

# Cyber Security Analyst Challenge

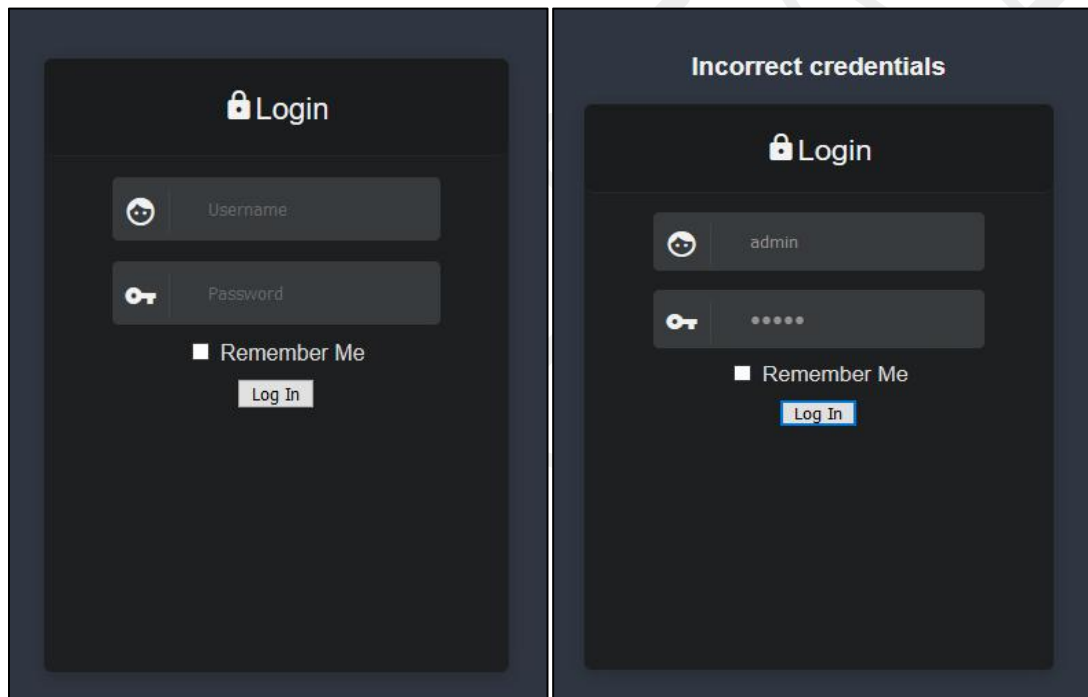
## REPORT - WALKTHROUGH CLOUDSEK XVIGIL CTF

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'

```
172     <input type="checkbox">
173     <span style="color: #DD">Remember Me</span>
174 </div>
175
176 <button onclick="loginFunction()">Log In</button>
177
178 </div>
179
180 </body>
181
182 <script src="md5.min.js"></script>
183 <script>
184     function loginFunction() {
185         var username = document.getElementById("username").value;
186         var password = document.getElementById("password").value;
187         var x = password.slice(0,9);
188         var y = password.slice(9);
189         var z = md5(y);
190         console.log("reached");
191         if (x == "\x13\x6C\x6F\x75\x64\x53\x45\x4B\x5F")
192         {
193             if (z == "06a3cccaafedc5b09b10b4b26f02a9e1")
194             {
195                 //document.getElementById("msg").innerHTML = "Right";
196                 window.location = "./loader.php?p=bWVzc2FnZTFidG9famFyZWQudH0Cg%3D%3D&password=" + password;
197             }
198             else
199             {
200                 document.getElementById("msg").innerHTML = "Try harder!";
201             }
202         }
203         else
204         {
205             document.getElementById("msg").innerHTML = "Incorrect credentials";
206         }
207     }
208 </script>
209 </html>
```

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.

