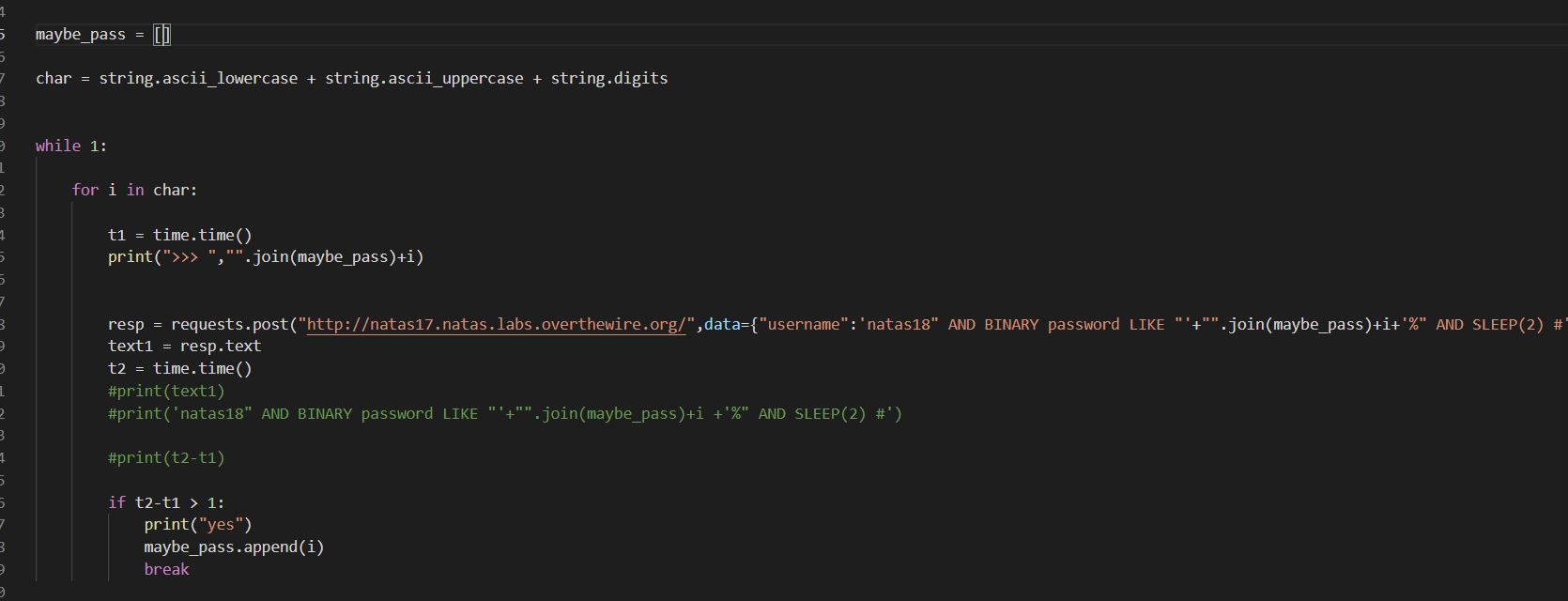
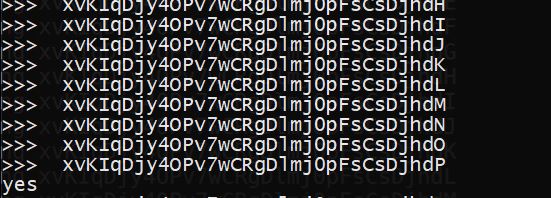




