# [Cyber Security Analyst Challenge](https://community.xvigil.com/t/cyber-security-analyst-challenge-hints/537" \o ")

**REPORT - WALKTHROUGH CLOUDSEK XVIGIL CTF**

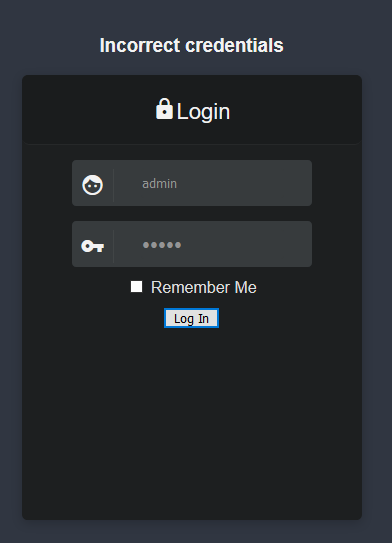
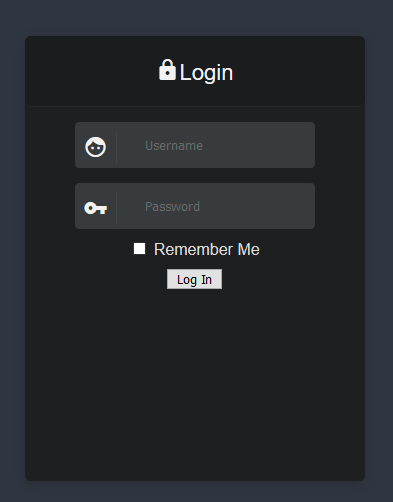
**By- Kashish Srivastava**

**Username(Kashish.Srivastava)**

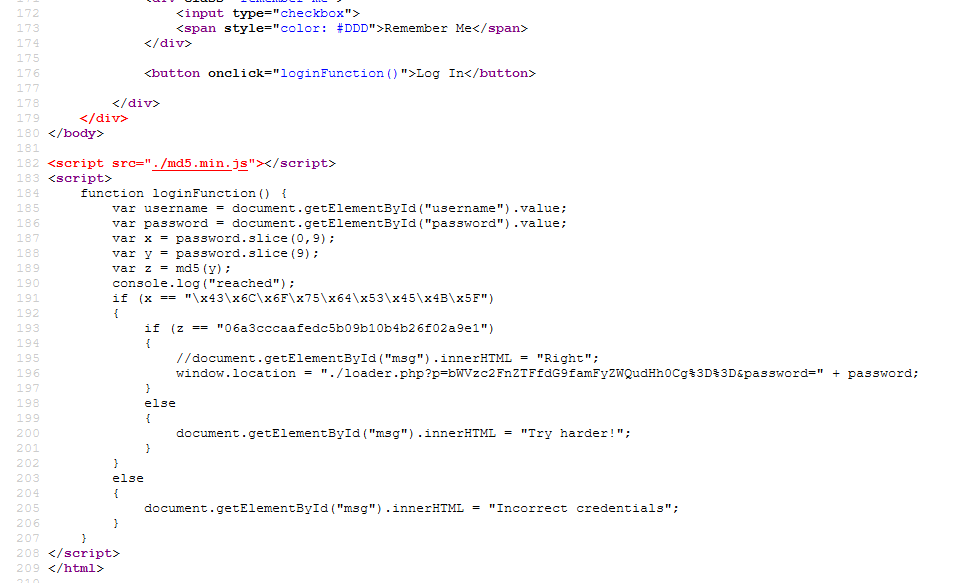
**LOGIN PORTAL**

We are provided with a login page with the **‘ip : 54.244.19.42’**

Tried logging with general username(admin) and password(admin) to check the response of the login (Invalid username/password) but did not get any useful response.

****

We check for the **‘source code of the login portal’**



In source code there’s a javascript code that can be reviewed.

As we review this code we see that, password is divided into two parts:

Slicing explains that the password is divided into two.

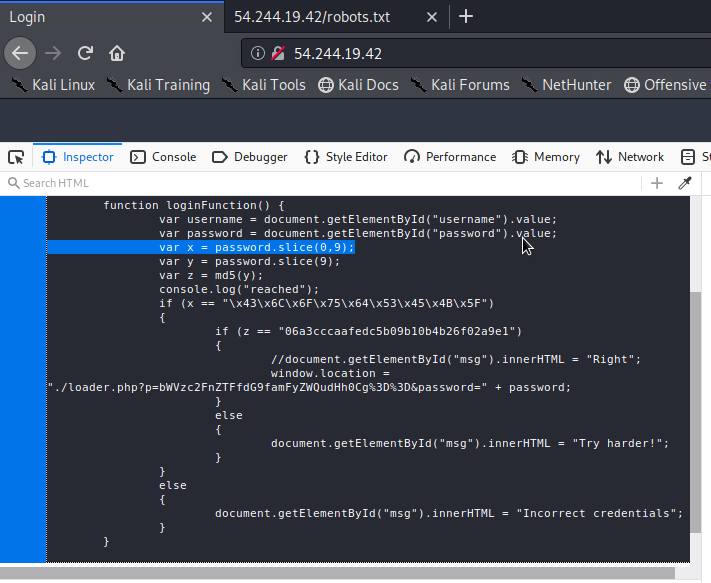
**var x = password.slice(0,9); var y = password.slice(9);**

x and y - **with x having values of hex (can be decoded by ant online hex decoder)**