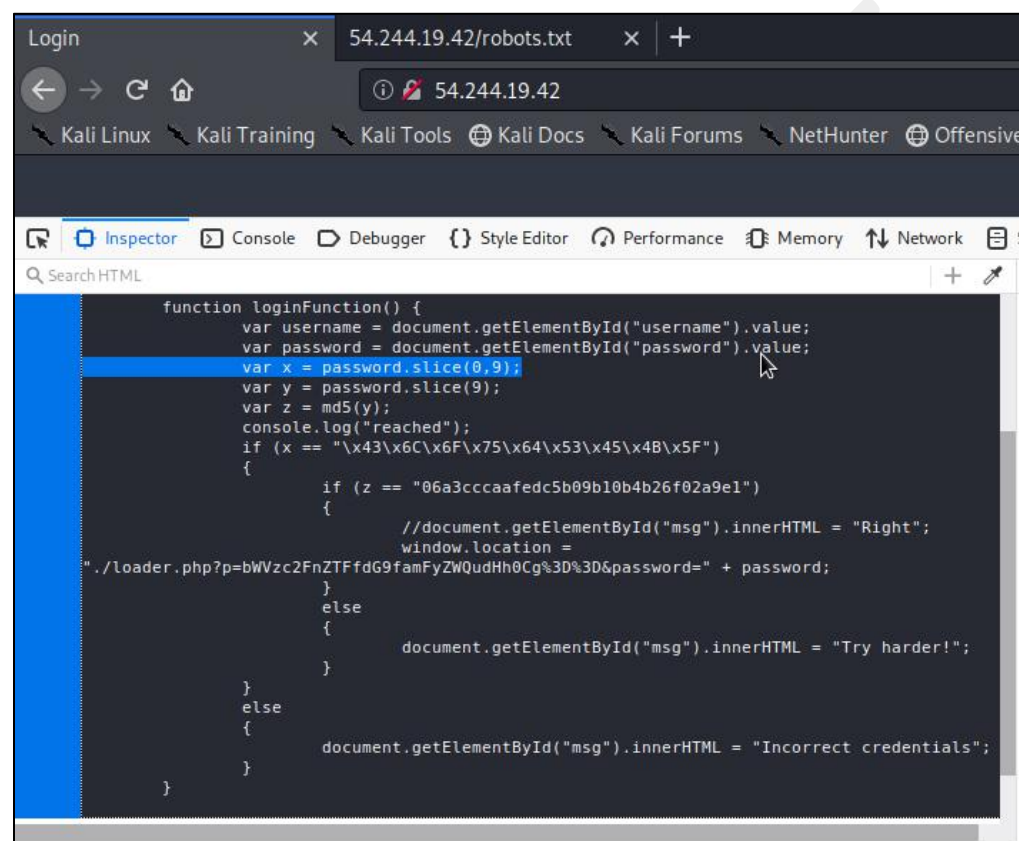
```
var x = password.slice(0,9);          var y = password.slice(9);
```

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)



```
function loginFunction() {
  var username = document.getElementById("username").value;
  var password = document.getElementById("password").value;
  var x = password.slice(0,9);
  var y = password.slice(9);
  var z = md5(y);
  console.log("reached");
  if (x == "\x43\x6C\x6F\x75\x64\x53\x45\x4B\x5F")
  {
    if (z == "06a3cccaafedc5b09b10b4b26f02a9e1")
    {
      //document.getElementById("msg").innerHTML = "Right";
      window.location =
        "../loader.php?p=bWVzc2FnZTFfdG9famFyZWQudH0Cg%3D%3D&password=" + password;
    }
    else
    {
      document.getElementById("msg").innerHTML = "Try harder!";
    }
  }
  else
  {
    document.getElementById("msg").innerHTML = "Incorrect credentials";
  }
}
```

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)}

VIEW		VIEW
Bytes ▾		Text ▾
FORMAT	GROUP BY	CloudSEK_
Hexadecimal ▾	Byte ▾	
43 6C 6F 75 64 53 45 4B 5F		

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

{Used:Crackstation(online)}

Hash	Type	Result
06a3cccaafedc5b09b10b4b26f02a9e1	md5	jeniffer

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

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 :).'*

```
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**