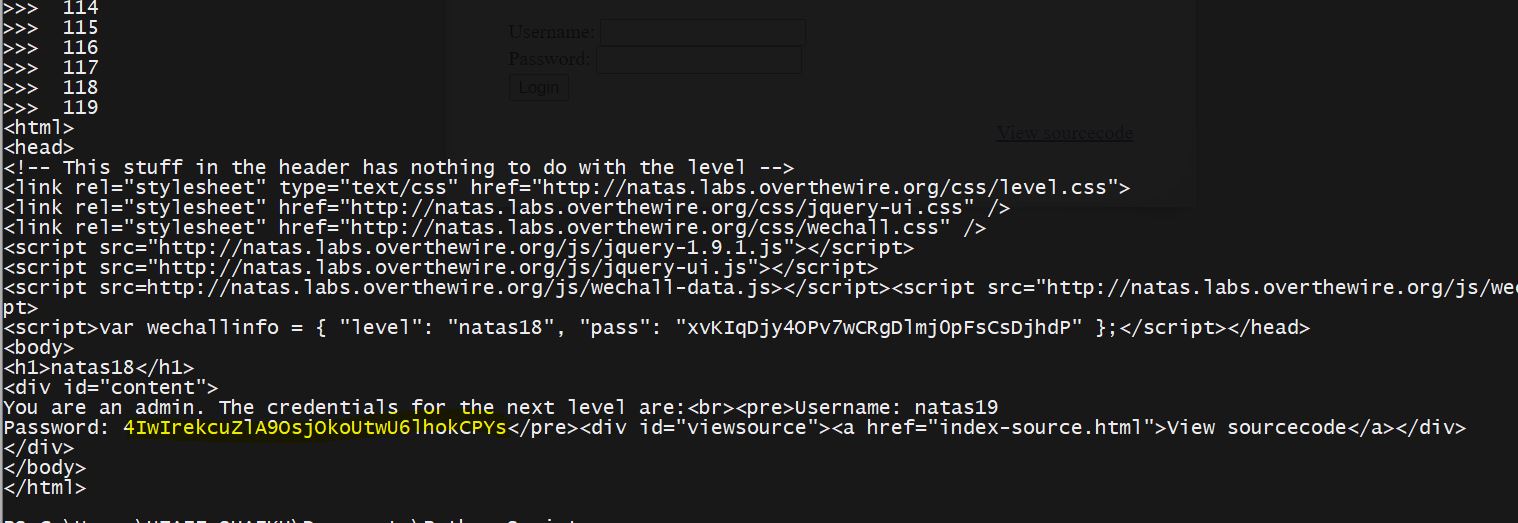
This level was really similar to the previous level as looking at the source code tell that's Linux command can be used when setting the word to look up in a dictionary.txt using word$("COMMAND") if the command fails the searched word will be returned and it the succeed than nothing is return this gives us a situation like the previous case where you could check the is the password matches the given string. So the same can be done for this case, use the same python script.



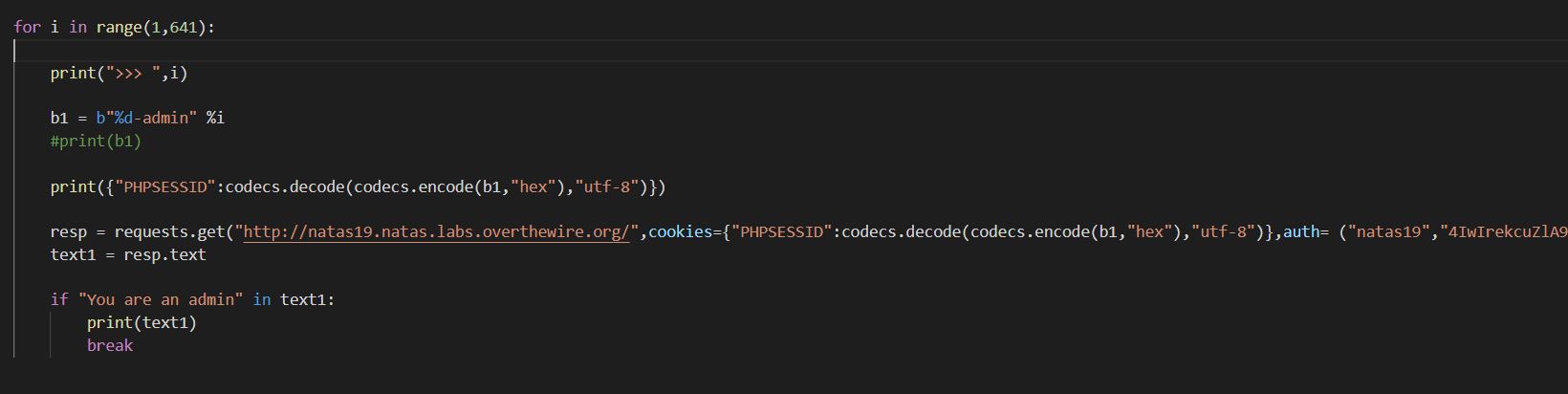
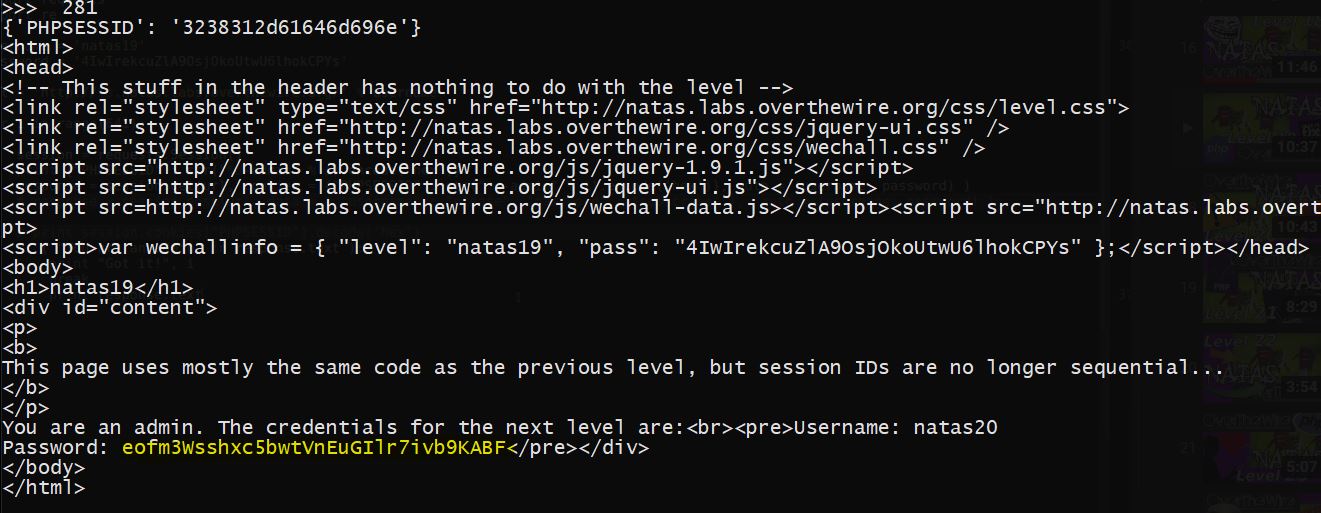
For Level 18, no input was given if the username exists so to get the password for this level, I had to use the SLEEP() command with the SQL query in order to make a delay between the request and the response. So using the same python script I can take out the time difference between the request and response so based on the time difference it's easy to check which character in the part of the password and after doing the brute force I got the password for the next level.