**var z = md5(y)**;

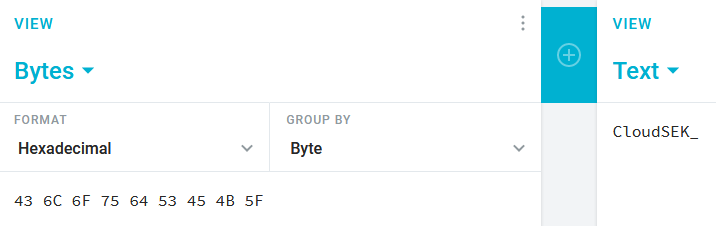
**Z is md5 of Y**, explaining that z has to be decrypted and that value will be of y

Therefore we can say password can be : X+Y (after decoding)



Online Hex-decoder is used to decode value of x : "**\x43\x6C\x6F\x75\x64\x53\x45\x4B\x5F**"

After decoding it was found as **CloudSEK\_ {Used:Cryptii(online)}**



z (**06a3cccaafedc5b09b10b4b26f02a9e1**) is md5 of y that is decoded as **jeniffer**

**{Used:Crackstation(online)}**



Whereas in the line:

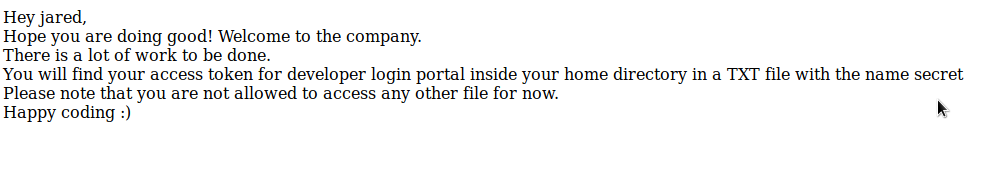
**window.location = "./loader.php?p=bWVzc2FnZTFfdG9famFyZWQudHh0Cg%3D%3D&password=" + password;**

**We get the password : CloudSEK\_jeniffer ( x+y)**

**LOGIN SUCCESSFUL, AFTER LOGIN**

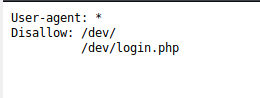
After login we get this message

*‘Hey jared,   
Hope you are doing good! Welcome to the company.   
There is a lot of work to be done.****You will find your access token for developer login portal inside your home directory in a TXT file with the name secret***  *Please note that you are not allowed to access any other file for now.   
Happy coding :)’*



Also checked **robots.txt**

**/dev/ is 403 forbidden**



**/dev/login.php**

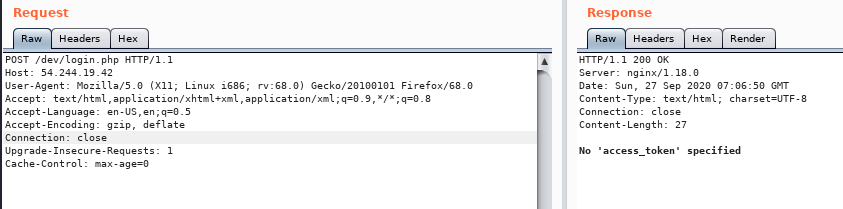
post request

**Picking up the 4 key-words as hints : Access token, home directory, secret file(txt) and Post request**

For /dev/login.php it mentioned that it will only accept post requests therefore tried burp suite for post request

We see that it says for **no ‘access\_token’ specified**

**{Used:Burp Suite}**



Therefore, this explains that for a successful post request we need to find the access token and pass it!

The information available to us after logging is that the access\_token is present in the home directory as a secret.txt file,

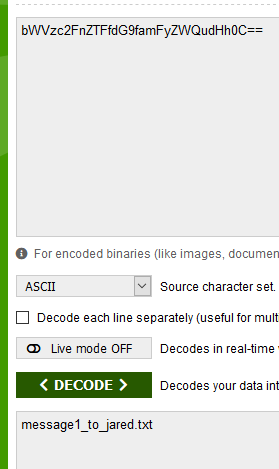
As a linux user its path can be framed as **/home/jared/secret.txt**

p is equal to the username and is base64 encoded

p=**bWVzc2FnZTFfdG9famFyZWQudHh0Cg==**

After decoding we get **message1\_to\_jared.txt** as the filename

**{Used:Base64decoder(online)}**



We need to find out a way to enter the home and view our secret.txt file, this is the moment when **Local File Inclusion** takes place

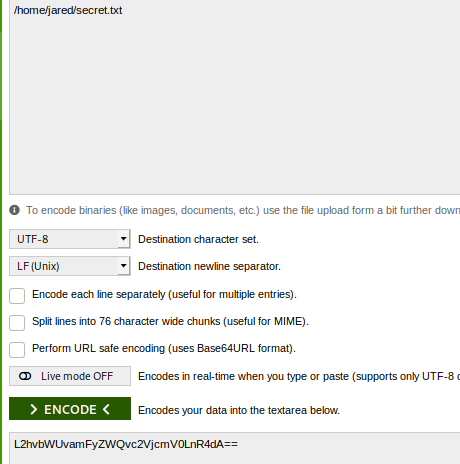
The link provided in hints for LFI helped a lot. Since **php code is vulnerable to LFI. We will try exploiting the loader.php code** in this reference.

