VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

OWASP VITCC

PRESENTS

Official Writeup for the **WASPCON** CTF cum Hands-on session held on 10th October

LEVEL 1:

The first level was the simplest one of all. All we had to do was write the command “!owasp give me flag” in the OWASP discord server and we get the flag.

Flag for level 1: WASP{G3t_th3_f1r5t_flag}

LEVEL 2:

Level 2 had a bunch of clickable images, all of them except one rick-rolling us. To find the correct image, instead of trying out every image one by one, we view the source code of the page.

```
Line wrap
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
<!-- Fonts -->
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Kanit:wght@300&display=swap" rel="stylesheet">


<!-- Favicon -->
<link rel="icon" href="../images/owasp-logo.png">
    <title>Image</title>
    <h1 style="text-align:center; font-family: 'Kanit', sans-serif; font-size: 4rem;">Find the next flag</h1>
  </head>
  <body>
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not
    <a href="./y.php"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not found" style="float: left; width: 3
    <a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ"><img src="https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fi.imgflip.com%2Fzl4hy.jpg&f=1&nofb=1" alt="Image not


  </body>
</html>
```

We can see the second last image to not be a youtube link. Clicking on ./y.php we are redirected to the flag.

WASP{1704S5WY7URM38IZU7HL}

Flag for level 2: WASP{1704S5WY7URM38IZU7HL}

LEVEL 3:

Level 3 starts off with a nicely designed web page. Since there is nothing to directly do here, we again view the source code.

In the source code we find a comment `<!--V0FTUHtsRXYzMV8zX3IzQGNoZWQhISF9→.`
This looks like a base64 encryption. Going online for a base 64 decoder, we find the flag.

Decode from Base64 format

Simply enter your data then push the decode button.

V0FTUHtsRXYzMV8zX3IzQGNoZWQhISF9

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

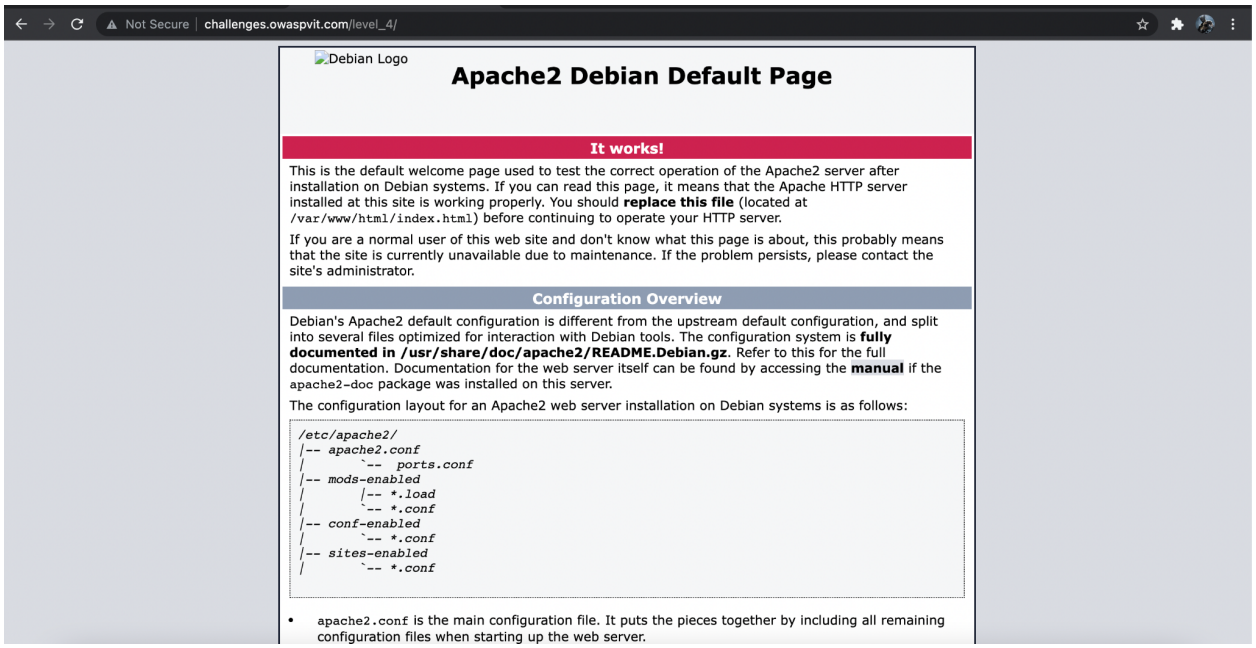
Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

WASP{lEv31_3_r3@ched!!!}

Flag for level 3: WASP{lEv31_3_r3@ched!!!}

LEVEL 4:



We have checked everything in level 4. There's nothing on the index page. Nothing of significance in the source code. So the next thing we do is check the robots.txt file. (robots.txt is a file on every webpage which contains additional setup instructions.)

So redirecting the webpage to http://challenges.owaspvit.com/level_4/robots.txt we find

```
User-agent: *
Disallow: /flag.txt/
```

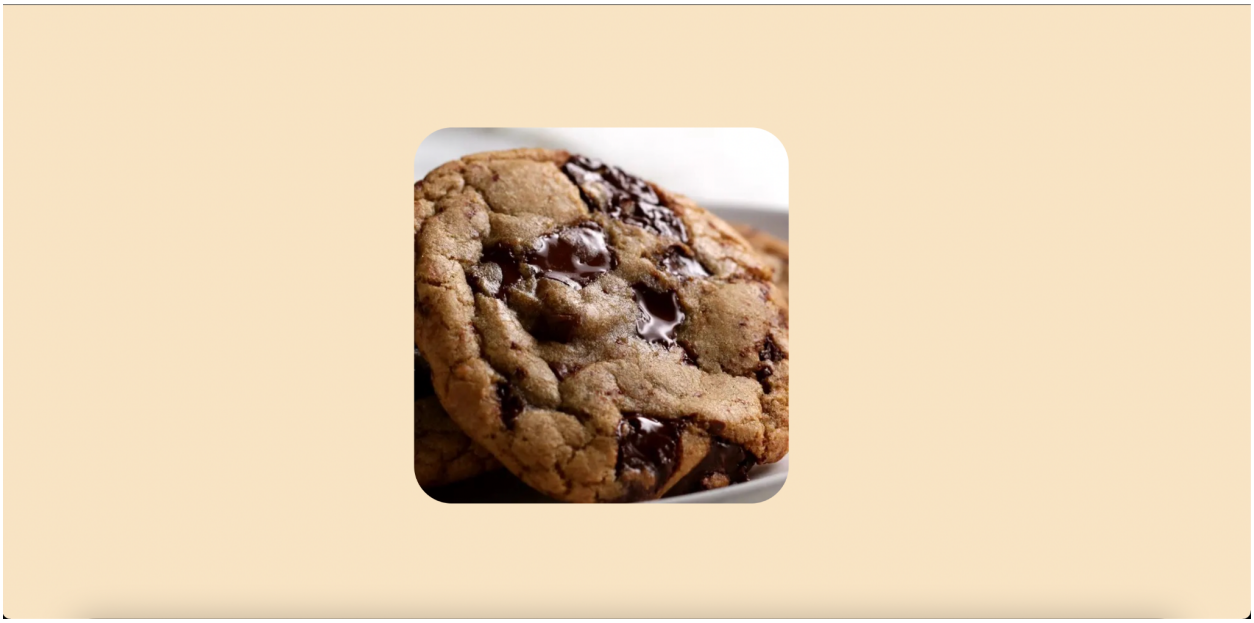
:

We see that there is another directory called flag.txt. So redirecting to http://challenges.owaspvit.com/level_4/flag.txt we get the flag.