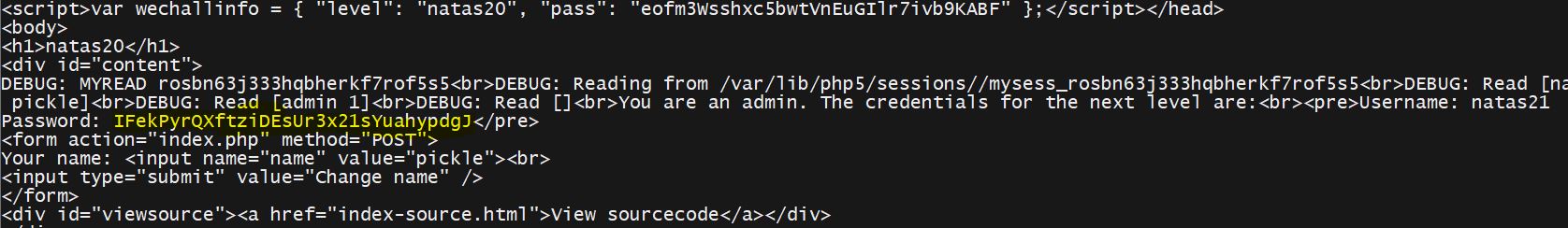
For this level, a login was given where every user is assigned a random number from 1 to 640, which will be set in your cookie, so in order to get the password for the level, I had to set the ID from 1 to 640 to find out which number is for the admin so I did this using a python script. and got the password for the next level the ID was the same as the admin.



