TIGER KING CTF 2020 WRITEUP (DERPCON 2020)

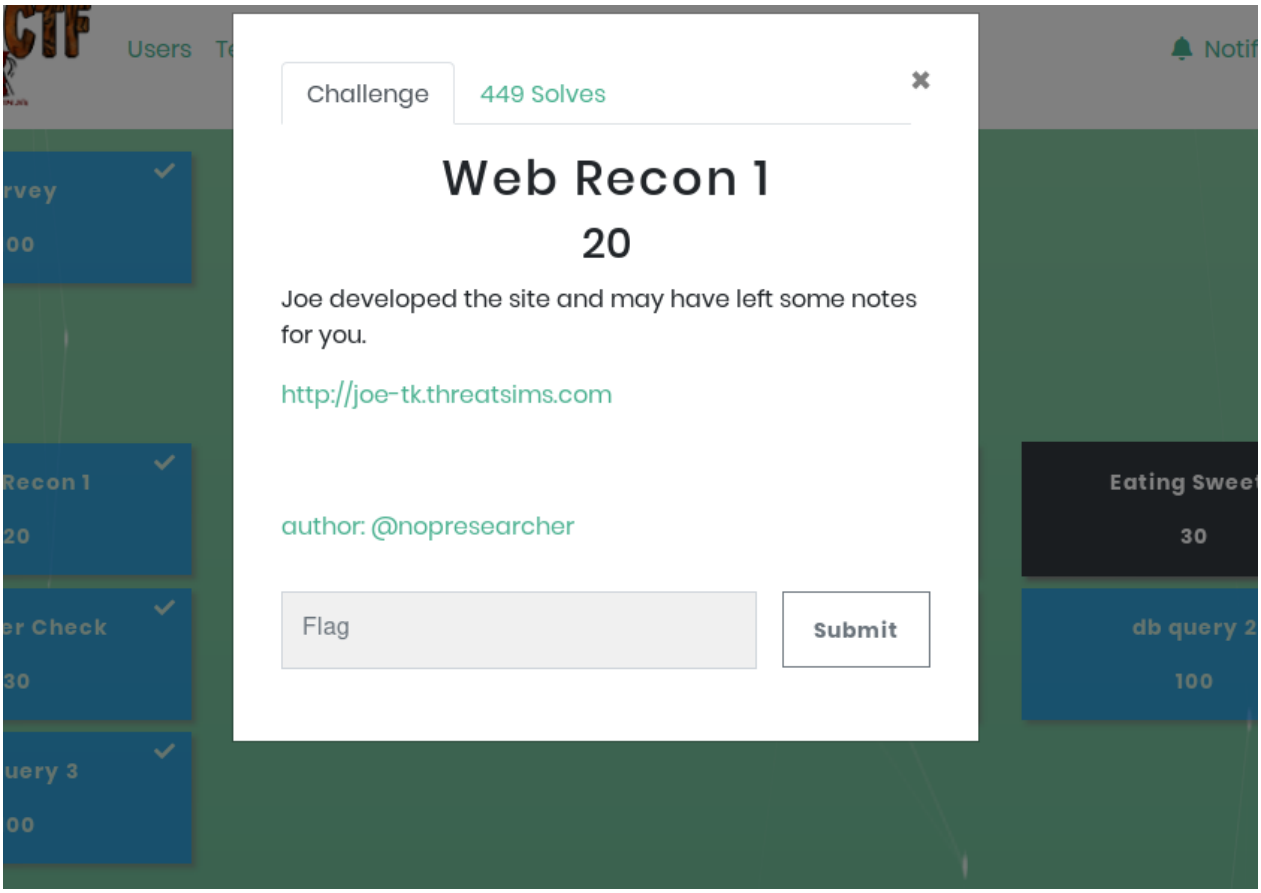
by CoronHack (Bl4ckEnergy)

Tiger King CTF 2020 (organized on May 1, 2020 online) (https://tigerking.threatsims.com/) is a CTF orginized by Threat Simulations and RunCode.ninja during Derpcon 2020. We participated under the team name **CoronHack**. We now have a team name on CTFTime which is **Bl4ckEnergy**.

The challenges consisted of **web exploitation, cryptography, forensics, Linux system exploitation** and others challenges whose names were little more personalized (**RunCode.ninja, Trainer, Survey**) which are among others web challenges in general.

Tiger King

- **Web recon 1**

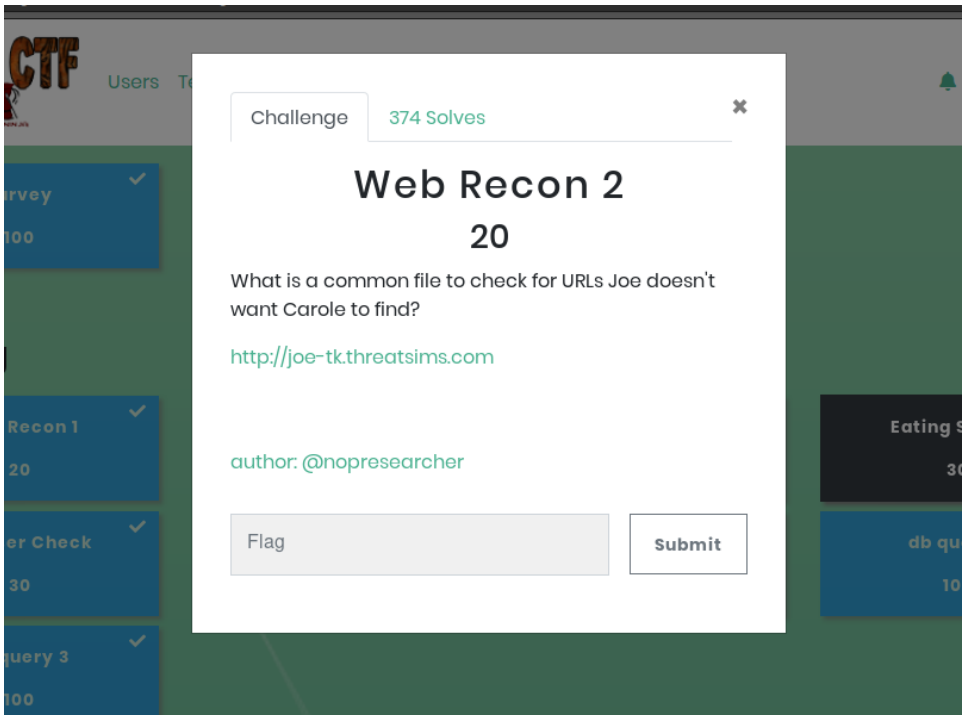


In this challenge, when we go to the indicated link and we display the source code, we see the flag in HTML comment.

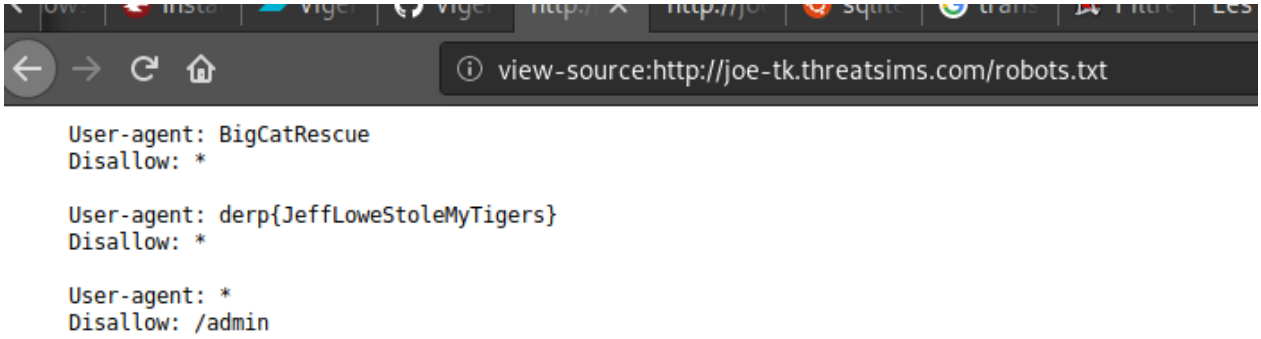
```
            <li><a href="/tigers">Tigers</a></li>
          </ul>
          <ul class="nav pull-right">
            <li><a href="/admin">Admin</a></li>
            <li class="divider-vertical"></li>
            </li>
          </ul>
        </div><!-- /.nav-collapse -->
      </div>
    </div><!-- /navbar-inner -->
  </div><!-- /navbar -->


<img src="static/images/ie_logo.gif" alt="internet explorer">
<br>
<img src="static/images/gwlogo.jpg" alt="tiger king">
<!-- I don't know how many times I have to say it! derp{CaroleBaskinDidIt} -->




 </body>
 <br>
 <br>
 <br>
 <br>
 <br>
 <br>
```