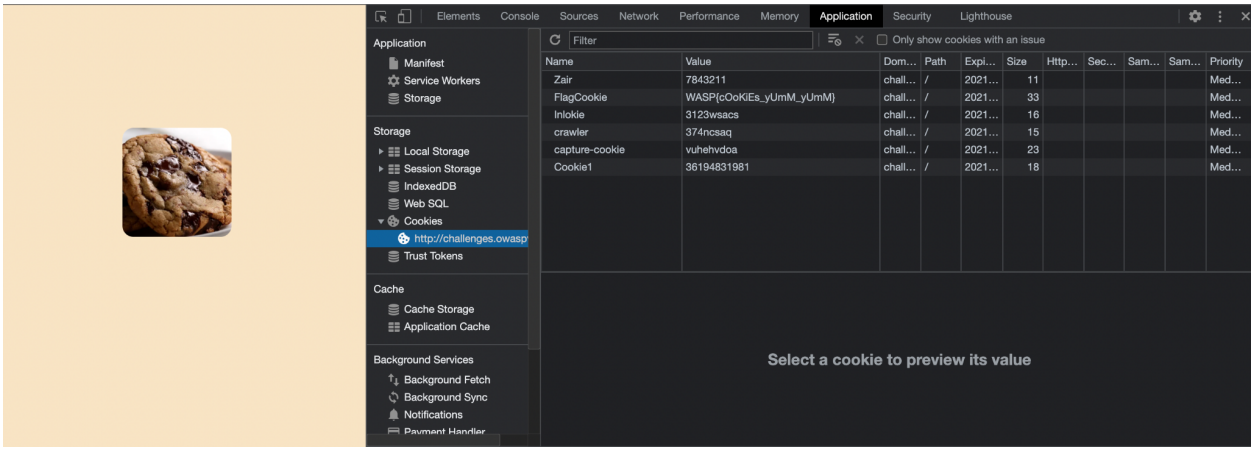

```
Nice work!
Flag = WASP{Mr_RoBoT_sAyS_hI}
```

Flag for level 4: WASP{Mr_RoBoT_sAyS_hI}

LEVEL 5:



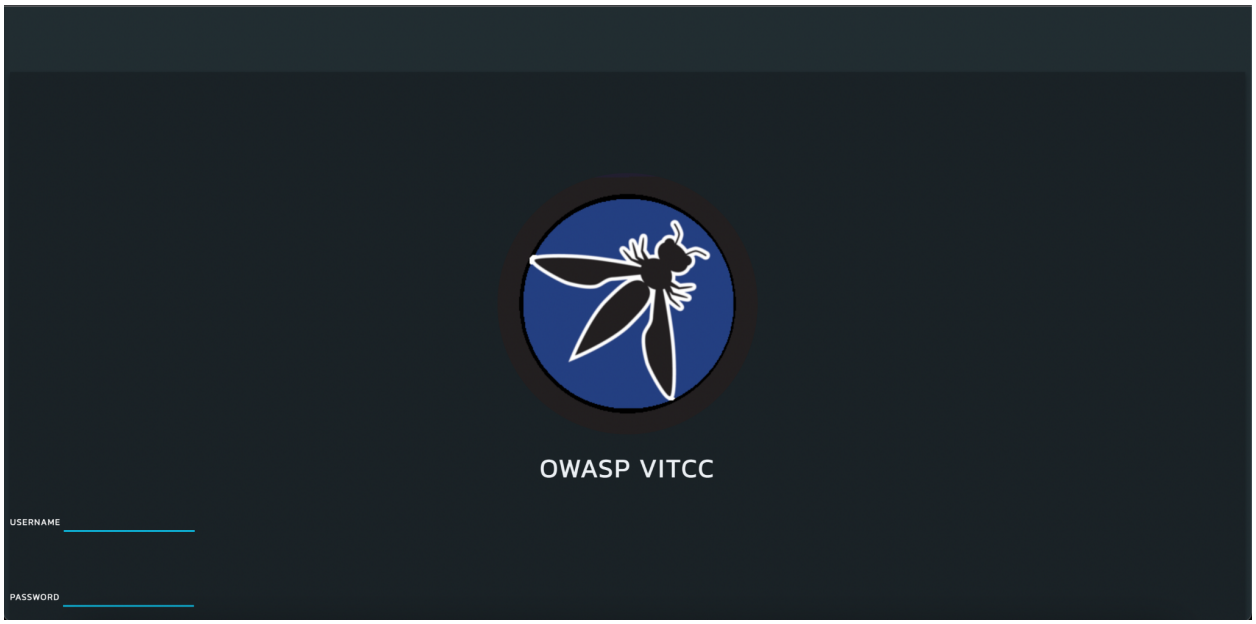
Level 5 is a simple html page with an image of a cookie in the middle. The cookie here refers to the cookies stored in the browser. We are expected to check the cookies here.
In chrome -> right click-> inspect -> Application -> Storage - > cookies.



We can see that the flag for this level is stored in the FlagCookie.

Flag for level 5: WASP{cOoKiEs_yUmM_yUmM}

LEVEL 6:



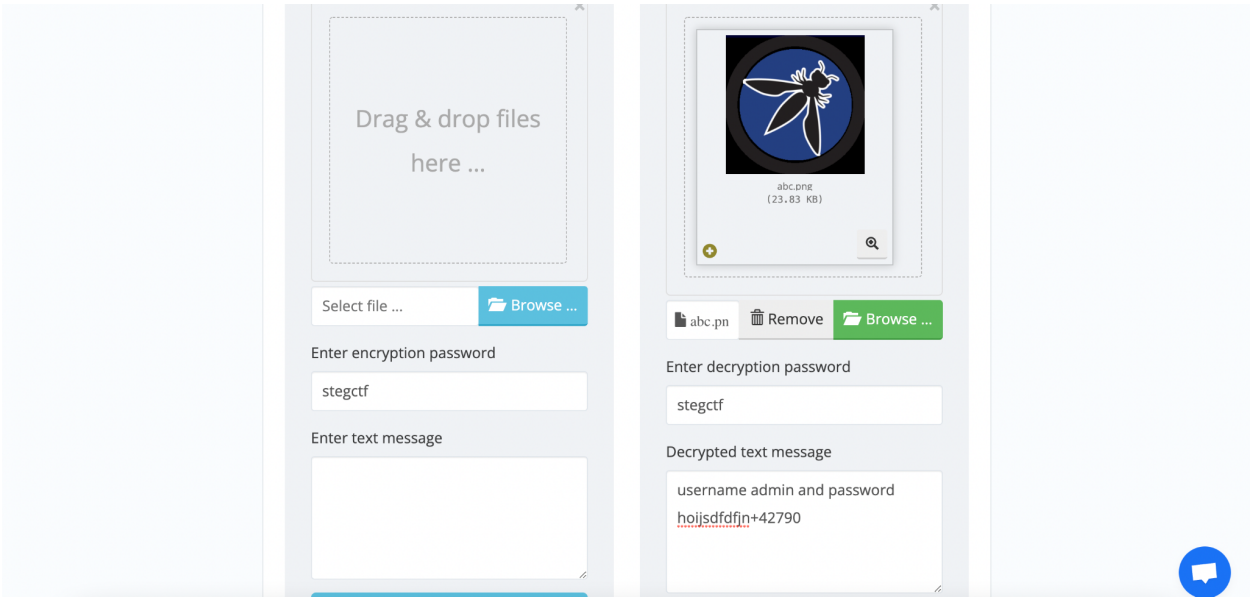
In the source code of this page we find two pieces of information.

```
            margin-right: auto;
            border-radius: 50%;
          }
      </style>
      <!--sdvvnhb iru vwhjdqrjudskb lv  vwhjfwi-->
  </head>
  <body>
      <div class="container">
          <div class="row">
          <div class="col-lg-3 col-md-2"></div>
          <div class="col-lg-6 col-md-8 login-box">
              <div class="col-lg-12 login-key">
                  <i class="fa fa-key" aria-hidden="true"></i>
              </div>
              <section class
                  <p><img src= ./Steg_CTF-encoded.png" class = "center" style="width:300px;height:300px;"  ></p>
              </section>
              <div class="col-lg-12 login-title">
                  OWASP VITCC
              </div>

              <div class="col-lg-12 login-form">
                  <div class="col-lg-12 login-form">
                      <form method = "GET" action = "./Steg.php">
                          <div class="form-group">
                              <label class="form-control-label" style="color:white;">USERNAME</label>
                              <input type="text" required name = "username" class="form-control">
                          </div>
                          <div class="form-group">
                              <label class="form-control-label" style="color:white;">PASSWORD</label>
                              <input type="password" required name = "password" class="form-control" i>
                          </div>

                          <div class="col-lg-12 loginbttm">
                              <div class="col-lg-6 login-btm login-text">
                                  <!-- Error Message -->
                              </div>
                              <div class="col-lg-6 login-btm login-button">
                                  <button type="submit" class="btn btn-outline-primary">LOGIN</button>
                              </div>
                          </div>
                      </form>
                  </div>
```

The first is a ROT encoded message. Going to rot13.com we can try out each ROT to find that it is ROT-23 encoded. The decoded message is - passkey for steganography is stegctf.