```
User-agent: *  
Disallow: /dev/  
          /dev/login.php
```

**/dev/login.php**

**This page only accepts 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}**

Request		Response	
Raw	Headers	Raw	Headers
<pre>POST /dev/login.php HTTP/1.1 Host: 54.244.19.42 User-Agent: Mozilla/5.0 (X11; Linux i686; rv:68.0) Gecko/20100101 Firefox/68.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Cache-Control: max-age=0</pre>		<pre>HTTP/1.1 200 OK Server: nginx/1.18.0 Date: Sun, 27 Sep 2020 07:06:50 GMT Content-Type: text/html; charset=UTF-8 Connection: close Content-Length: 27  No 'access_token' specified</pre>	

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,

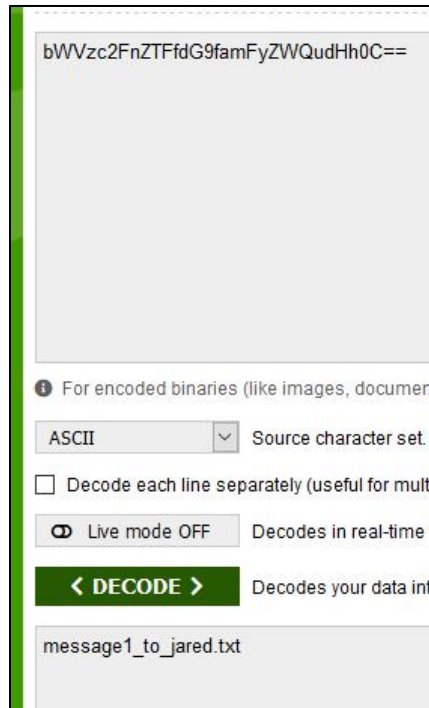
As a linux user its path can be framed as **/home/jared/secret.txt**

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)}**

/home/jared/secret.txt

**i** To encode binaries (like images, documents, etc.) use the file upload form a bit further down

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

☐ Encode each line separately (useful for multiple entries).

☐ Split lines into 76 character wide chunks (useful for MIME).

☐ Perform URL safe encoding (uses Base64URL format).

☒ Live mode OFF Encodes in real-time when you type or paste (supports only UTF-8)

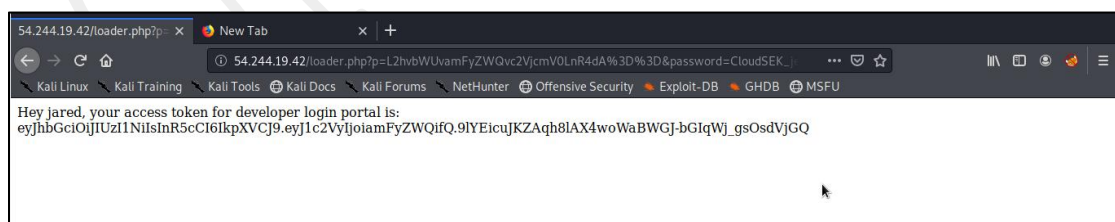
**> ENCODE <** Encodes your data into the textarea below.

L2hvbWUvamFyZWQvc2VjcmV0LnR4dA==

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**



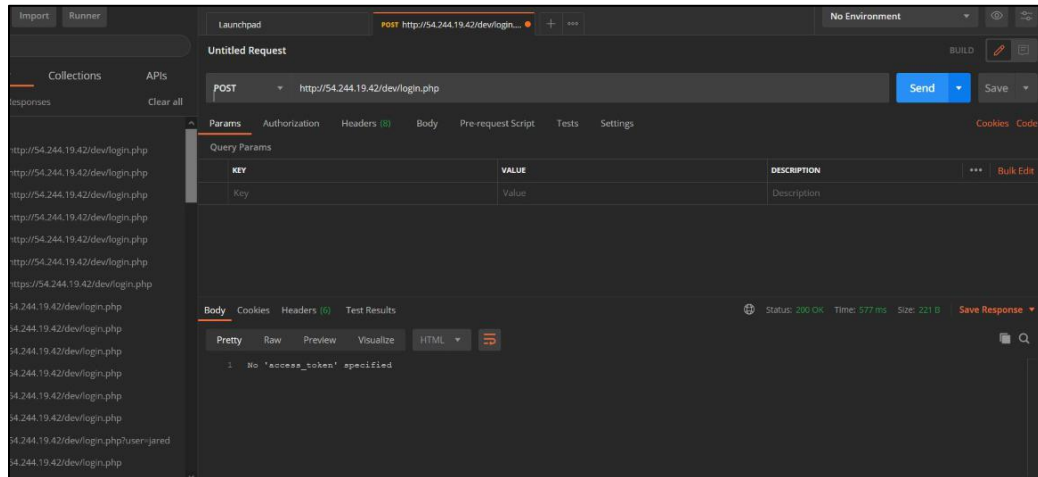
**Access token is :**  
**eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiamFyZWQifQ.9IYEicuJKZAqh8lAX4woWaBWGJ-bGIqWj\_gsOsdVjGQ**

---

## ACCESS TOKEN FOUND! JWT VIA POSTMAN

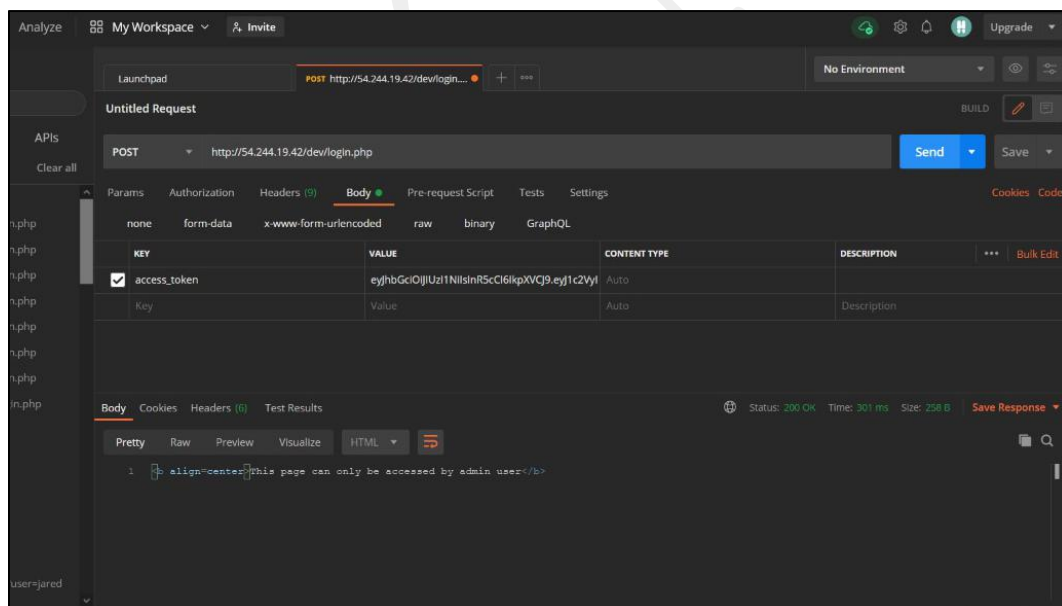
Using **POSTMAN** for **JWT implementation**, first of all the access token is passed as header it gives the error saying no access\_token specified.

{Used:POSTMAN}



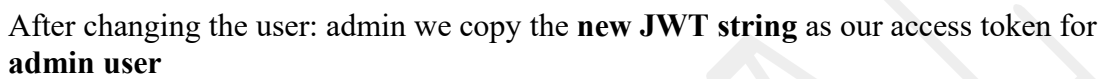
With the help of the hints provided we see that request has to be passed on as **POST data**.

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

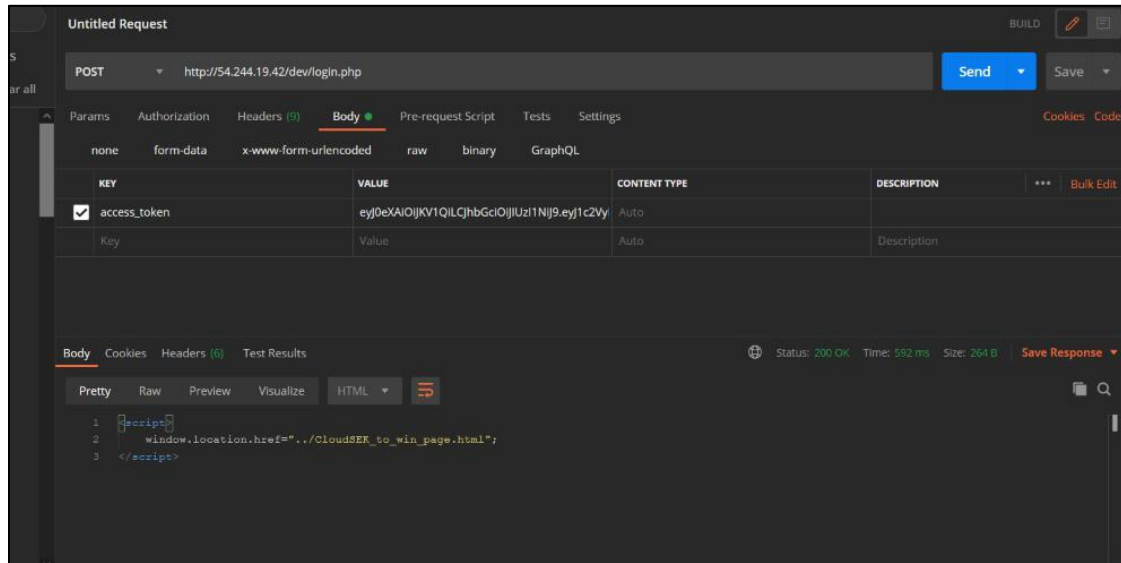


Since it is mentioned as admin user we change user: jared to user: admin

{Used:Jsonwebtoken.io(online)}







**`../CloudSEK_to_win_page.html`**

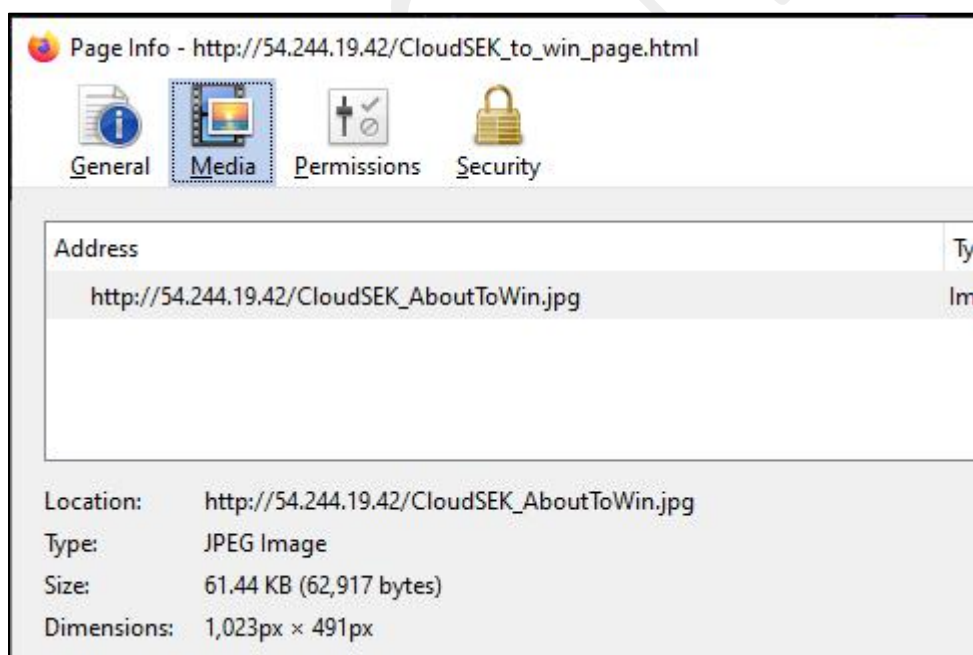
---

## **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 **steganography** 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'