The flag is : **DERP{CaroleBaskinDidIt}**

➢ Web recon 2

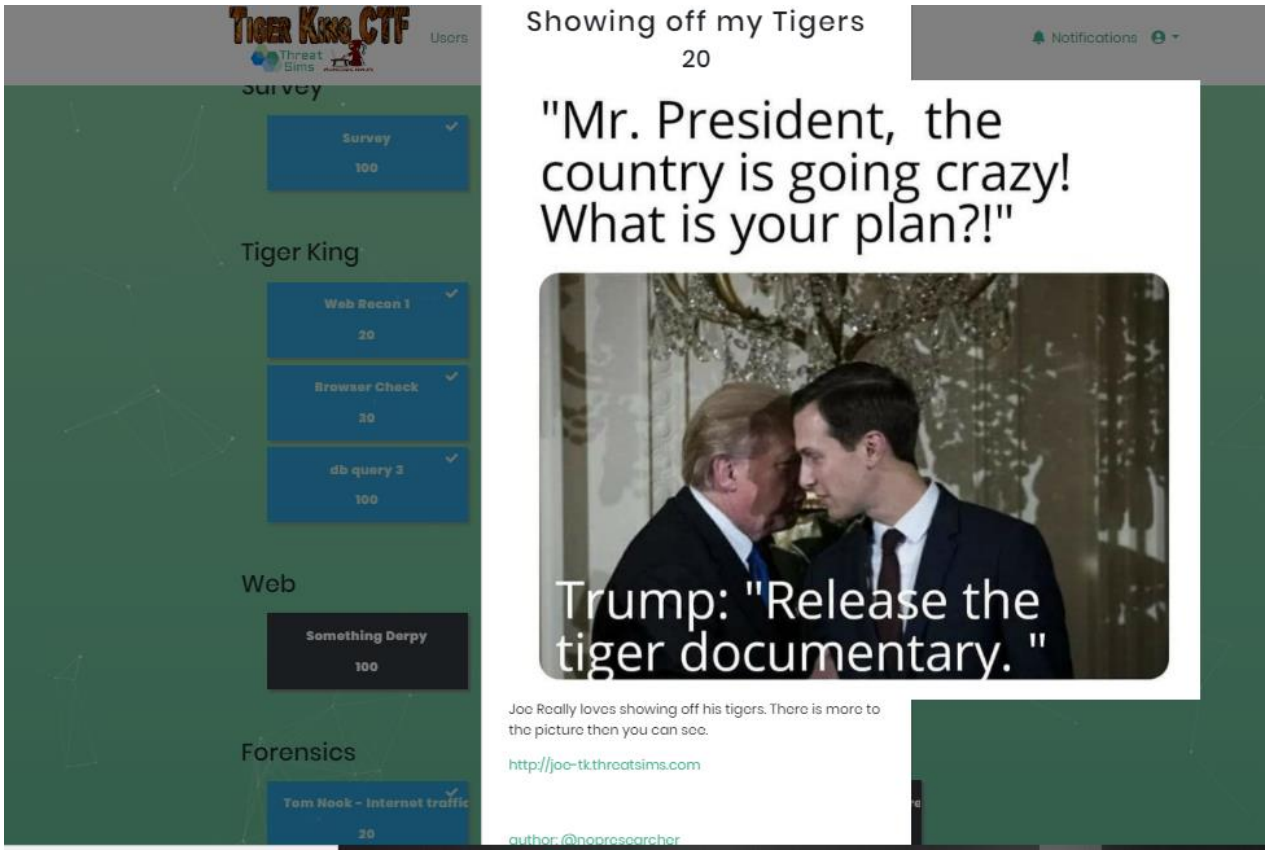


In this challenge, the announcement lets us see that there are hidden files that should not be seen and therefore for the fact in programming we configure a robots.txt file at the root to prevent the indexing of certain files by search engines and therefore an inspection of this file via the link http://joe-tk.threatsims.com/robots.txt gives us the flag.



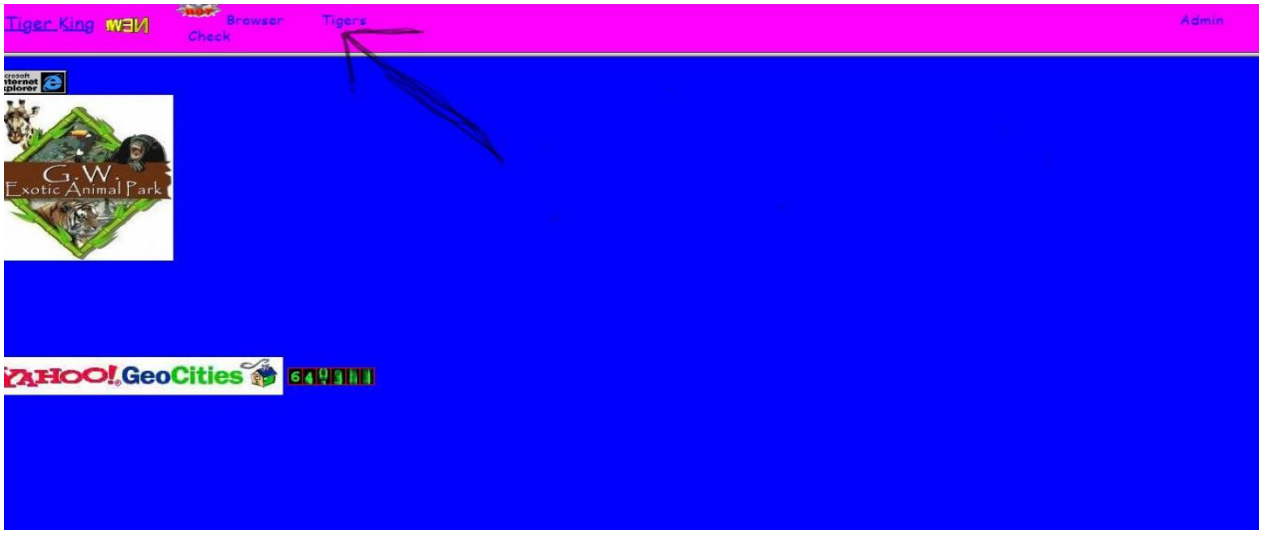
The flag is : **DERP{JeffLoweStoleMyTigers}**

- **Showing off my Tigers**



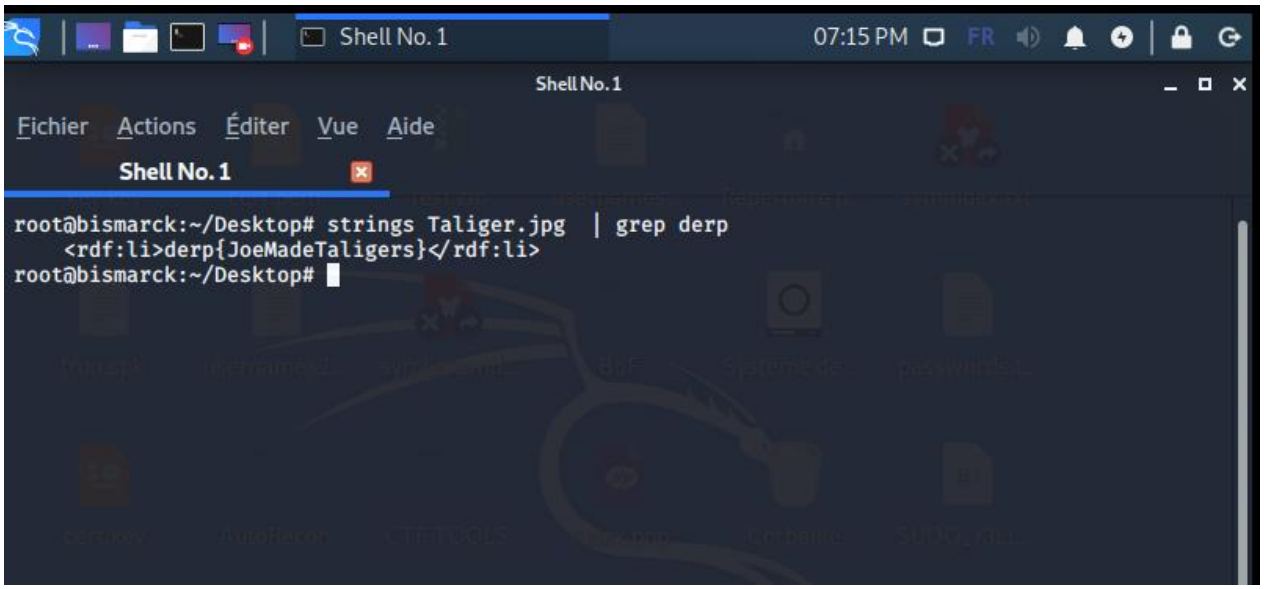
The title "**Showing off my tigers**" lets us think of "a concealment" and therefore a steganography challenge because to show means to show and the word off gives meaning to something inactive, to hide.