So we have found a passkey. And also steganography is mentioned. Steghide is one such steganography tool which is used to encode/decode hidden messages in photos and audio files.

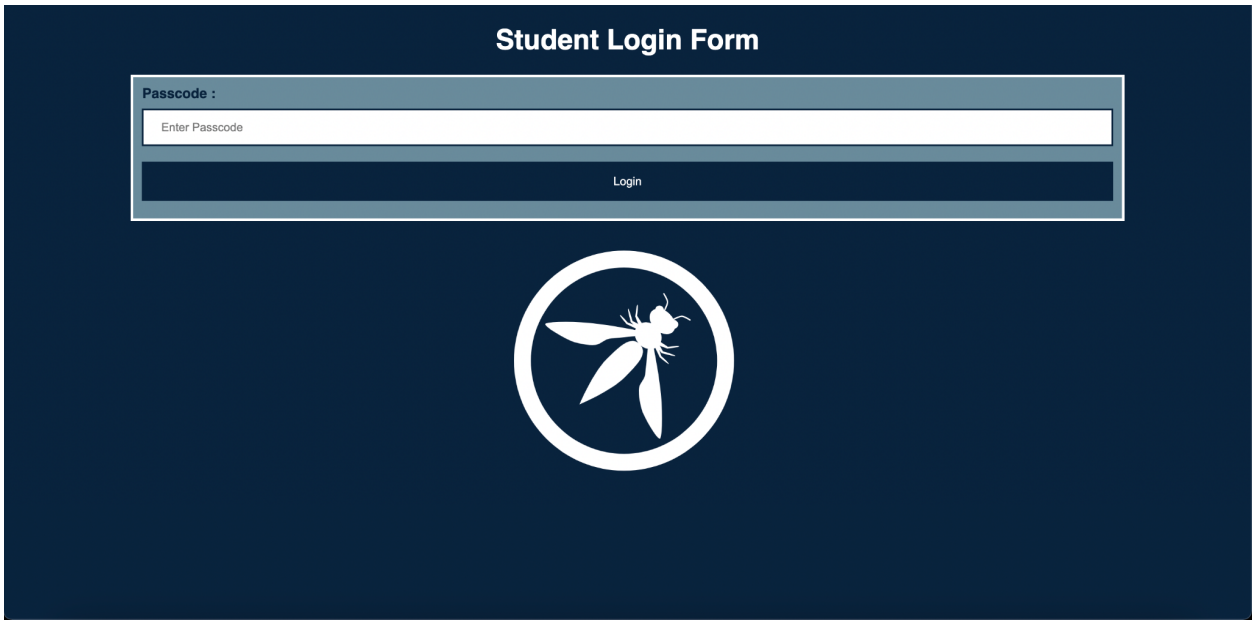
Downloading the image we found in the source page we move towards an online steghide tool and enter the passkey to get -



Submitting the username and password in the index page of level 6 we get the flag.

Flag for level 6: WASP{CTF_stg1729}

LEVEL 7:



‘

Here in the source code of the page at the bottom we find a code.

```
function generateId(len) {
  var arr = new Uint8Array((len || 40) / 2);
  window.crypto.getRandomValues(arr);
  return Array.from(arr, (dec)=>dec.toString(16).padStart(2, "0")).join("");
}


var caesarShift = function (str, amount) {
  if (amount < 0) {
    return caesarShift(str, amount + 26);
  }

  var output = "";

  for (var i = 0; i < str.length; i++) {
    var c = str[i];

    if (c.match(/[a-z]/i)) {
      var code = str.charCodeAt(i);

      if (code >= 65 && code <= 90) {
        c = String.fromCharCode(((code - 65 + amount) % 26) + 65);
      } else if (code >= 97 && code <= 122) {
        c = String.fromCharCode(((code - 97 + amount) % 26) + 97);
      }
    }
    output += c;
  }

  return output;
};

validate = () => {
  var form = document.getElementById("theform");
  function handleForm(event) {
    event.preventDefault();
  }
  form.addEventListener("submit", handleForm);
  let username = document.getElementById("pc").value;
    if (caesarShift(username, 446) == "PIzIpwIzIrJpeK") {

      document.getElementById("flag").innerHTML =
        "Congratulations! Your flag is your input encoded in format WASP{input}";
  }
  else{
```

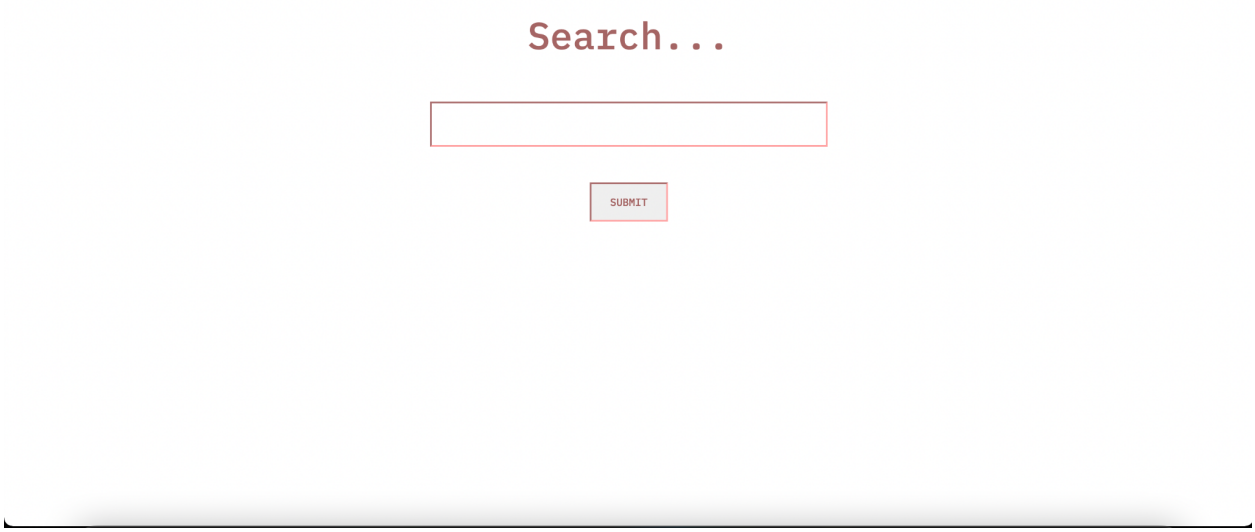
Looking closely at the code we can understand what it is trying to do. It’s obvious that there is caesar shift involved, with the shift number 446, and the encoded word - `PIzIpwIzIrJpeK`

Any online caesar shift decoder will decode the message and give the output as `LEvElsEvEnFlaG`

Submitting this in the login page we are told that LEvElsEvEnFlaG is the flag in the format WASP{flag}

Flag for level 7: WASP{LEvElsEvEnFlaG}

LEVEL 8:



This level is dedicated to simple XSS vulnerability. Having some basic knowledge of XSS will help.