```
root@kashish:~/Downloads# ls
40fcf7065a35550f95fde5b8b0552358af938a72b52c1ea5b4b9c40ace348ba9f0bef835c807f396
AndroidManifest.xml
cacert.der
classes.dex
CloudSEK_AboutToWin.jpg
crypto-chunk
d361ac7dcff63e8b11f1be069f4a633de29f5c08d0c65069c86244596c6d8fbd1c79877354b5c2f3
DontOpen
file.txt
'Flag!Flag(1).jpg'
'flagflagflag.jpg'
'Flag!Flag.jpg'
foxy-locker
foxy-locker.apk
image.zip
im_telling_you_again_mortyISevil.jpg
root@kashish:~/Downloads# chmod 777 CloudSEK_AboutToWin.jpg
root@kashish:~/Downloads# Strings CloudSEK_AboutToWin.jpg
bash: Strings: command not found
root@kashish:~/Downloads# strings CloudSEK_AboutToWin.jpg
JFIF
6Photoshop 3.0
8BIM
Depositphotos
8http://ns.adobe.com/xap/1.0/
<?xpacket begin='
' id='W5M0MpCehiHzreSzNTczkc9d'?>
<x:xmpmeta xmlns:x='adobe:ns:meta/' x:xmptk='Image::ExifTool 11.16'> in the da
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<rdf:Description rdf:about=''
xmlns:dc='http://purl.org/dc/elements/1.1/'>
```

Using it as an hint, used exiftool on the image

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

**{Used:Exiftool}**