We therefore go to the address indicated (http://joe-tk.threatsims.com/), then in the "**Tigers**" section to download our image and try to see the content.

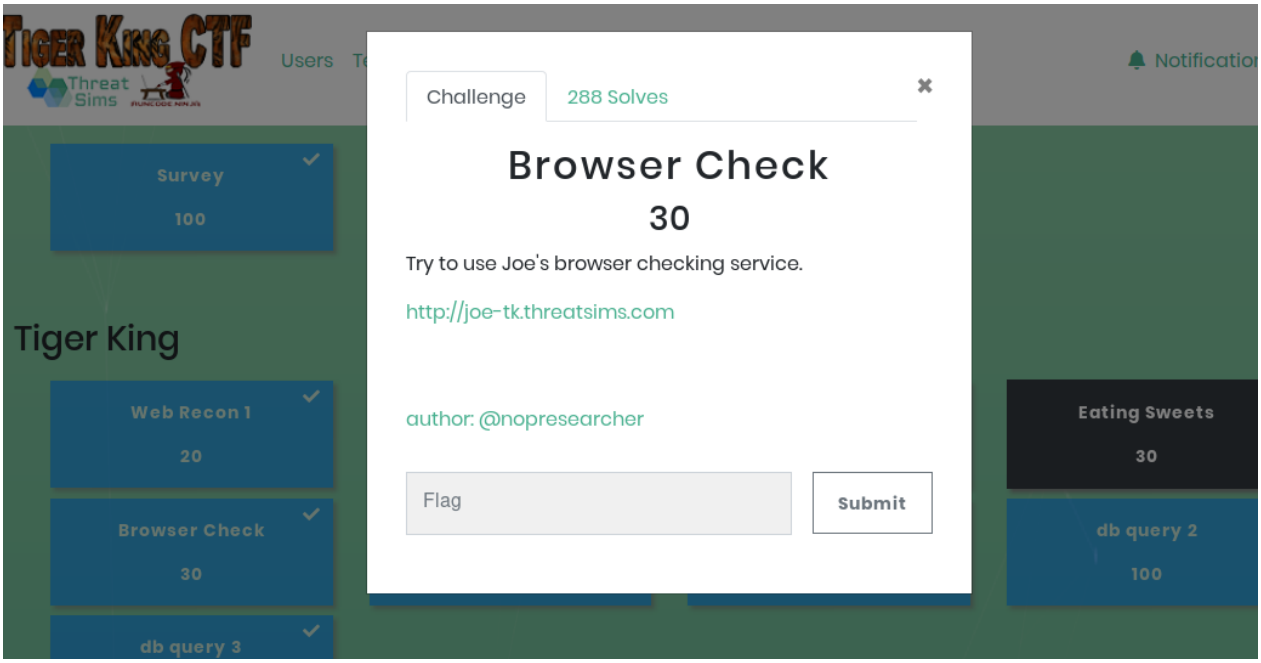


We went into Kali Linux to use the "**strings**" tool to extract all the strings from the image, which gave us a disordered string. Since the flags to be submitted start with

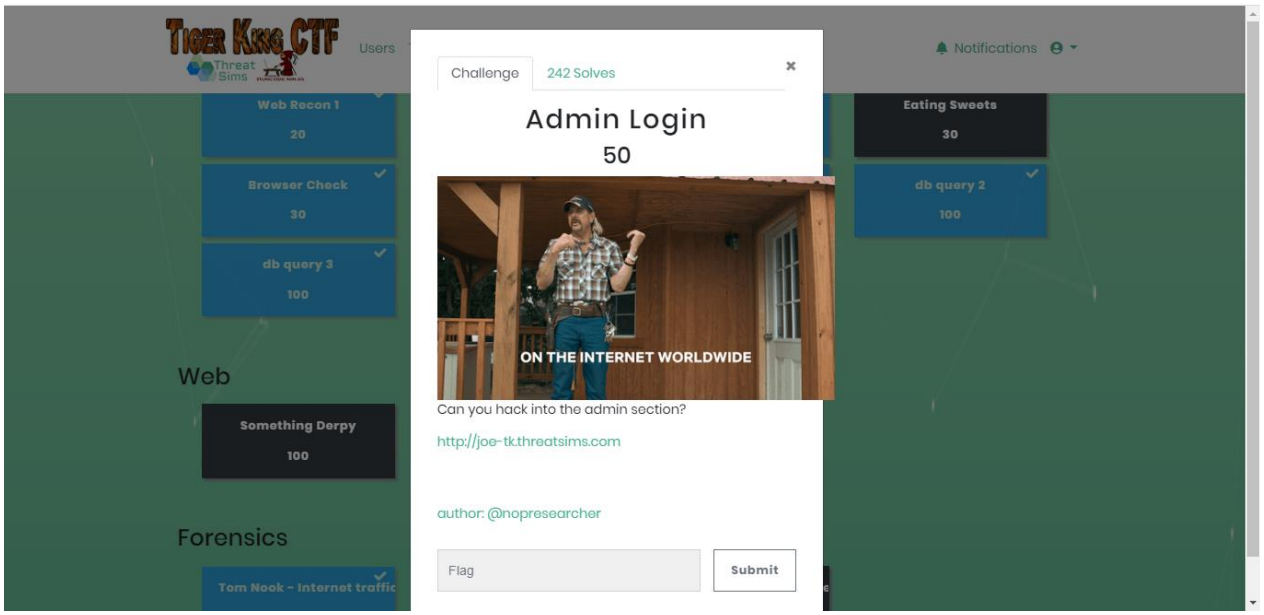
the word "**derp**", then we have filtered our chain by derp: **strings Taliger.jpg | grep derp,** which gives us our flag: **derp{JoeMadeTaligers}**.



- **Browser check**



- **Admin login**



The "Admin login" theme lets us believe that at first glance it is a SQL injection bypassing authentication.

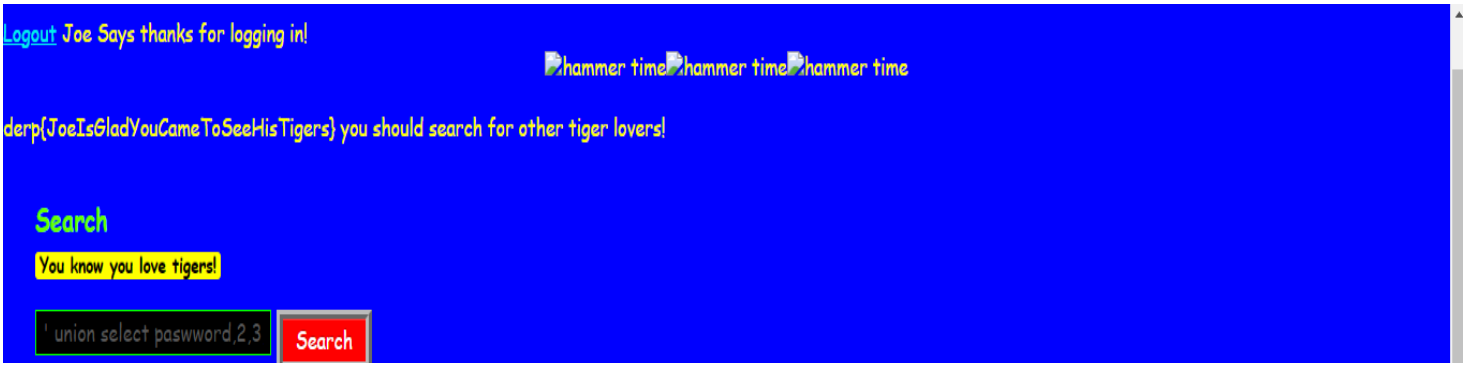
After some research on google for payloads based on bypassing authentication via SQL injection, we found the adequate payload for our form at the following

address: http://www.securityidiots.com/Web-Pentest/SQL-Injection/bypass-login-using-sql-injection.html