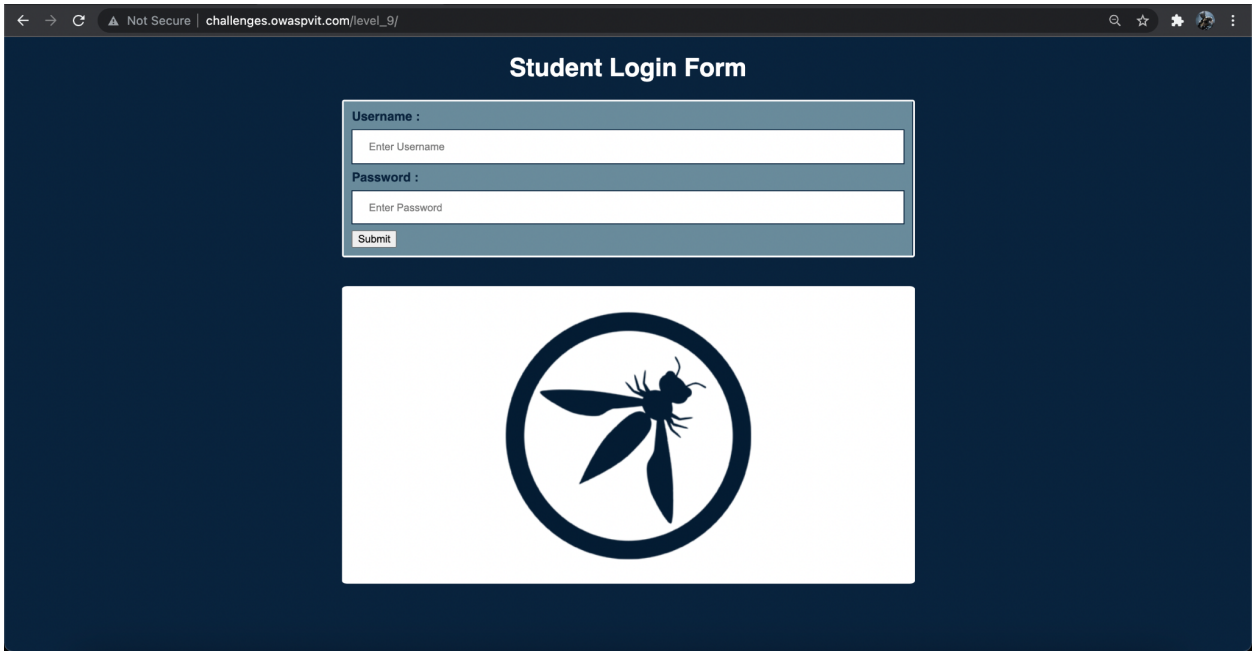
Since this level requires no filtering a simple command like - **<script>alert(0)</script>** will result in the flag.

WASP{cOnGrATuLaTioNs_Challenge8_cOmPl3ted!}sila

P.S. This is not the only method to solve this level.

Flag for Level 8:WASP{cOnGrATuLaTioNs_Challenge8_cOmPl3ted!}

LEVEL 9:

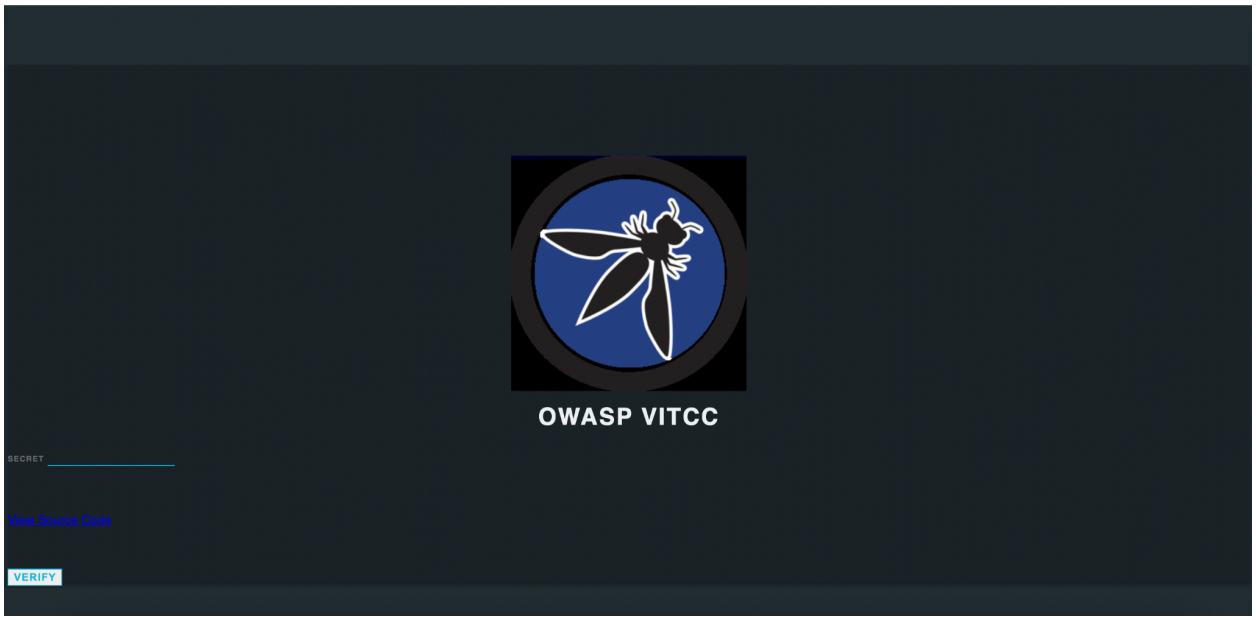


This level is dedicated to Sqli (Sql injection). To solve this level you need to have a very basic understanding of the same. Googling a little of sqli will make it easier.

Here the username should be - **' or '1' = '1' #**
And you can type anything in the password field.
Submitting it will result in the flag.

Flag for Level 9: WASP{level_9_flag}

LEVEL 10:



In the view-source code link just above the verify button we get a useful piece of information -

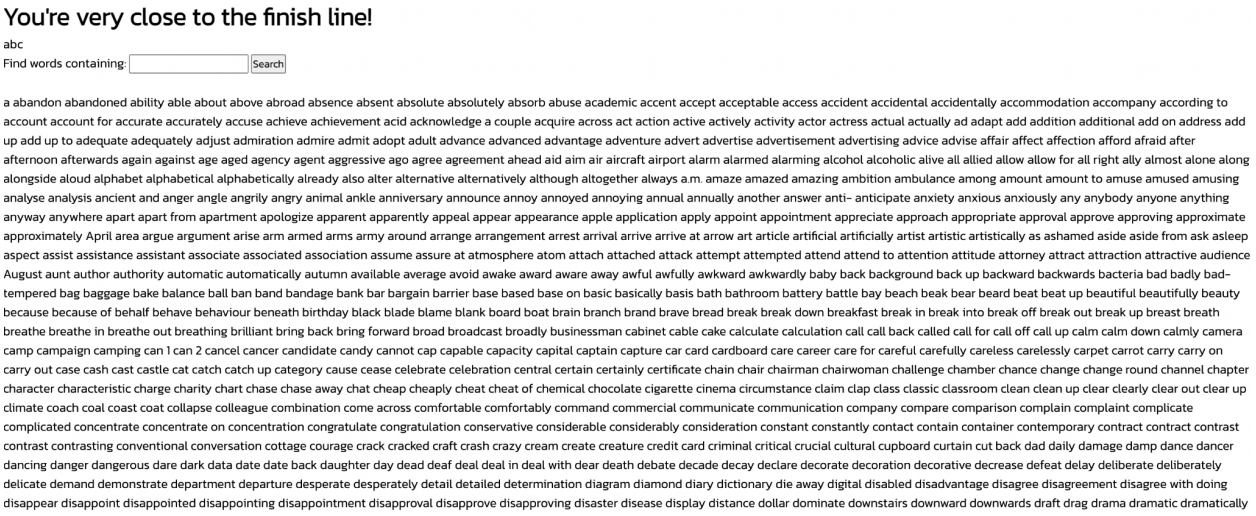
```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<
<body>
<h1>natas6</h1>
<div id="content">

<?

include "/secret.txt";

    if(array_key_exists("submit", $_POST)) {
        if($secret == $_POST['secret']) {
        print "Access granted. The password is <censored>";
    } else {
        print "Wrong secret";
    }
    }
?>

<form method=post>
Input secret: <input name=secret><br>
<input type=submit name=submit>
</form>

<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
```

There is a directory called /secret.txt. Redirecting the webpage to http://challenges.owaspvit.com/level_10/secret.txt we find a text, which when submitted in the secret field in the index page, we get the flag.