```
root@kashish:~/Downloads# exiftool CloudSEK_AboutToWin.jpg
ExifTool Version Number      : 12.04
File Name                    : CloudSEK_AboutToWin.jpg
Directory                    : .
File Size                     : 61 kB
File Modification Date/Time   : 2020:09:27 11:10:55+05:30
File Access Date/Time        : 2020:09:27 11:19:21+05:30
File Inode Change Date/Time   : 2020:09:27 11:15:30+05:30
File Permissions              : rwxrwxrwx
File Type                    : JPEG
File Type Extension          : jpg
MIME Type                     : image/jpeg
JFIF Version                  : 1.01
Resolution Unit               : None
X Resolution                  : 1
Y Resolution                  : 1
Current IPTC Digest           : 67d38bf3a447f23b4b5c9fb63c1ee226
Credit                       : Depositphotos
Application Record Version    : 4
XMP Toolkit                   : Image::ExifTool 11.16
Creator                      : xscorp
Comment                       : '/ThE_FlAg_PaGe.html'
Image Width                   : 1023
Image Height                  : 491
Encoding Process              : Baseline DCT, Huffman coding
Bits Per Sample               : 8
Color Components               : 3
Y Cb Cr Sub Sampling          : YCbCr4:2:0 (2 2)
Image Size                    : 1023x491
Megapixels                    : 0.502
root@kashish:~/Downloads#
```

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 steganography** again.



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

As well as the creator!