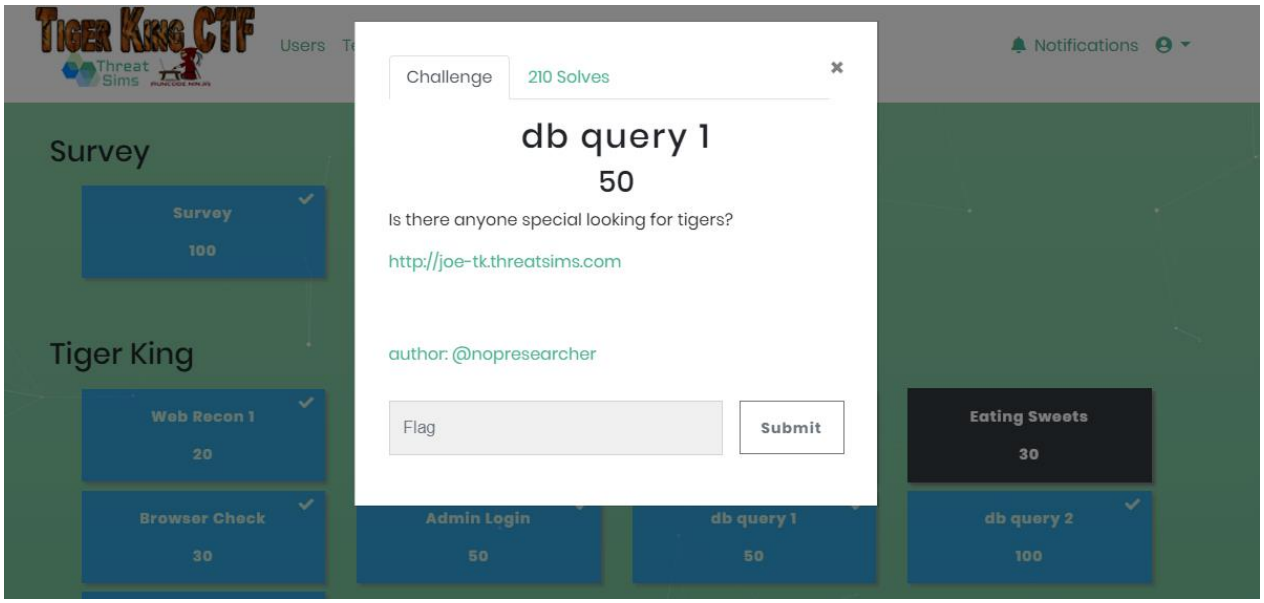
Username : ' or 1--
Password :
what we did is we left the password field empty and commented out the rest of the query. so lets try and check the Query part.

```
select username,pass from users where username='' or true--' and password='' or ''='' limit 0,1;
```

which authenticates us and brings us back to the administrator's home page with de the flag **derp{JoeIsGladYouCameToSeeHisTigers}**:



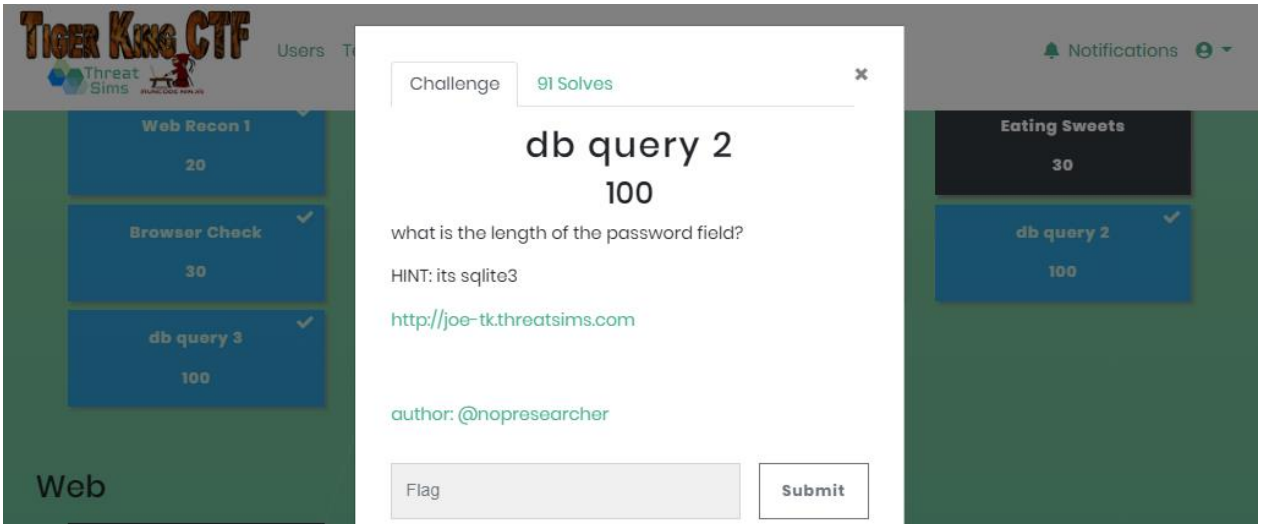
- **db query 1**



The title of the challenge lets us believe that we should make a sql request in order to display a user with a particular login. So, we used the same payload that we used to bypass the form (**login: 'or 1--**), which will allow us to display all system users and certainly our particular user.

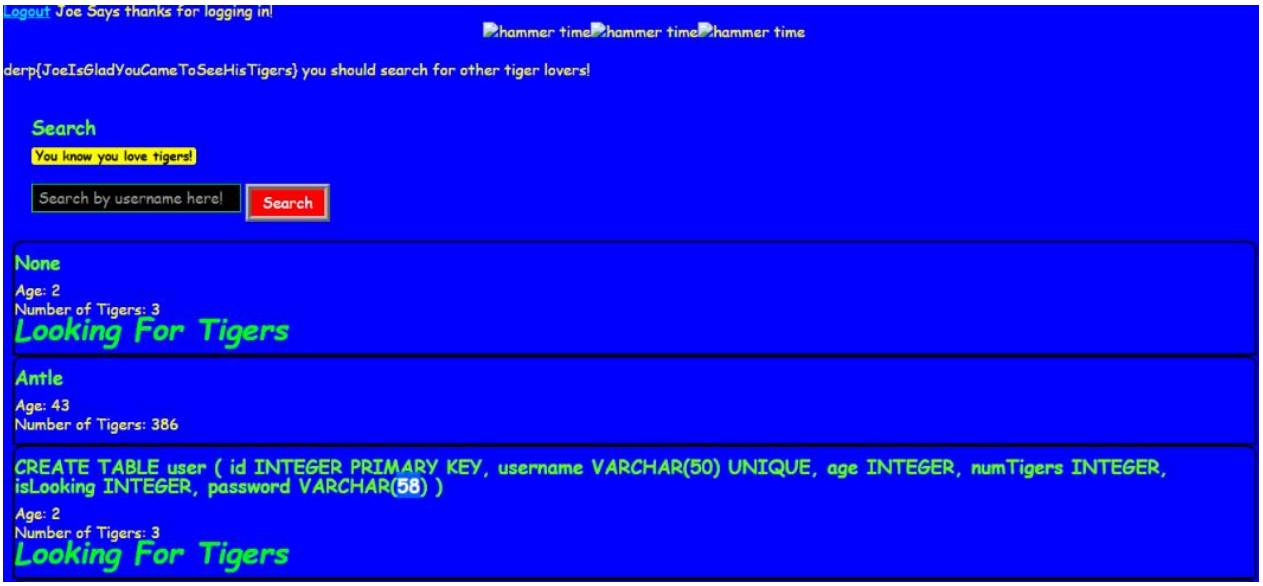
In the image above, we can see that our request allowed us to display all the users, our particular user at a login which is our desired flag (we omitted to take a picture of the result :( ).