Flag for level 10: WASP{CTF_256}

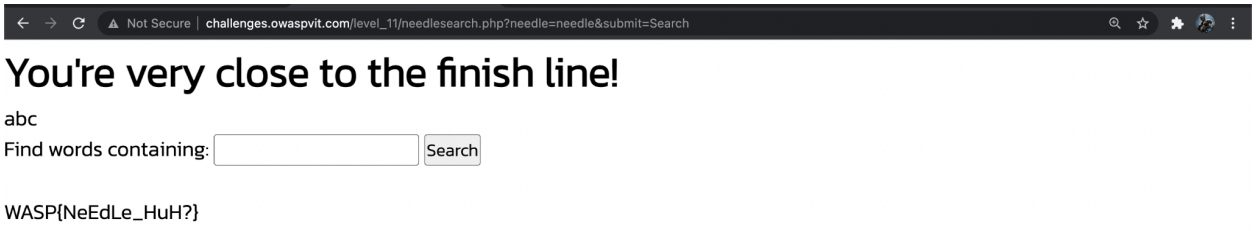
LEVEL 11:

Here typing random words as “flag” or just letters are providing us with a bunch of random words.



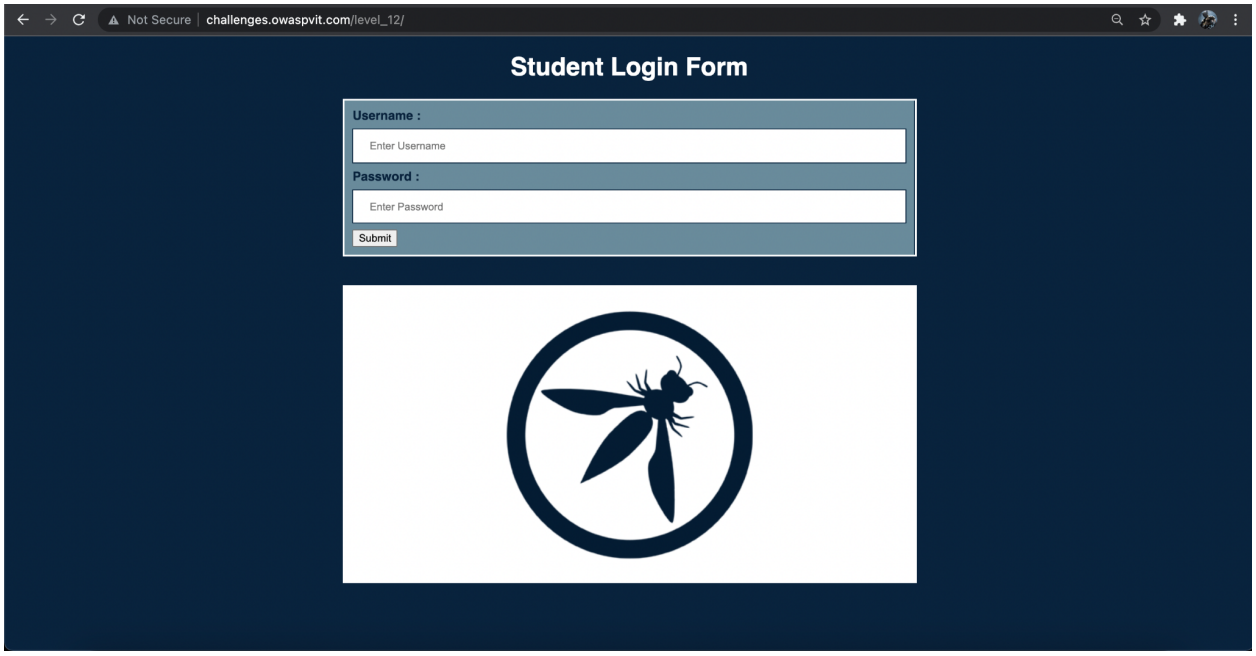
But instead if we look closely at the URL, we can see that this is not named index.php as usual, but needlesearch.php

Simply typing “needle” in the search box yields us the flag.



Flag for level 11: WASP{NeEdLe_HuH?}

LEVEL 12



This page is very similar to Level 9. It also focuses on Sqli, *but* with filtering.

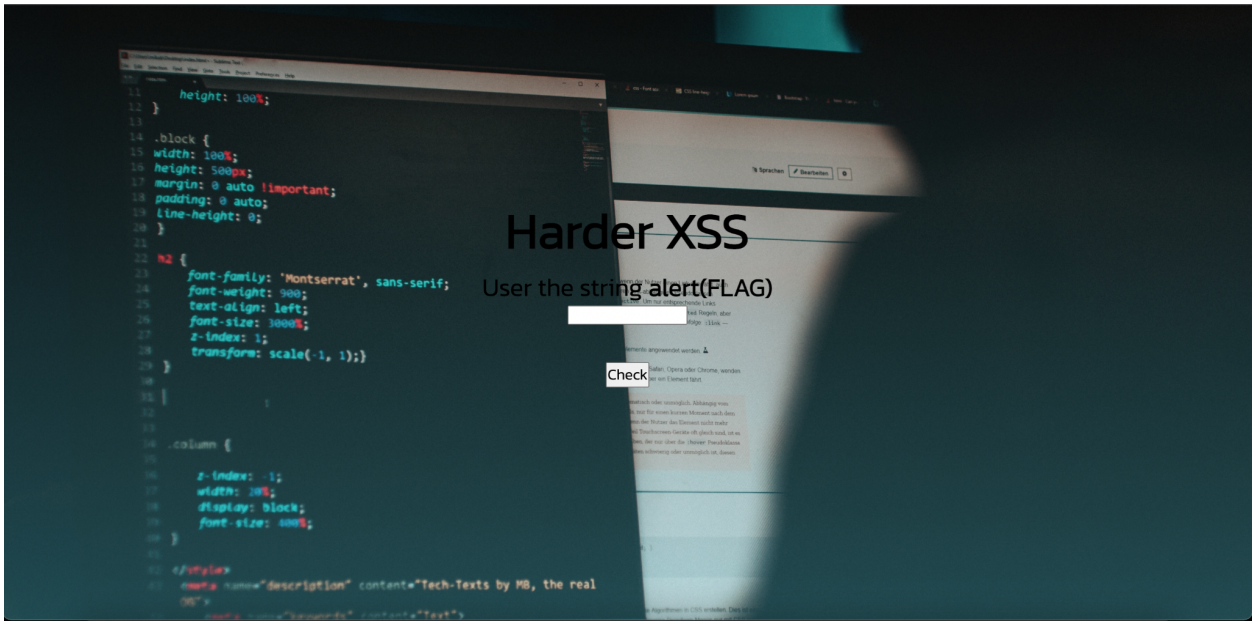
The username could be : **'or1'='1'/**/union/**/Select/**/flag,1/**/from/**/owasp#**
And you can type anything in the password field.
Submitting it will result in the flag.

admin auwdjawoidjawiodj
WASP{You_ReAcheD_LeveL_13} 1

P.S. This is not the only method to solve this level.

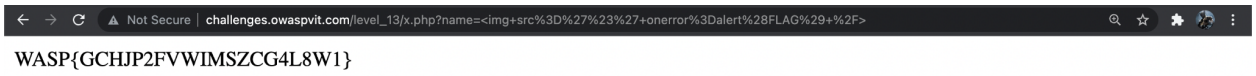
Flag for Level 12: WASP{You_ReAcheD_LeveL_13}

LEVEL 13:



As it suggests this is the harder version of level 8 - XSS with filtering.

Something like : **** will result in the flag for the level.