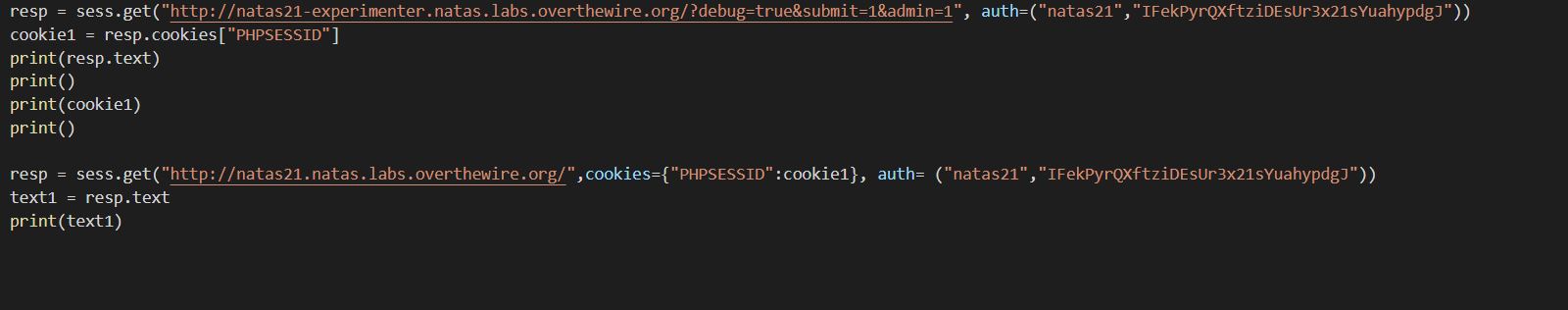
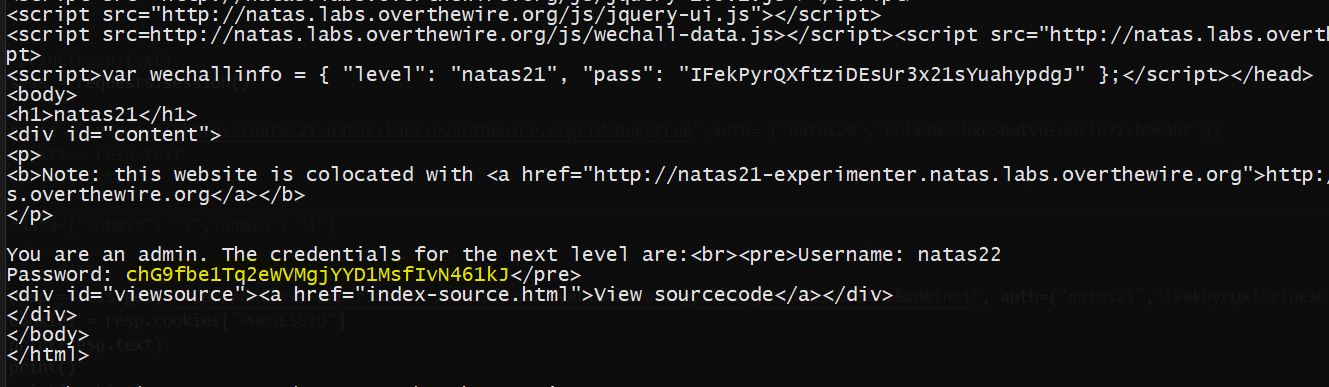
For Level 20, the session-id was kinda different but all of that sessions-id had 2d common in them it was somewhere in the middle for most of the time. So after testing out the hex into the Ascii, it had a random number with a "-" and the username, so to access as admin the username should be admin and to get the cookie for admin all I had to do is run a loop which sent cookie for every number with "-admin" doing that I got the password for the next level.



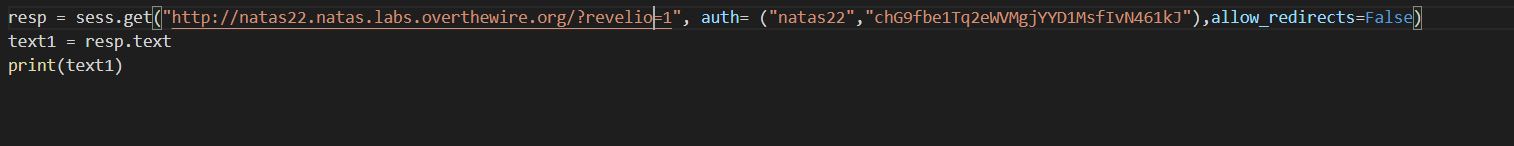
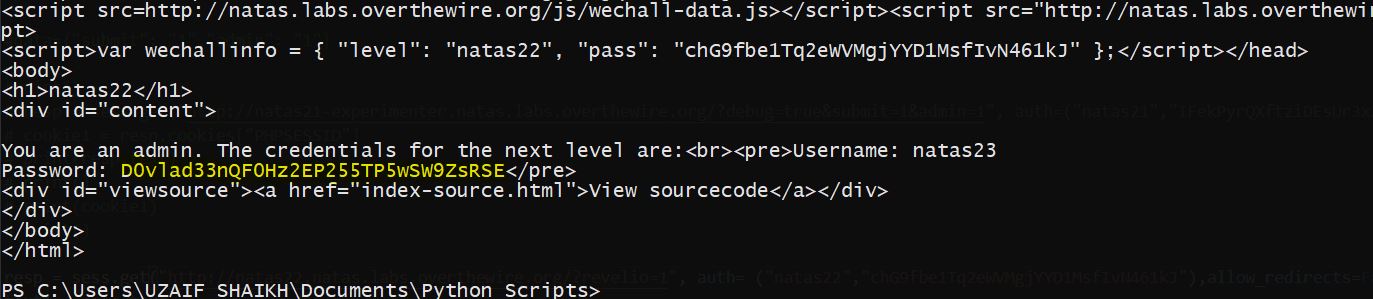
For this level, I had to looked at the source code and figure out in the debug set to true, it checks for a filename, and if it does exist it will created that file and and store the name if the requests are in the same session, so I used a python script to sent in the requests and set the name as "<name>\nadmin 1" so this will also write admin 1 in the file so in the next request it reads the name and set's the admin to 1 and I got the password for the next level.