- **db query 2**



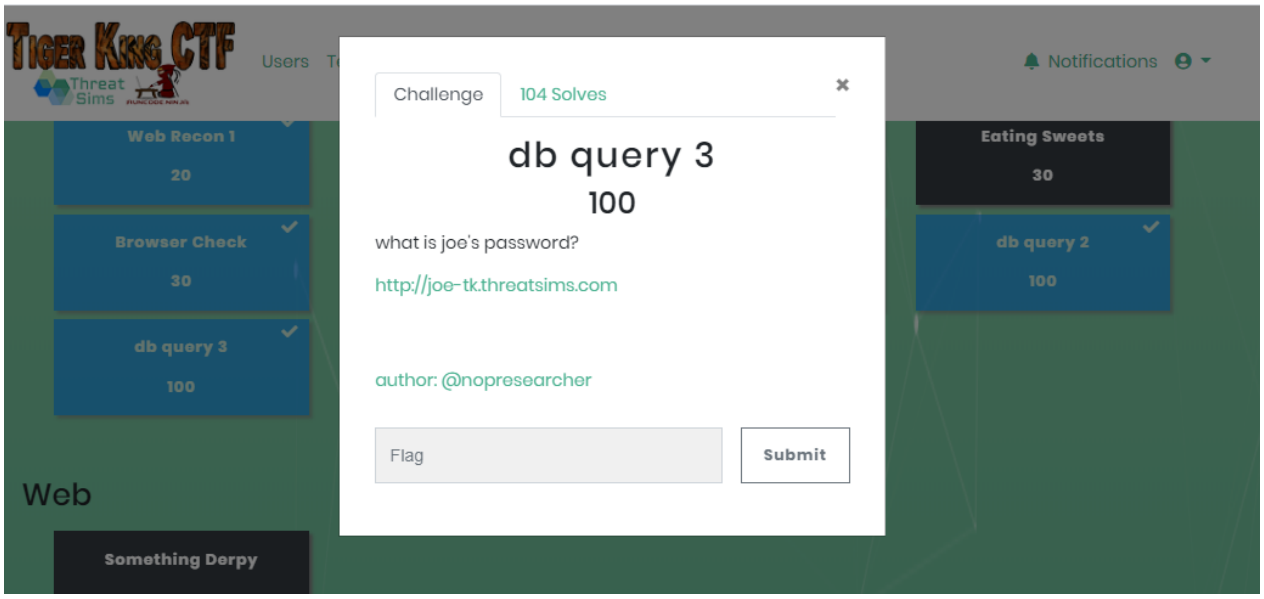
In this SQL injection challenge, we have to find the size of the password field. A hint here was given to us: sqlite3, which tells us that our SBGD works with sqlite3. So, we will have to do a sqlite3 injection.

So we will have to know the number of columns by doing: **'order by 1--** up **to' order by 4--** we have the query that runs normal but when we do **'order by 5--**, we have a message error code: **"Unknown column '5' in 'order clause'**, which means that there are no columns 5 so we have a maximum of 4 columns. We finally make this request to extract all the information from our database (in MySQL we **have INFORMATION_SCHEMA** in sqlite it is **sqlite_master**), here we have only one table which is user: **'union select sql, 2,3,4 from sqlite_master –**



We can see here that the size of the password field is **58** which is also our flag to validate this challenge.

- **db query 3**

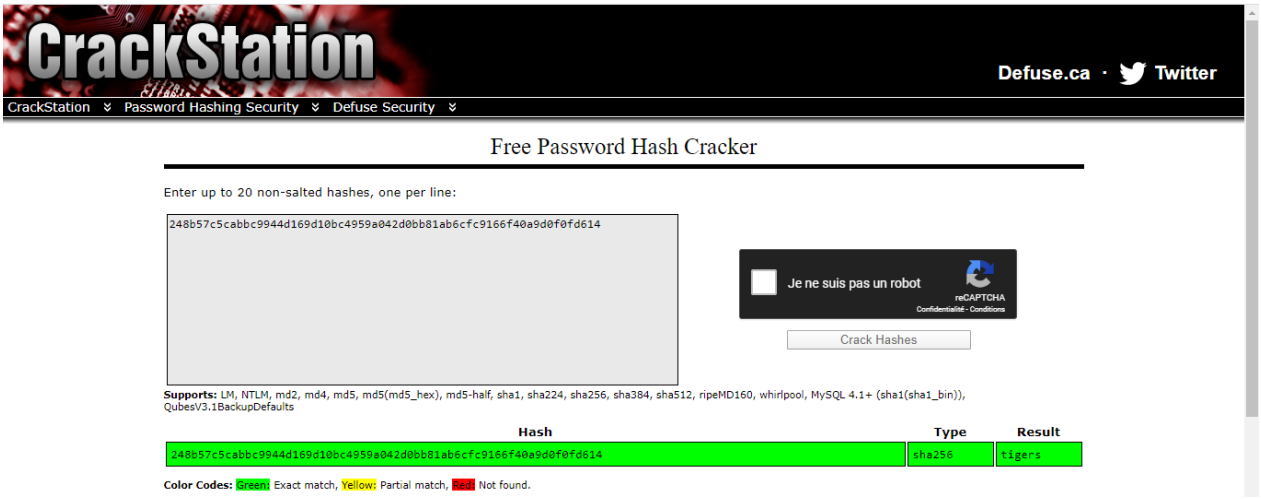


We are going to modify our payload here to recover the administrator password.

The previous challenge let us know at sqlite_master that user was the only table available. We will therefore make a request to recover Joe's password:

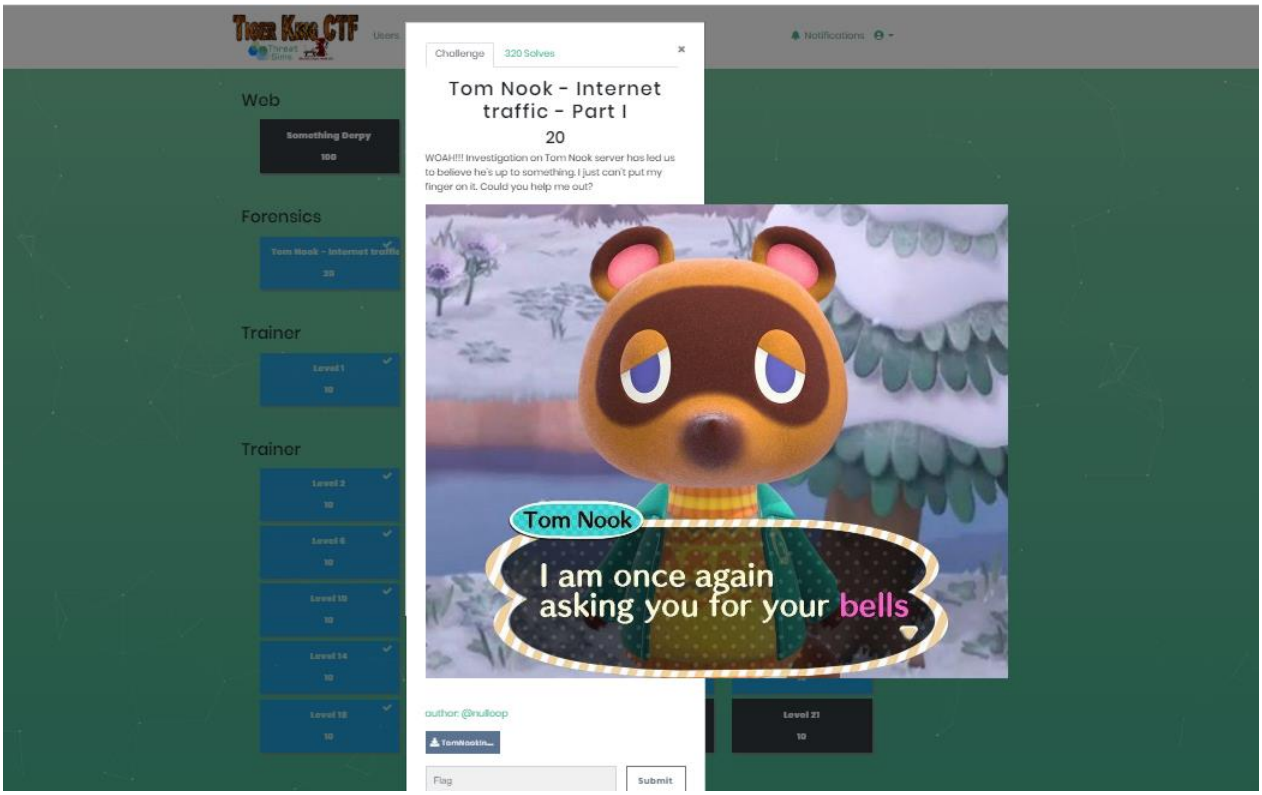
'union select 1,2, password, 4 from user --, we have the password of the users but will recover that of Joe who encrypted.

we are going to https://crackstation.net/ to decrypt the password. The password found is: **tigers** which was hashed on sha-256.