WASP{GCHJP2FVWIMSZCG4L8W1}

P.S. Even if this is not the only method to solve the level, the usage of alert(FLAG) is mandatory as mentioned in the level.

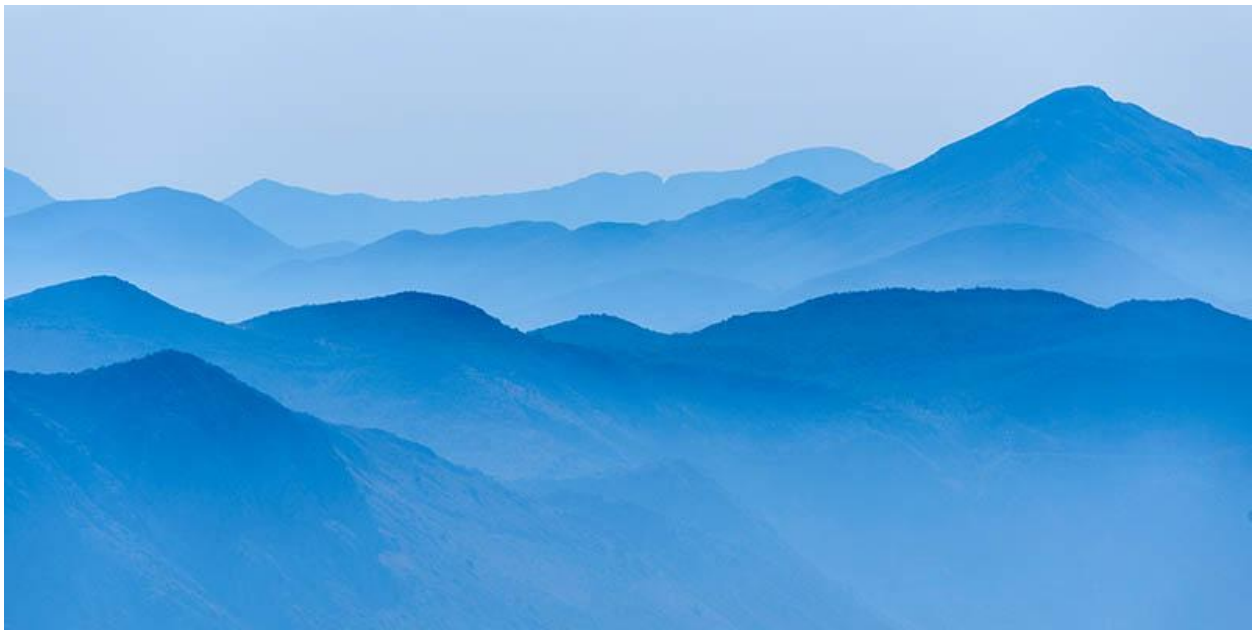
Flag for Level 13: WASP{GCHJP2FVWIMSZCG4L8W1}

LEVEL 14: (OSINT)

Find the name of Noelle's sister's dog, and wrap it in WASP{} in small letters to get the flag.

FORMAT: WASP{dog name}

P.S. Noelle's family only uses twitter

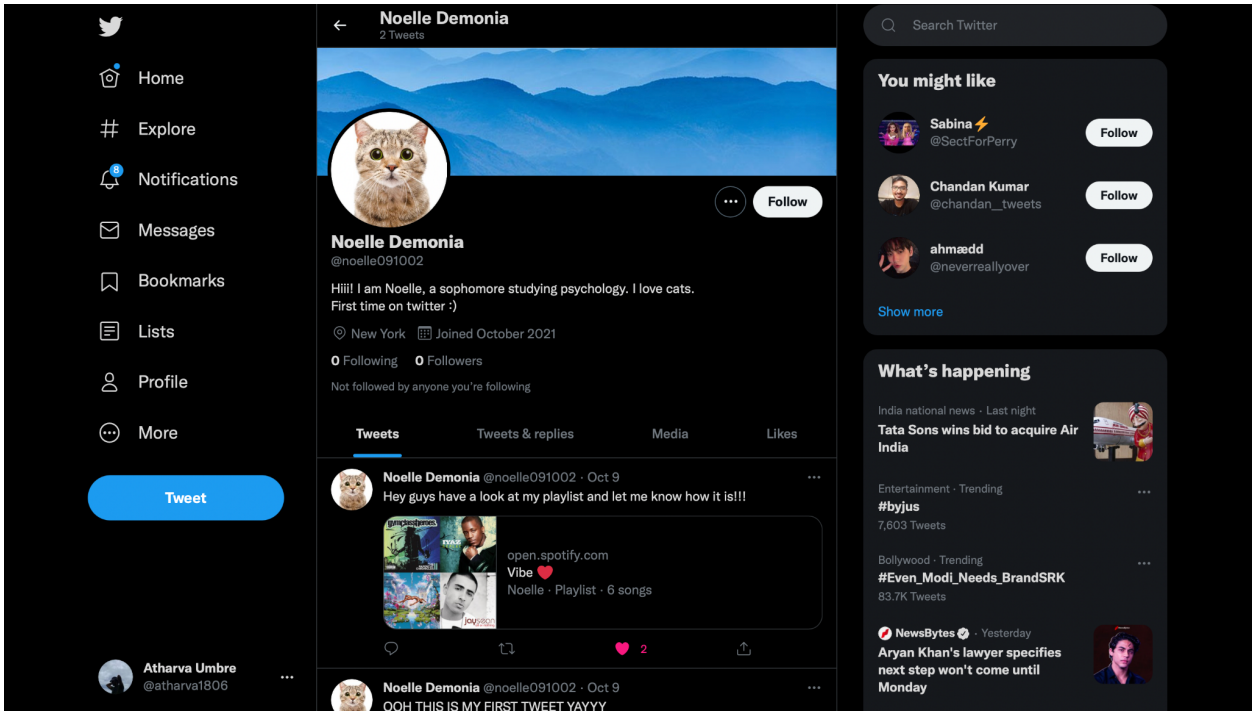


So this is an OSINT challenge. Basically meaning we have to search through google and social media to get information.
Looking at the question we can formulate a plan. First we have to find information about noelle. Then about her sister. And finally about her dog.

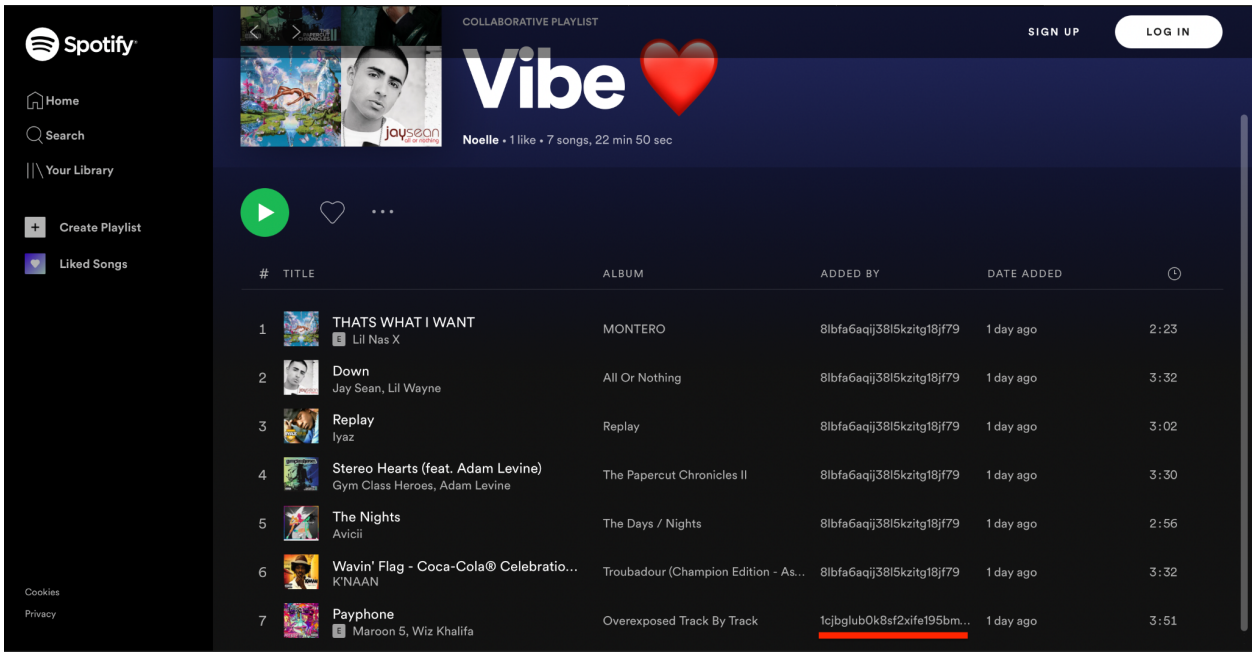
The image provided to us stores some useful information in its metadata. Using exiftool, we can see through its contents.