**“./loader.php?p=bWVzc2FnZTFfdG9famFyZWQudHh0Cg%3D%3D&password="**

Here value of p is the **username** we will replace it by **/home/jared/secret.txt** to enter into the home directory and for our secret file.

To be apply it, it should be **base64 encoded** there dore we encode it:

**{Used:Base64encoder(online)}**

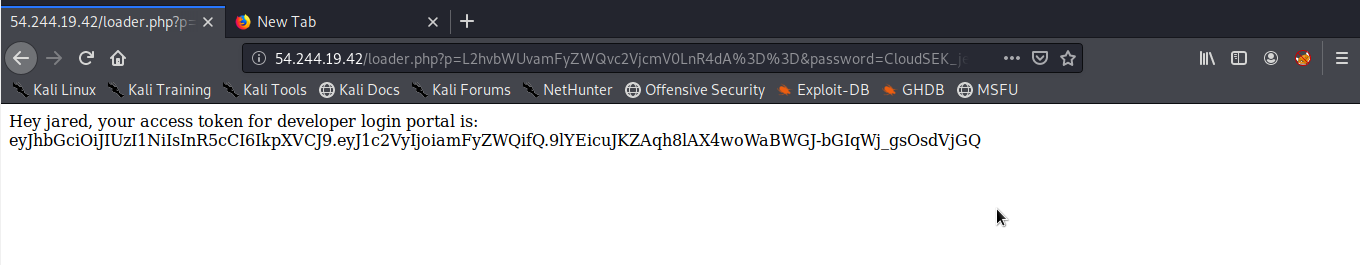


After encoding **/home/jared/secret.txt** we get : **L2hvbWUvamFyZWQvc2VjcmV0LnR4dA==**

We put the value of p equal to this encoded text in the url

./loader.php?**p=L2hvbWUvamFyZWQvc2VjcmV0LnR4dA%3D%3D**&password=

After searching on the url with different encoded value of p we find the **Access Token! for developer login portal**

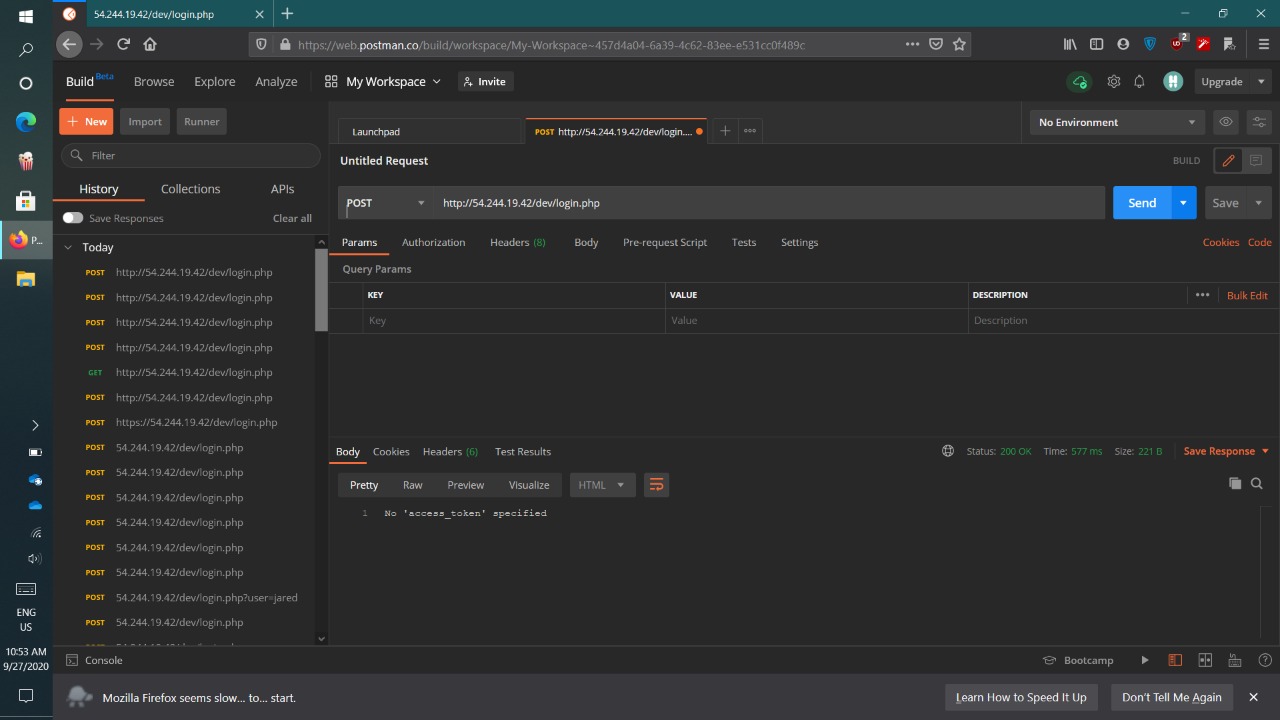


**Access token is : eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiamFyZWQifQ.9lYEicuJKZAqh8lAX4woWaBWGJ-bGIqWj\_gsOsdVjGQ**