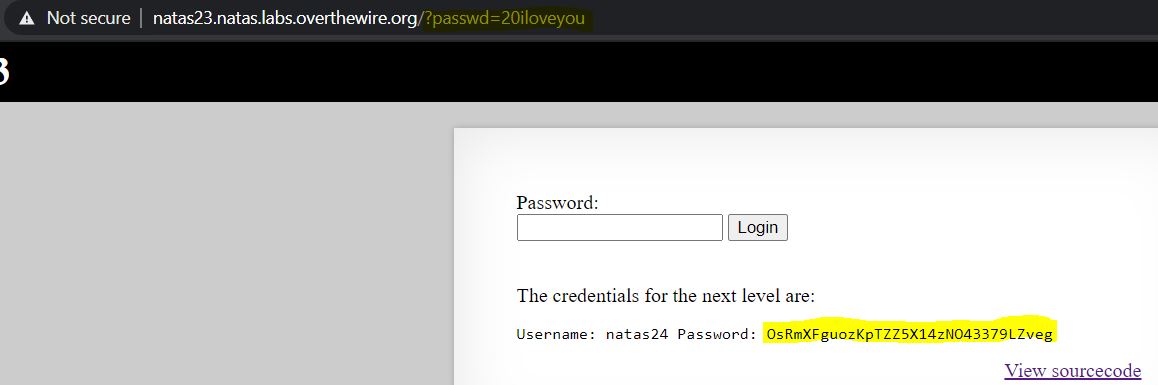
For level 22, there was an experiment webside where we could set the submit and the admin field as 1 in order to get the cookies for the admin and used the same cookies to get the password of the next level.



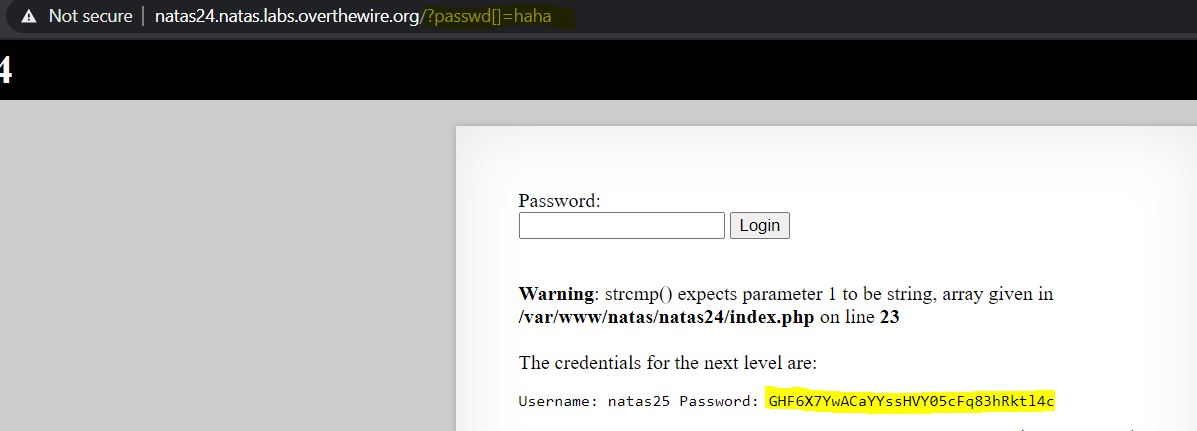
This level, didn't had any injection vulnerability, so looking at the source code it looks like it the admin is not 1 then it just redirects to a location so for this level, I just had to block the redirection and I got the password.