Directory	/tmp
File Size	23 KiB
File Modification Date/Time	2021:10:10 05:55:15+00:00
File Access Date/Time	2021:10:10 05:55:15+00:00
File Inode Change Date/Time	2021:10:10 05:55:15+00:00
File Permissions	-rw-------
File Type	JPEG
File Type Extension	jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	inches
X Resolution	72
Y Resolution	72
XMP Toolkit	Image::ExifTool 12.16
Creator	noelle091002
Image Width	800
Image Height	400
Encoding Process	Progressive DCT, Huffman coding
Bits Per Sample	8
Color Components	3

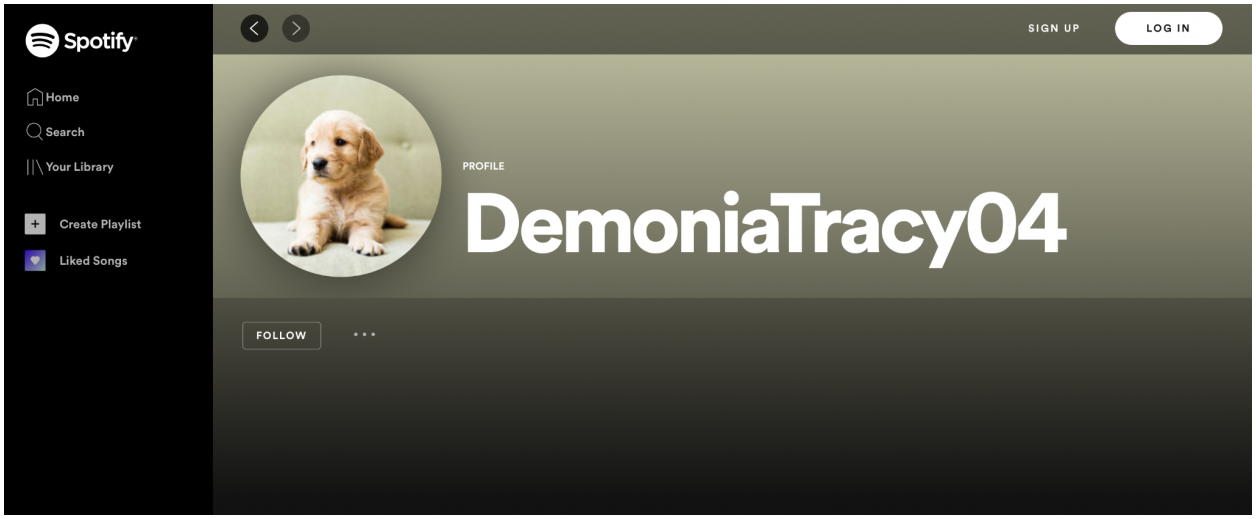
We see a username. We were told in the question that Noelle's family uses only twitter. So searching this username in twitter yields us Noelle's twitter profile.



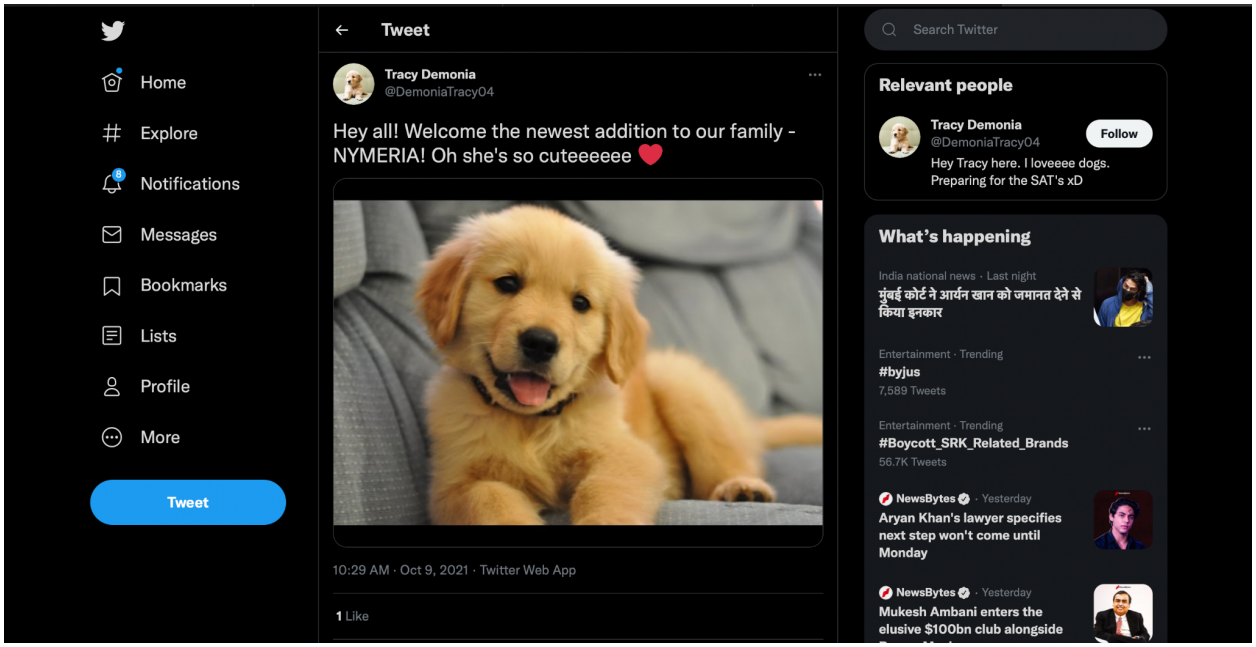
We aren't able to find anything which could help us directly in her profile. But she has shared a spotify playlist link. So let's check that out.



Here we see that it is a collaborative playlist indicating that people other than Noelle have contributed to this. And if we take a look at the “Added by” column, we can see that surely the last song has been added by another user. Taking a look at that user’s profile -