**ACCESS TOKEN FOUND! JWT VIA POSTMAN**

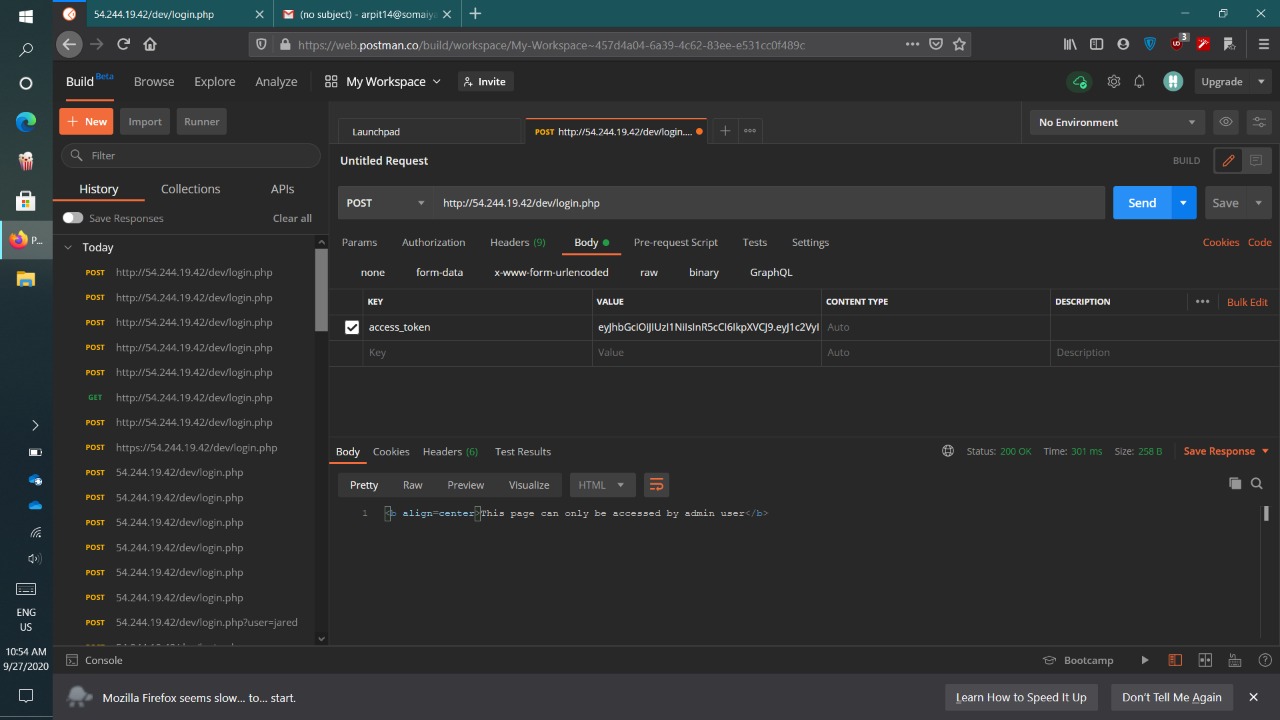
Using **POSTMAN for JWT implementation**, first of all the access token is passed as header it gives the error saying no access\_token specified.

**{Used:POSTMAN}**



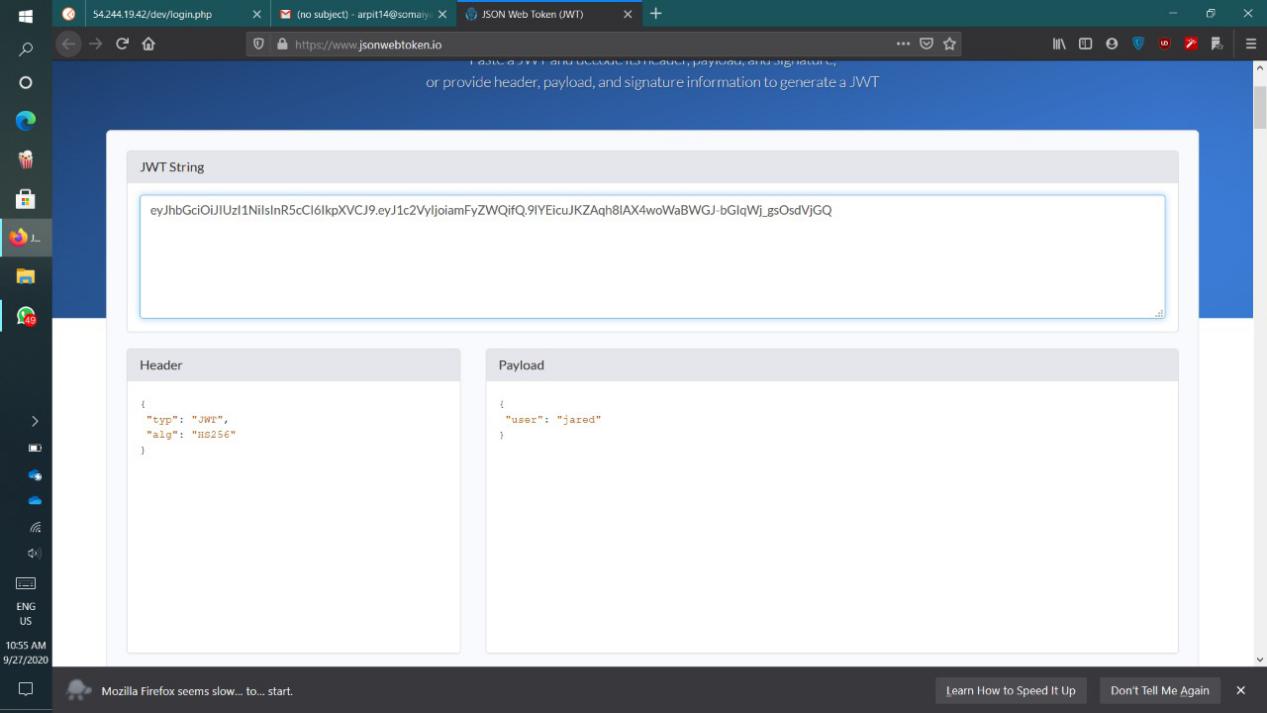
With the help of the hints provided we see that request has to be passed on as **POST data**.