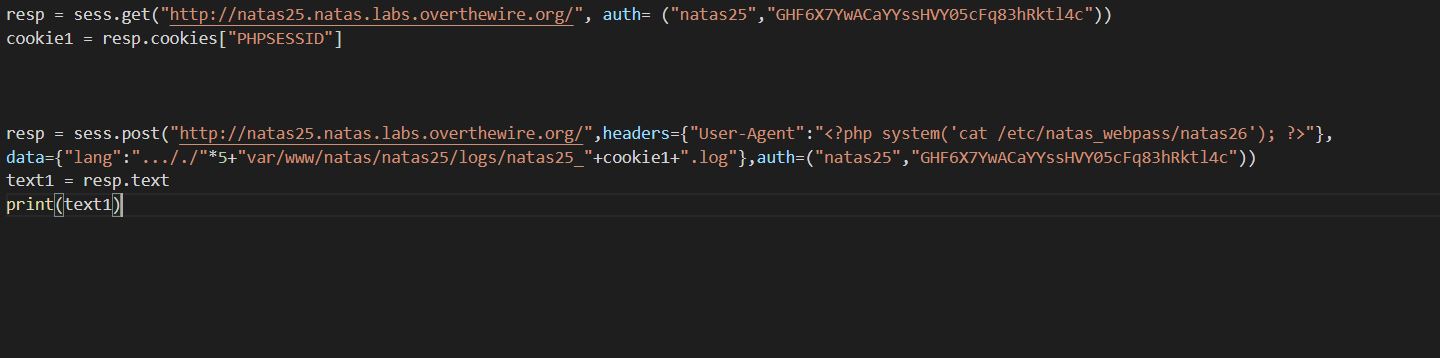
For this level, looking at the source code tells that the input password is strstr() with the string "iloveyou" and password > 10, so I looked at the data type in PHP and turns out PHP doesn't really care about the data type so I could use something like "23iloveyou" which will be true for both statement, and I got the password the next level.



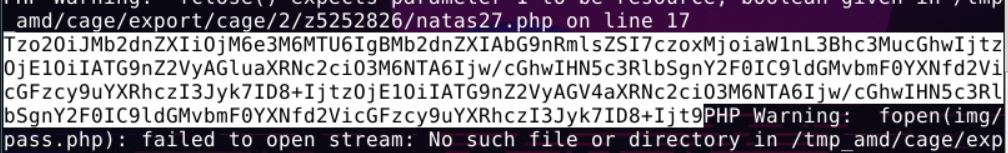
For level 25, the source code only was comparing the given password with password of natas25, so looking at the documentation of the PHP strcmp it turns out that if the argument is given as an array rather than a string, it will return 0 and we can get the password for the next level.

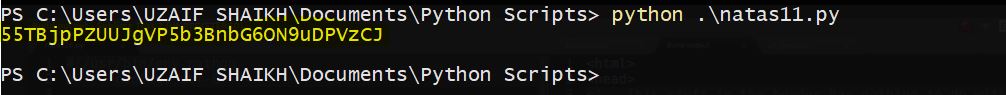


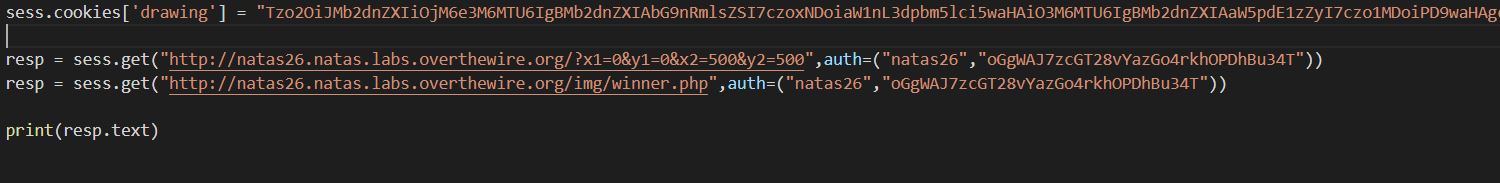
For Level 26, I had to do a lot of teacher and looking at the source code we can figure out that it does some sort of directory traversal, and we can use this to access the file "/var/www/natas/natas25/logs/natas25\_"+cookies+".log" to check what else can we do and looking at the file, it takes in the header so if we can use the header field to call the password for natas27, and it worked.

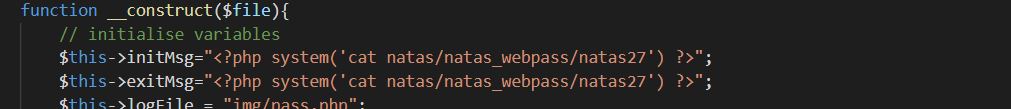


For level 27, looking at the source code it tells that the cookie contains a drawing field that is coming from the PHP code for the class so, I tried to set the class fields to set as a command which will get the password to the next level. so I used PHP code to get the cookie for it. and then injected that to the get request with the cookie, so I can get the password for the next level.