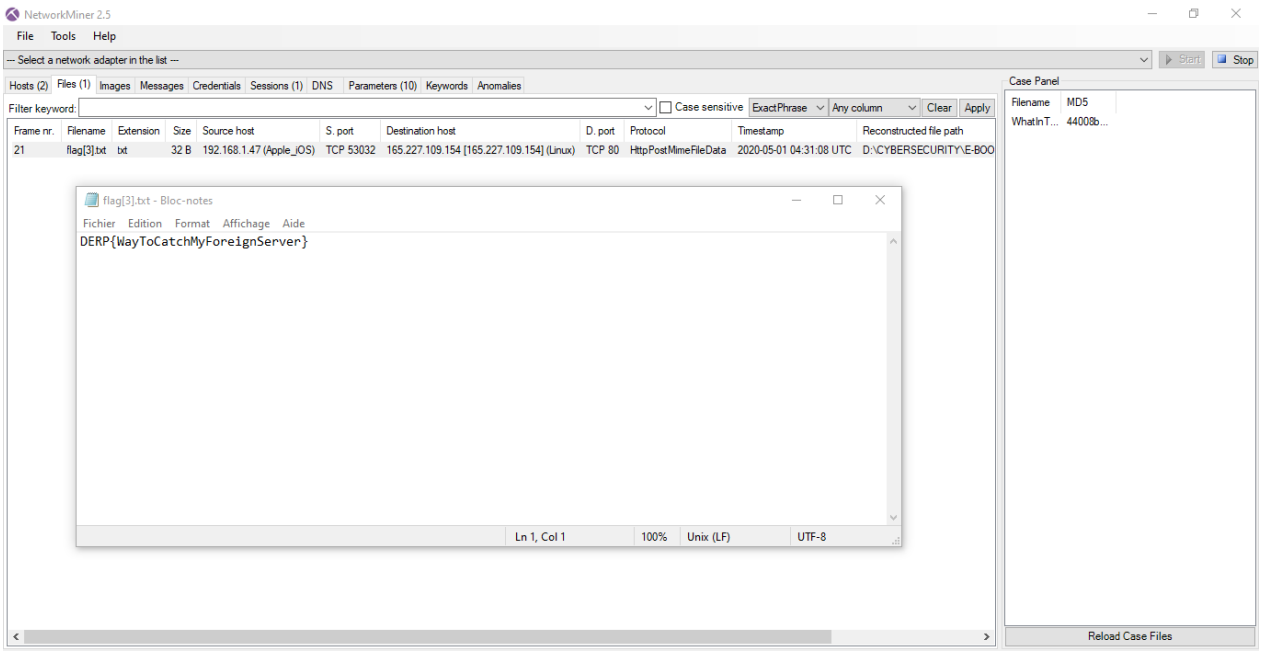


Forensics

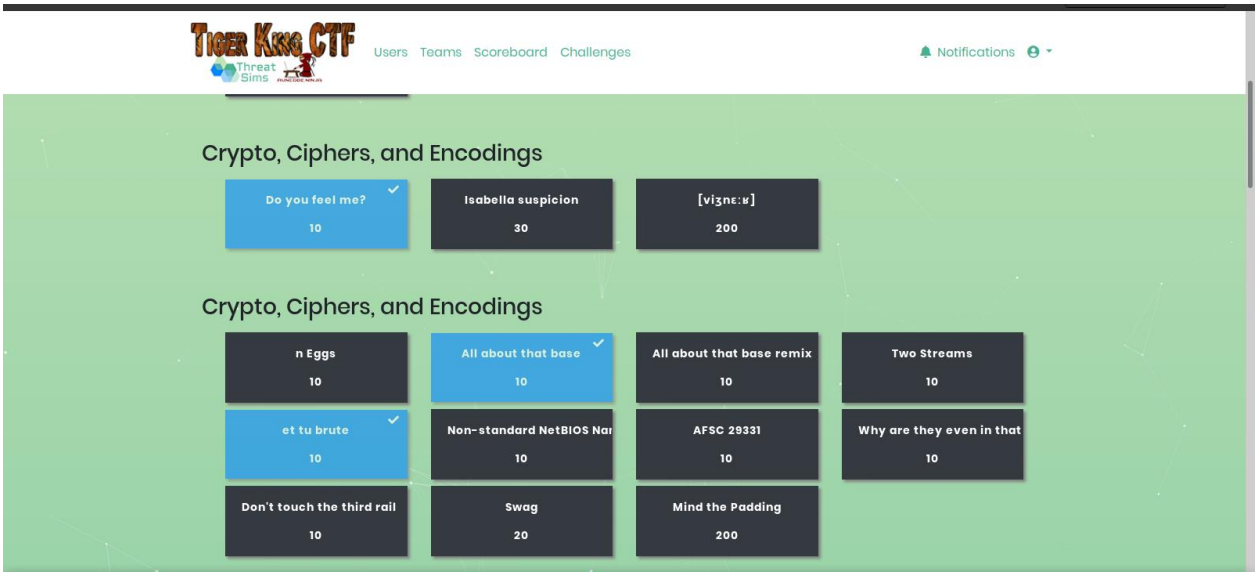
➢ Tom Nook - Internet traffic - Part I



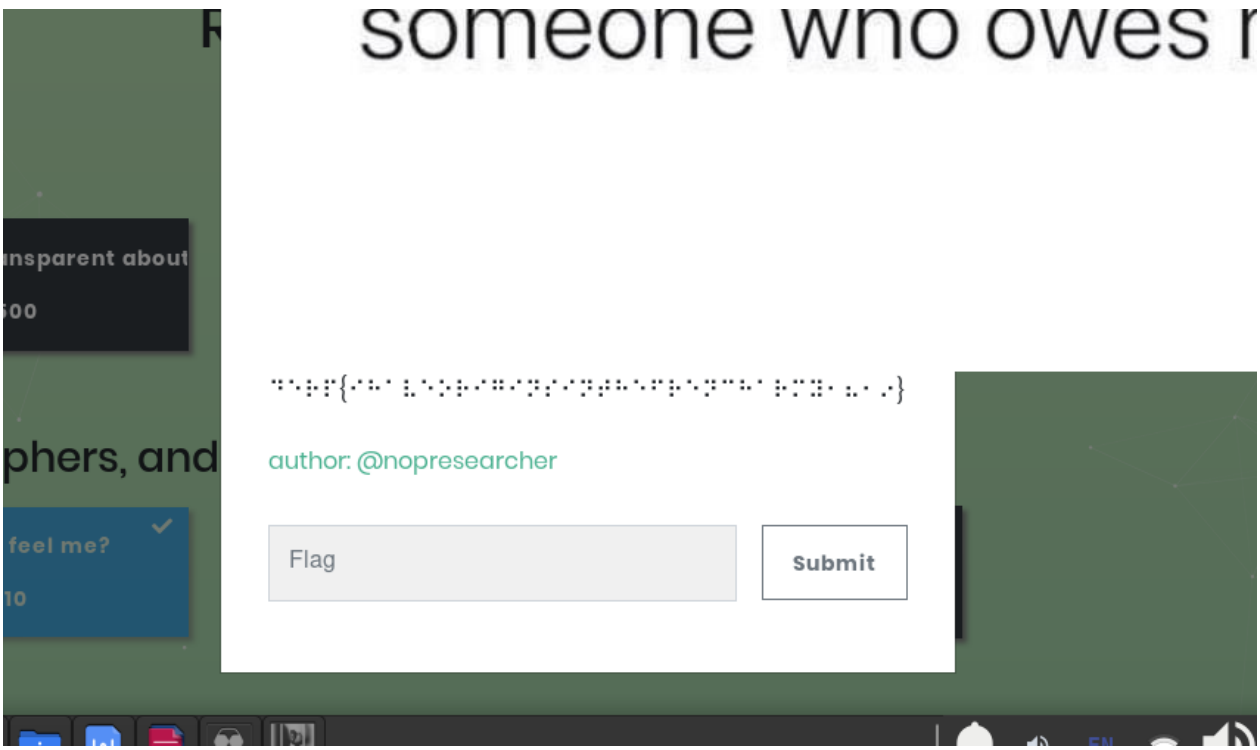
The test gives us a pcap file (**WhatInTheExfil.pcap**) to analyze. **NetworkMiner** which is an excellent tool to analyze pcap files. When we open, we see that our capture contains a file called flag.txt which is our wanted flag : **DERP{WayToCatchMyForeignServer}**.