It gives response as **“This page can only be accessed by the admin user”**

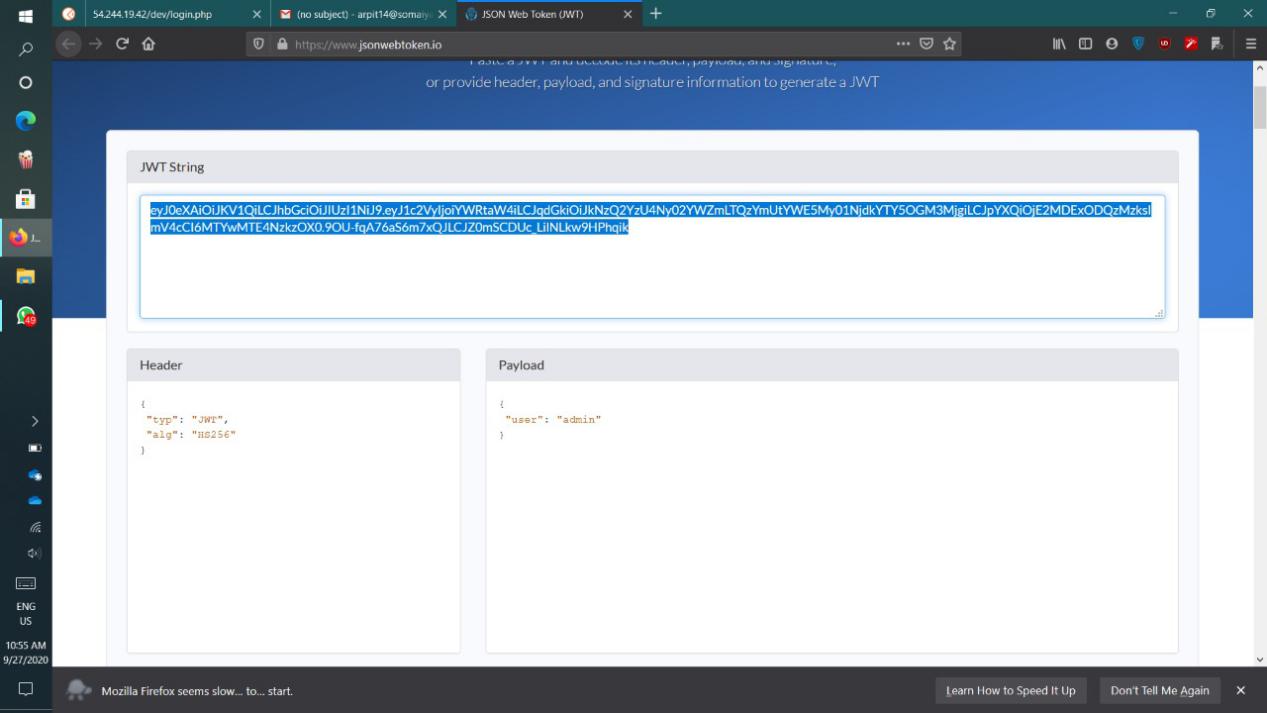
****

Since it is mentioned as admin user **we change user: jared to user: admin**

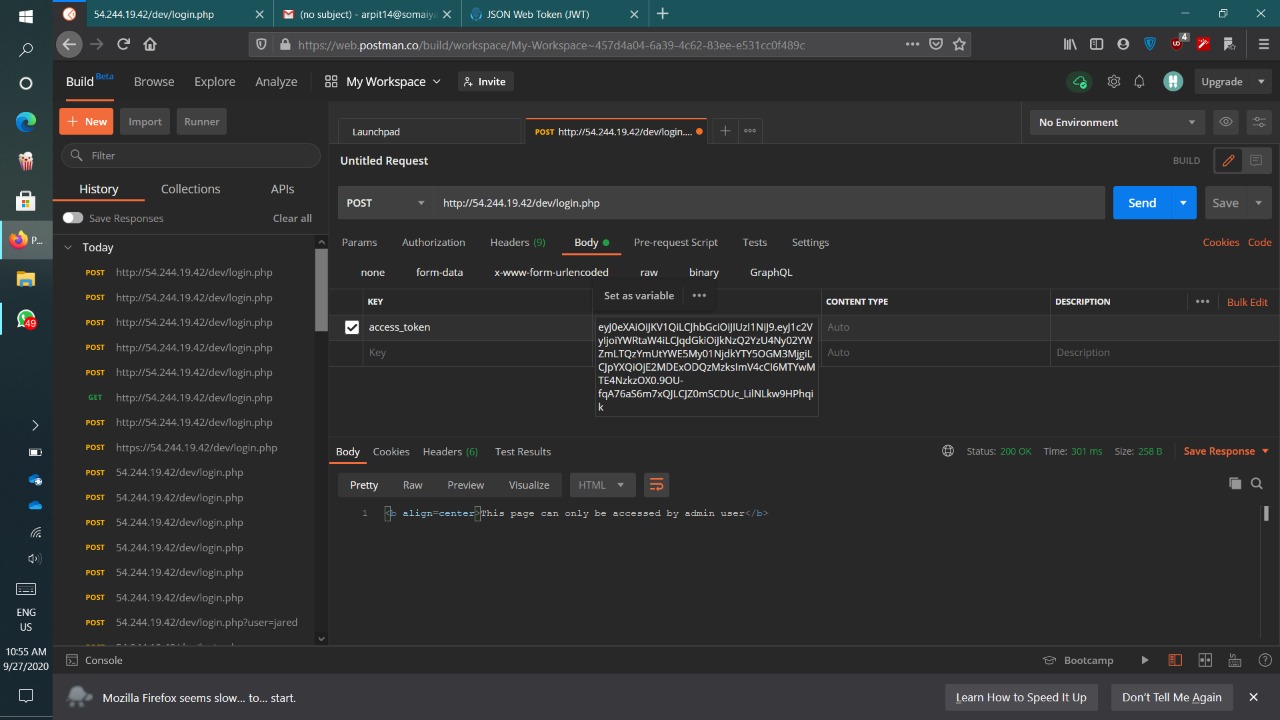