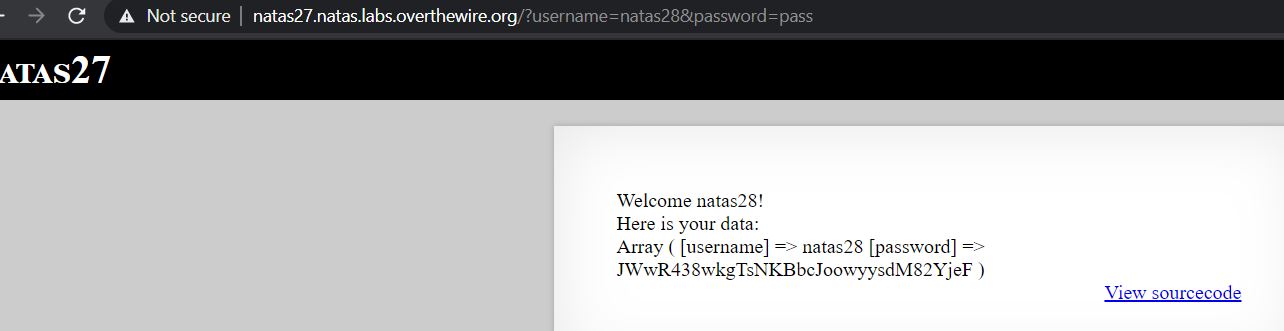


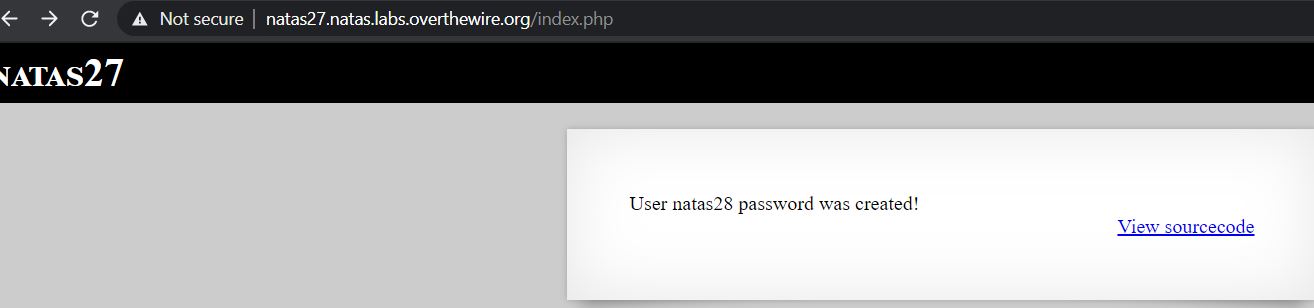






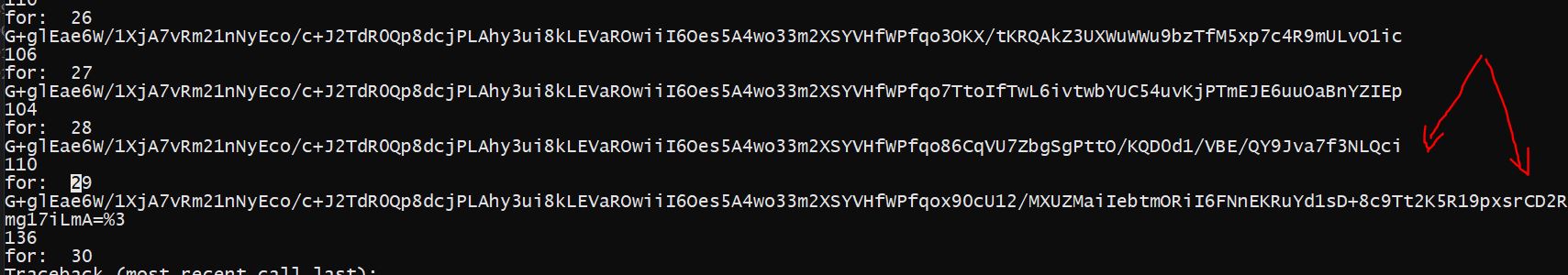
For Level 28, I looked at the source code and how clear that I had to take advantage of the SQL query so after some google searching I found out if the given input is more than the number of character, then it will be stored by cutting the extra number of character, so I injected the username as natas28 " "\*60 and the password for this as pass, the user has been created so to get the password, I just have to login with username "natas28" and the password pass as in SQL it will ignore the space and show all the data which have username natas28 and I got the password for it.