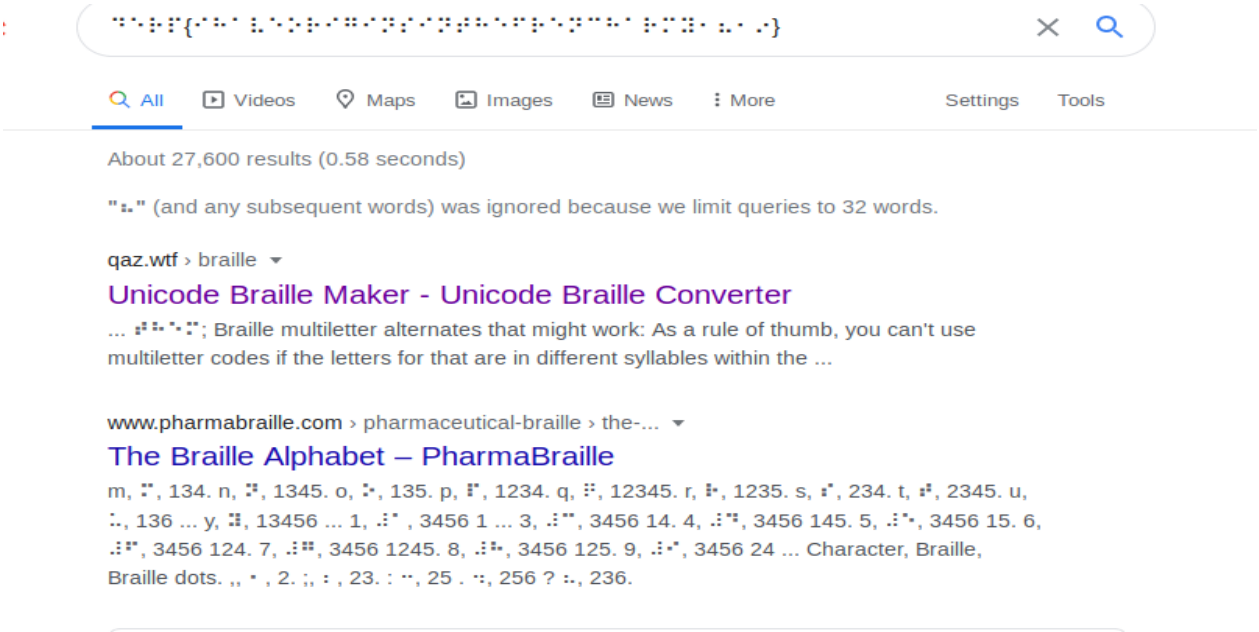
Crypto, Ciphers, and Encodings



➢ Do you feel Me



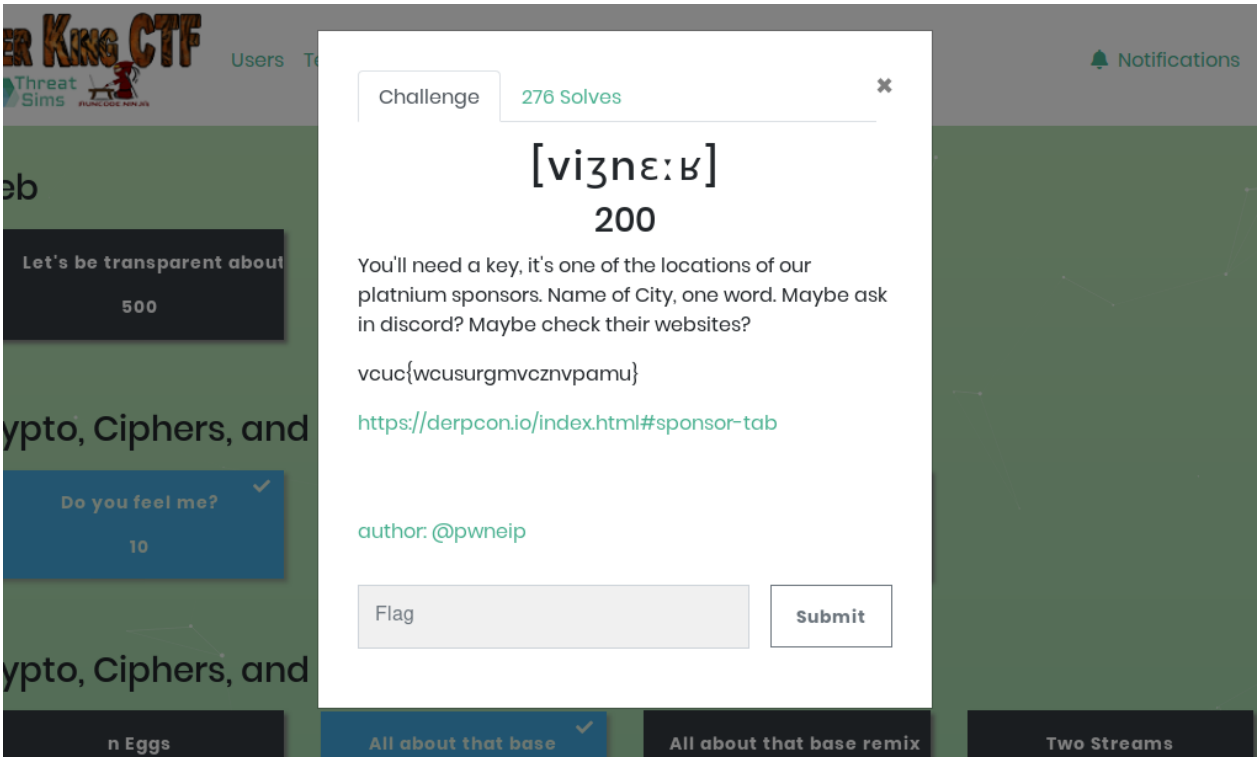
In this challenge we see a series of special characters which seems to constitute the flag and when we do a little research on googling it is clear that it is in Braille format intended for the blind.



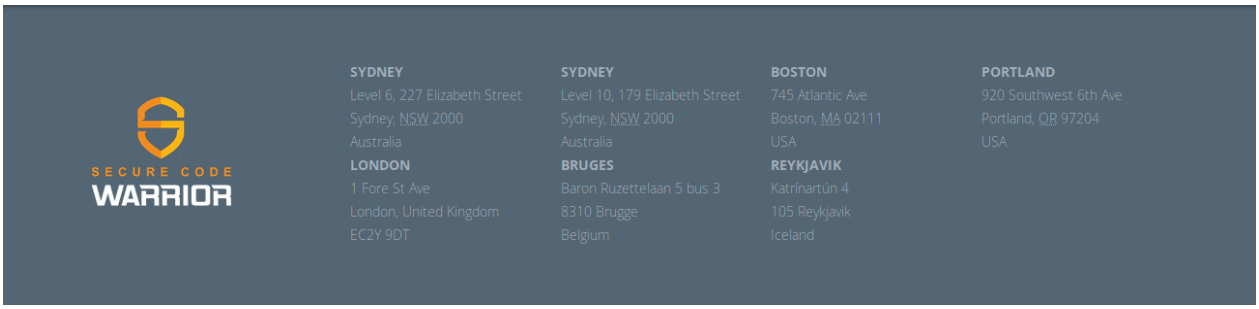
For the rest we tried to decrypt this format using this link https: /www.dcodefr/braille-alphabet and we had as a result

The flag is : **DERP{IHAVEORIGINSINTHEFRENCHARMY}**

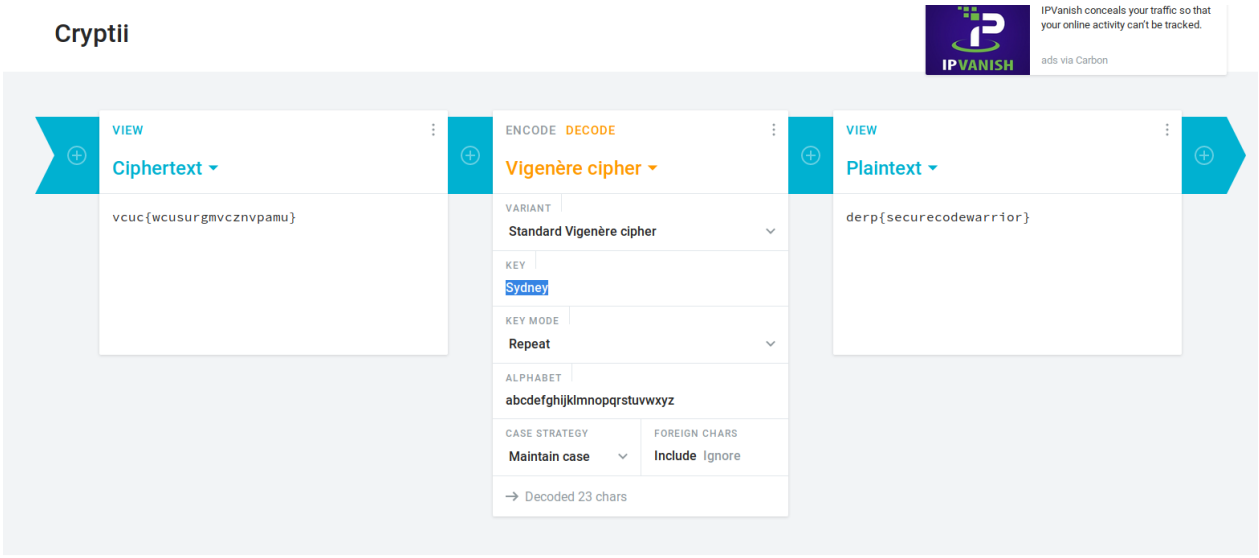
➢ [VIGNERE]