**{Used:Jsonwebtoken.io(online)}**



After changing the user: admin we copy the **new JWT string** as our access token for **admin user**

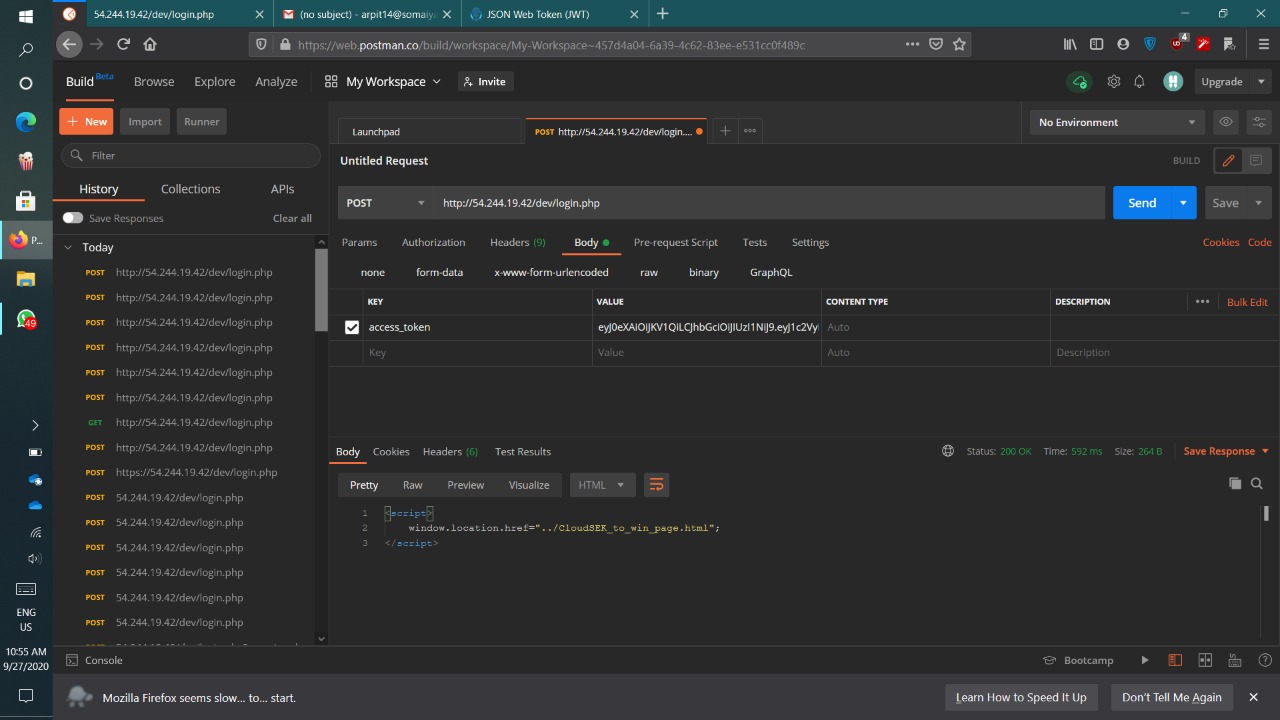


We send the **post data request with new access token**(JWT String)