```
1 <html>
2 <body>
3 <div align=center>
4 <br/>
5 
6 <h3>You are indeed the winner, soldier! Here is your reward, the flag!</h3>
7 <h1 style=color:red>CloudSEK_CTF_2020{H4cKiNG_i$_FuN}</h1>
8 <i style=color:blue>Wondering where to submit this flag? Well.. that file containing the form link must be somewhere in here.</i>
9 <br/><i style=color:blue>The flag is the key to the next door</i>
10 </div>
11
12
13 <!-- ++++++ CTF CREATED BY @xscorp7 from CloudSEK family ++++++ -->
14 <!-- (Shashank Barthwal) -->
15 <!-- ++++++ -->
16
17
```

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 contents 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}

```
root@kashish: ~/Downloads
im_telling_you_again_mortyISevil.jpg
meme.jpg
root@kashish:~/Downloads# steghide extract -sf you_are_winner_indeed_img.jpg
Enter passphrase:
wrote extracted data to "compl3tion_m3ssag3.txt".
root@kashish:~/Downloads# ls
40fcf7065a35550f95fde5b8b0552358af938a72b52c1ea5b4b9c40ace348ba9f0bef835c807f396b2942633e090a4ddc805689f781e81473fb547f197d58a3d
AndroidManifest.xml
cacert.der
classes.dex
CloudSEK_AboutToWin.jpg
compl3tion_m3ssag3.txt
crypto-chunk
d361ac7dcff63e8b11f1be069f4a633de29f5c08d0c65069c86244596c6d8fbd1c79877354b5c2f3aba9718a8fb74ab85372ea25e8bbdf2c3e71a9ad76fa6f43
DontOpen
file.txt
'Flag!Flag(1).jpg'
flagflagflag.jpg
'Flag!Flag.jpg'
foxy-locker
foxy-locker.apk
image.zip
im_telling_you_again_mortyISevil.jpg
root@kashish:~/Downloads# cat compl3tion_m3ssag3.txt
Congratulations on making it to the end!
Please submit a detailed walkthrough PDF along with proper steps and screenshots on the link below.
We hope to see you in the interview:

https://forms.gle/CA9vHT6XaisS9HgR6

Happy Hacking!
```

Found the form link

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

<https://forms.gle/CA9vHT6XaisS9HgR6>