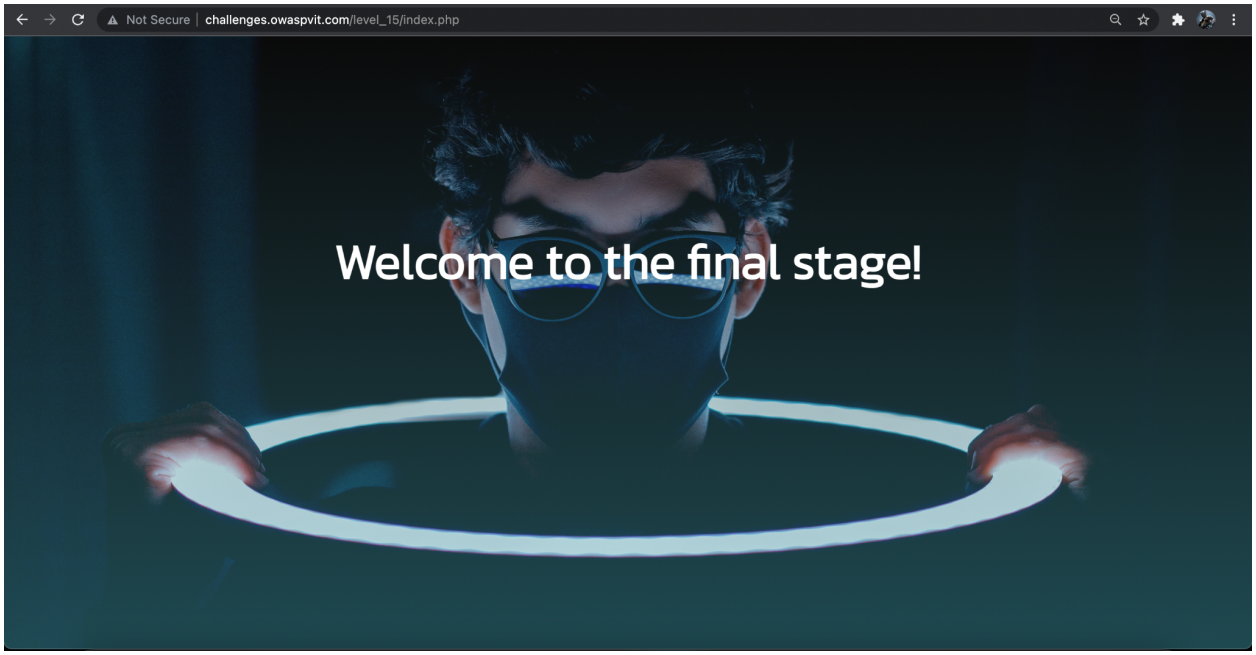
She shares the same surname as Noelle, and she also has a dog as her pfp, so it's safe to assume that she may be her sister. Searching this exact username on twitter we find -



And voila! We found her dog's name.

Flag for level 14: WASP{nymeria}

LEVEL 15:



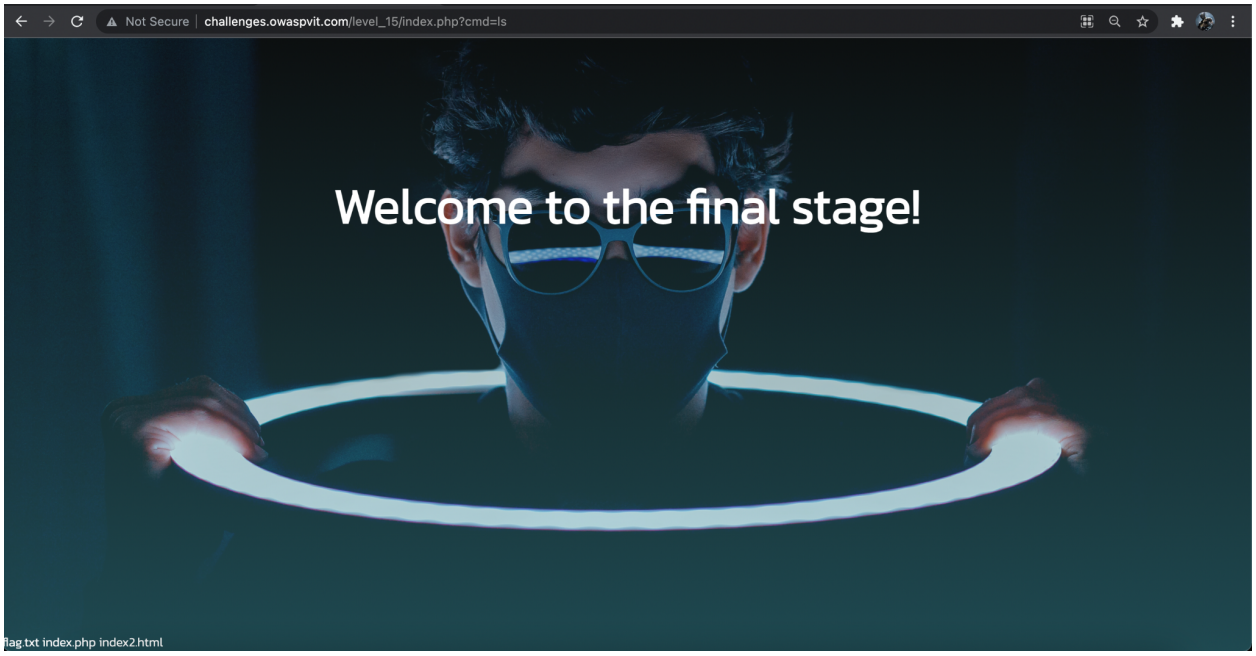
We have reached the final stage. Snooping around in the view page source we find another base64 encoded message - <!--
Q29tbWFuZCBpbmplY3Rpb24gdXNpbmcgY21kIGlzIGEgZnVuIHRoaW5nIHRvIGRvIQ==
-->

Decoding this we get a clue to solving this round - Command injection using cmd is a fun thing to do.

We can see that level 15 has an index.php page. To learn more about this we can search command injection in php url.

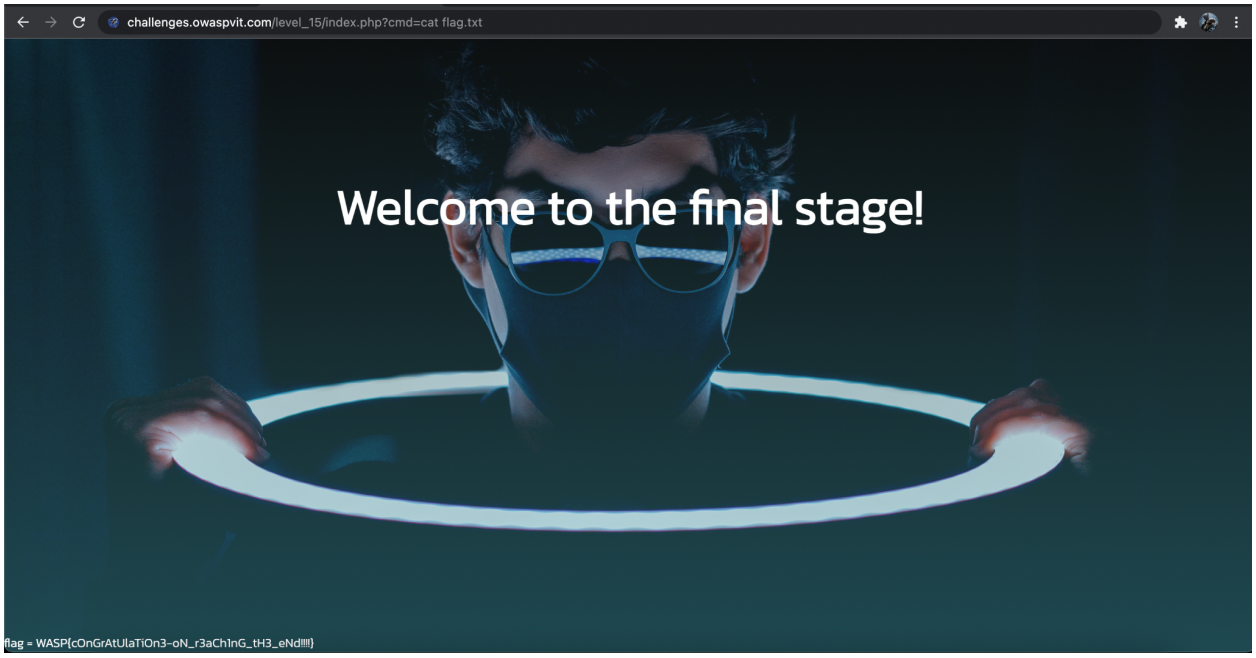
To solve this we will add - ***?cmd=ls*** to the url. So the new url looks like - http://challenges.owaspvit.com/level_15/index.php?cmd=ls

We have injected the command ***ls*** in the url, which lists all directories and files stored in this level.



At the bottom of the page we see that this level contains an index.php, and index2.html file, and a flag.txt.

To open the flag.txt we will use the ***cat*** command. Injecting ?cmd=cat flag.txt in the url - (new url - https://challenges.owaspvit.com/level_15/index.php?cmd=cat%20flag.txt)



We get the flag!

Flag for level 15: flag = WASP{cOnGrAtUlaTiOn3-oN_r3aCh1nG_tH3_eNd!!!!}