After the request is send,

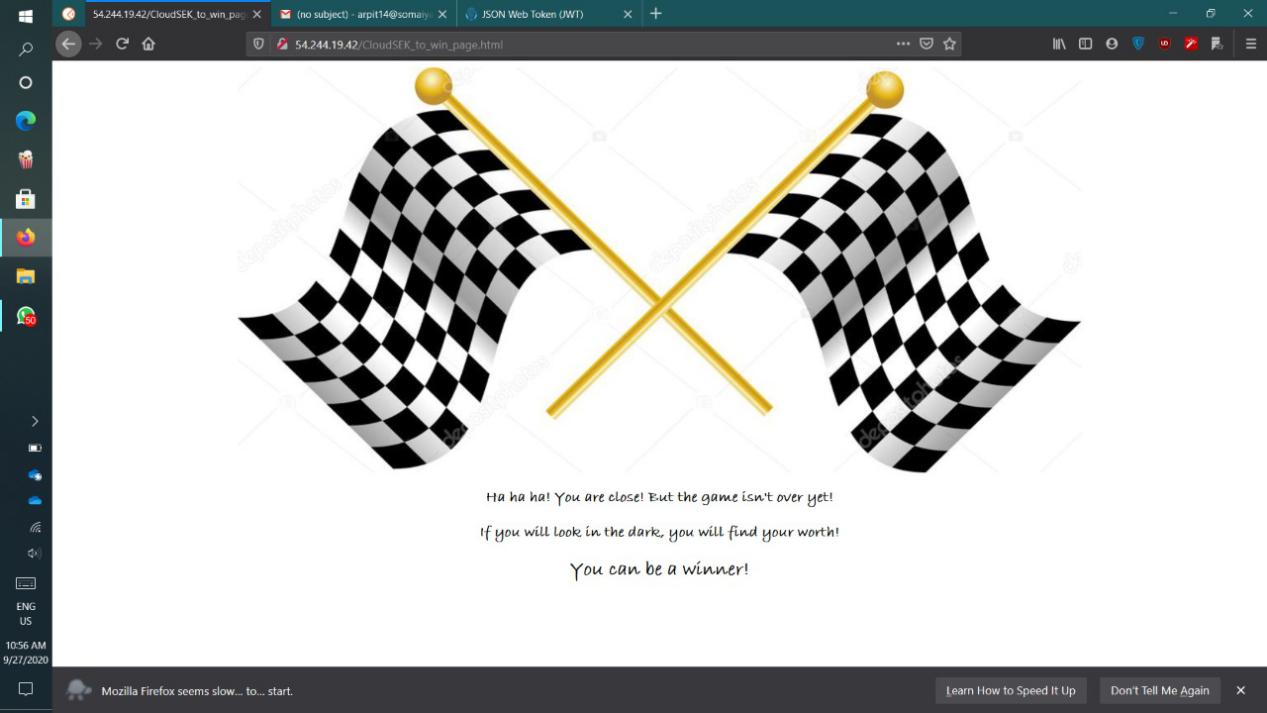
We find **./CloudSEK\_to\_win\_page.html** mentioned below in the body.



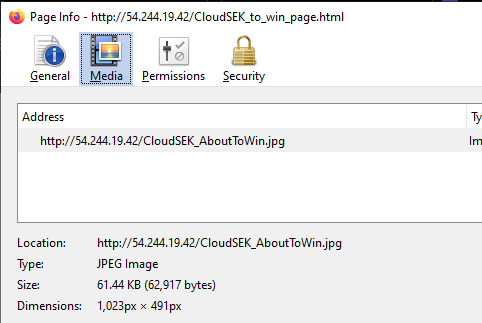
**./CloudSEK\_to\_win\_page.html**

**JWT ACCESS TOKEN DONE AND CLOUDSEC\_TO\_WIN\_PAGE!**

We access the page **./CloudSEK\_to\_win\_page.html** and find this image of two flags.



We check the image info that tells the name of the image as **CloudSEK\_AboutToWin.jpg** that hints about **stegnography** in this part of the challenge.

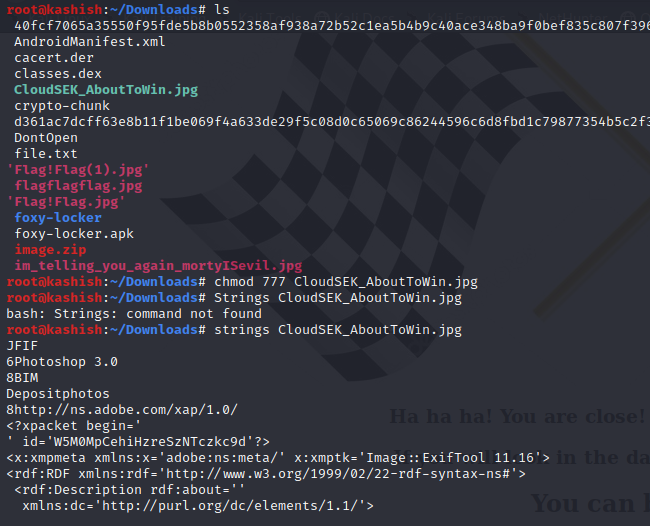


Its image name is CloudSEK\_AboutToWin.jpg, After downloading the image, all the permissions are given to the image.