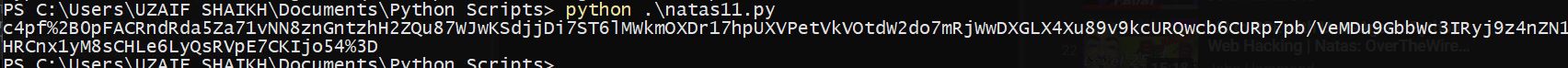


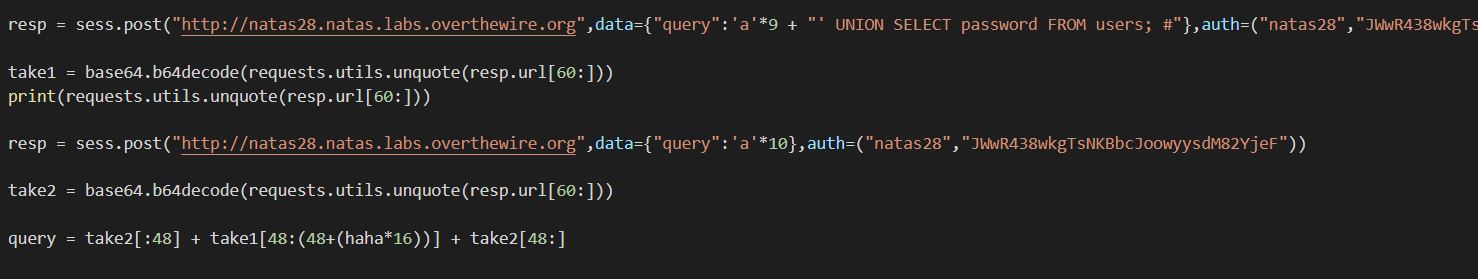


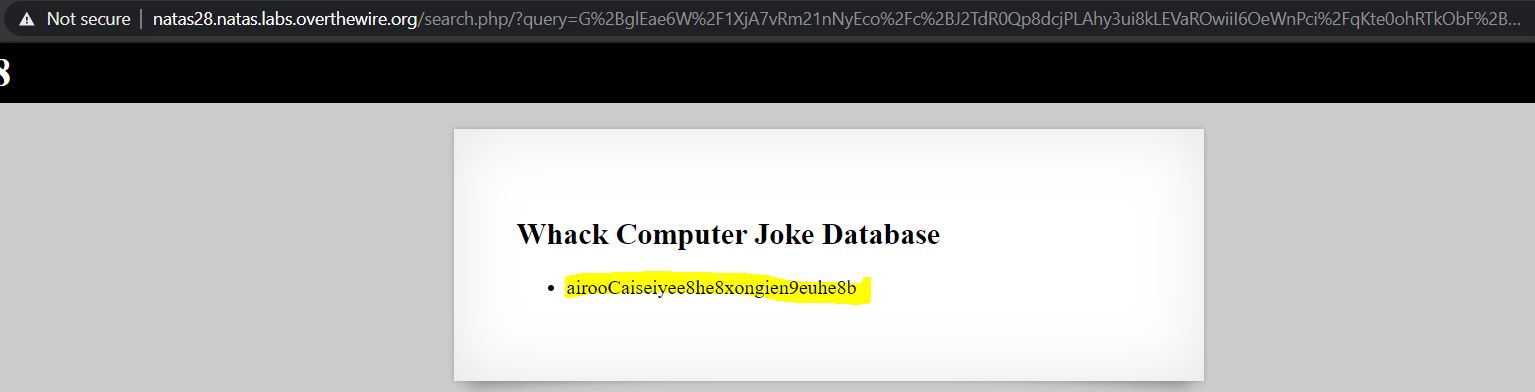
AHHH, This level took me a complete day to do, they don't give any source code so it was hard to find a hint, but when I looked at the URL I found the query is not plain text but encode and when I try to play around with query I got an error that incorrect padding in the block-cipher, So it was clear that the query is a block cipher encoding so I researcher about block cipher "https://www.youtube.com/watch?v=unn09JYIjOI" this video really helped me to understand about it and I after following the I found able to figure out the block size and the number of character which makes a block full and turns out it was 10 character, so using this I injected the SQL query for the password to the next level, and I had to set the block padding to make sure It doesn't throw an error, and after trying I was able to set the injection query in between a normal search query and injection the new query I got the password the next level.











For Level 30, I looked at the URL and try to cat index.pl and I found the code running at the back, that looking at that, it seems like it blocks whenever the file contains "natas" in it so I had to separate the word natas and then URL encode it and after that I code the password.