Using command : **chmod 777 CloudSEK\_AboutToWin.jpg**

Next command is **: strings CloudSEK\_AboutToWin.jpg**

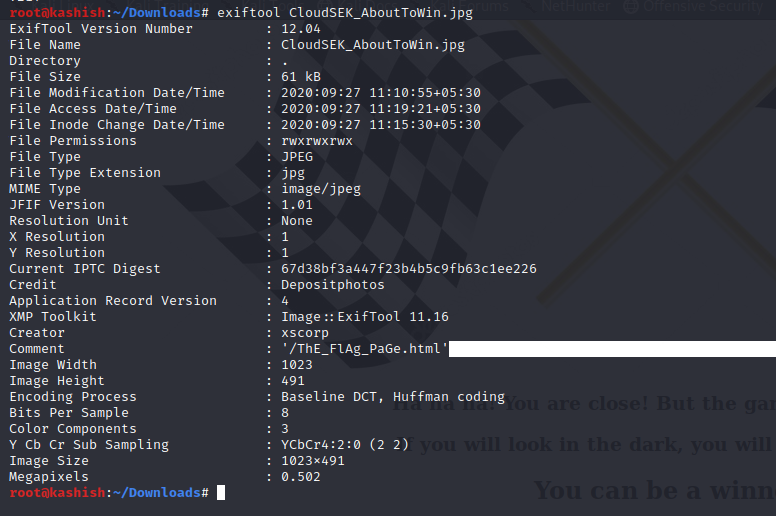
**There it is mentioned in a statement ‘Image::Exiftool 11.16’**



Using it as an hint, used exiftool on the image

Command used: **exiftool CloudSEK\_AboutToWin.jpg**

**{Used:Exiftool}**



We find the **Comment** mentioned as **‘/ThE\_FlAg\_PaGe.html’**

**FOUND THE FLAG! SUBMISSION LINK**

After opening **/ThE\_FlAg\_PaGe.html** we get the flag!

**FLAG: CloudSEK\_CTF\_2020{H4cKiNG\_i$\_FuN}**

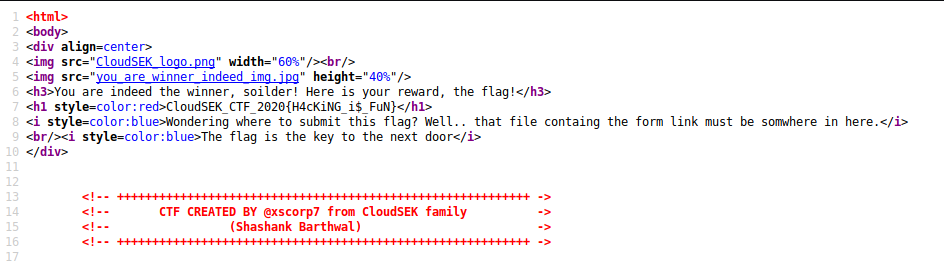
**We need to now find the link for submitting the flag and to access the form for submission!**

It also has an image **‘You Win’** explaining this might be a part of **image stegnography** again.



Checking the **page source** we see the image name as **you\_are\_the\_winner\_indeed\_img.jpg**

**As well as the creator!**



We use **steghide** to **extract the information** from the image

It is extracted to a **compl3tion\_m3ssag3.txt** file

Command used: **steghide extract -sf you\_are\_the\_winner\_indeed\_img.jpg**

The passphrase can be entered as the flag of our challenge.

**Enter passphrase: CloudSEK\_CTF\_2020{H4cKiNG\_i$\_FuN}**

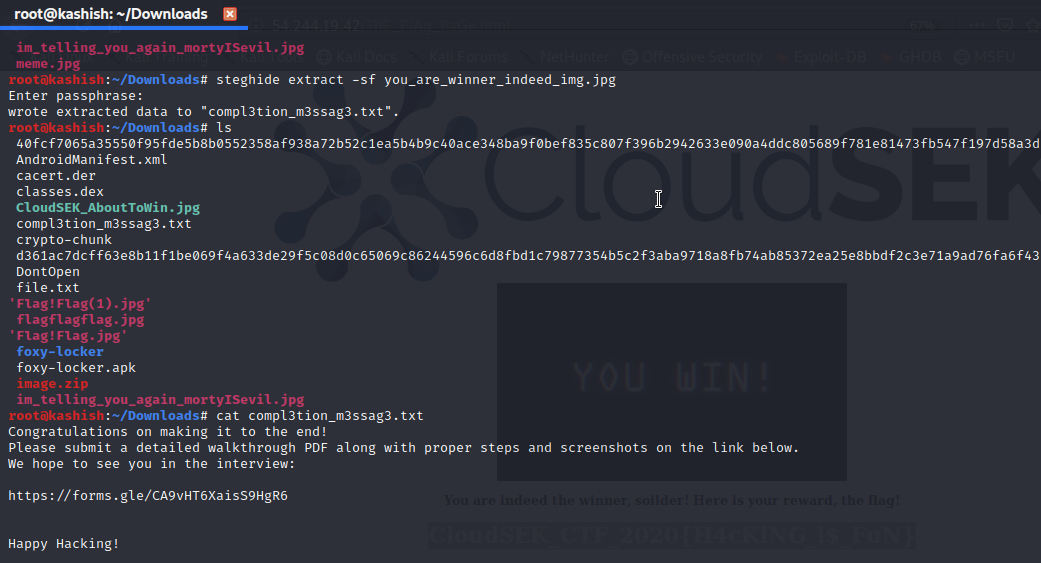
We see the message/contents of that image file has been extracted to a text file named as compl3tion\_m3ssag3.txt

After reading the conetnts of the file compl3tion\_m3ssag3.txt

We get a google form link for our report submission - <https://forms.gle/CA9vHT6XaisS9HgR6>

**Command used: cat compl3tion\_m3ssag3.txt**

**{Used:Steghide}**



Found the form link

**Thankyou this was a great ctf learned a lot!**

https://forms.gle/CA9vHT6XaisS9HgR6