In this challenge based on the announcement, we see that it is an encoding based on the viginere and therefore a key is required to decrypt it as an index, we are informed that the key to decipher the flag and the name of the city or its siue a Platinum partner. After inspection it is from the partner secure code warrior and its location was the city of Sydney because it was repeated twice on its site https://securecodewarrior.com/

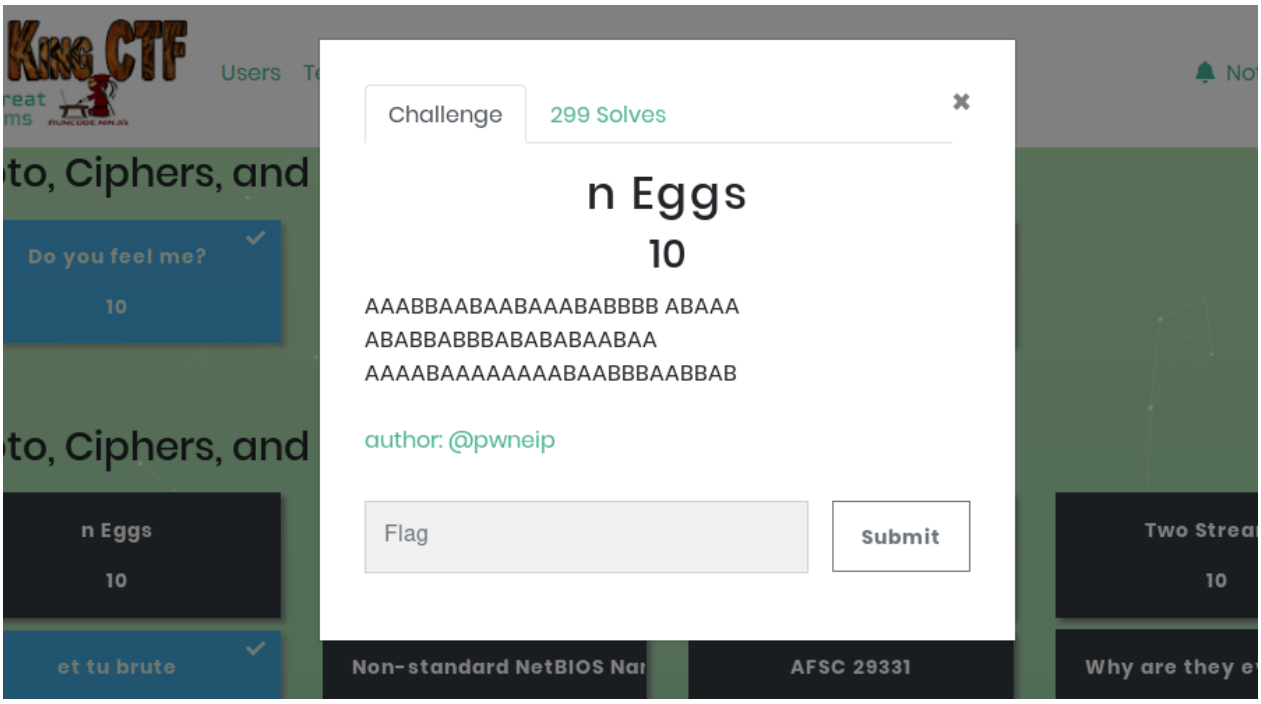


And so to decipher the flag we use the site https://cryptii.com/ using the viginer chiper option and we get

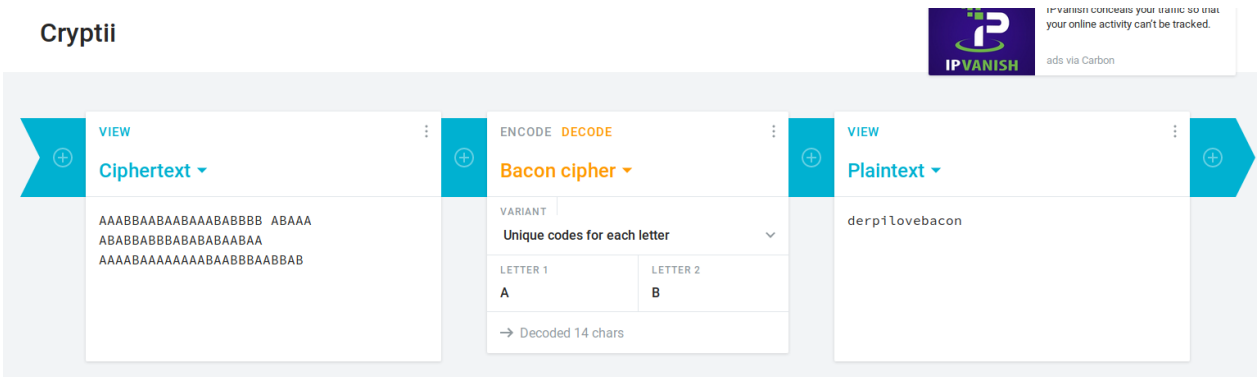


The flag is : **derp{securecodewarrior}**

➢ n Eggs

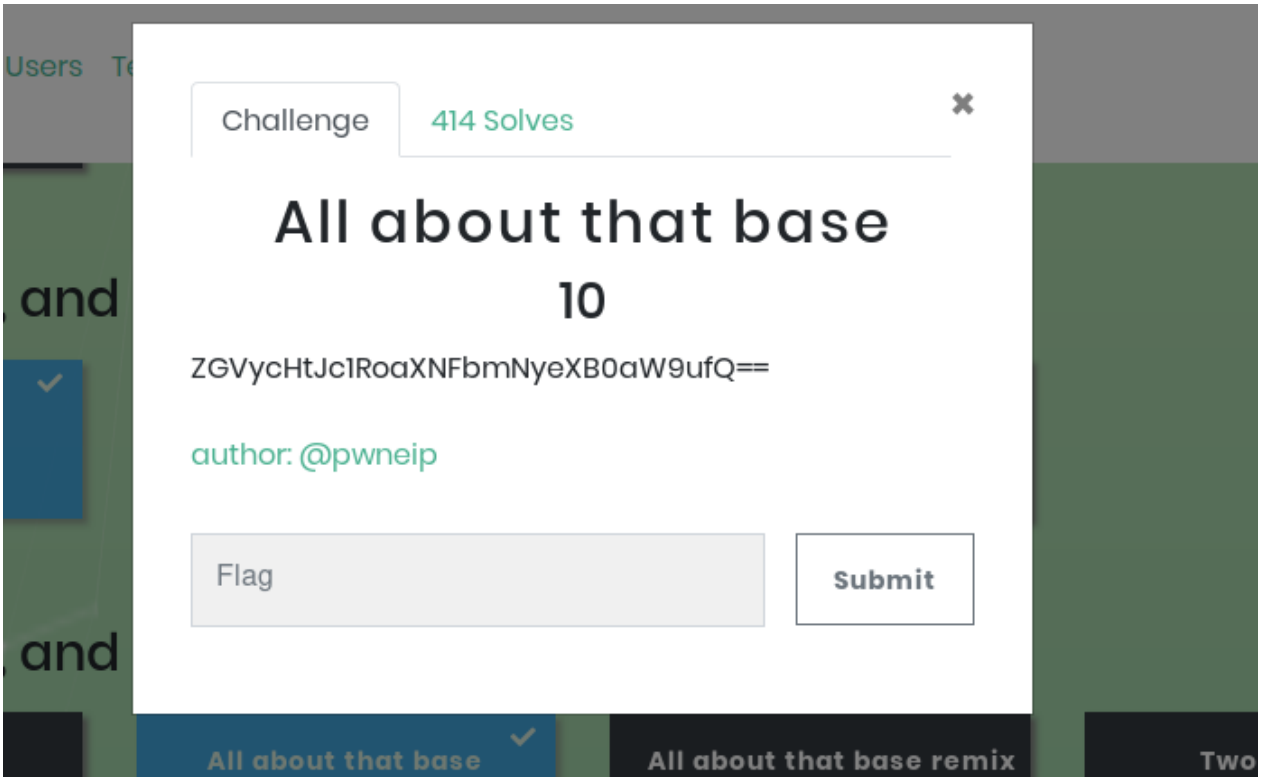


For this challenge, We notice that we have a sequence of A and B and after a little research it appears that this sequence of A and B is an encoding format called Beacon chiper and to decode it we used the platform https://cryptii.com/ using the viginer chiper option and we get

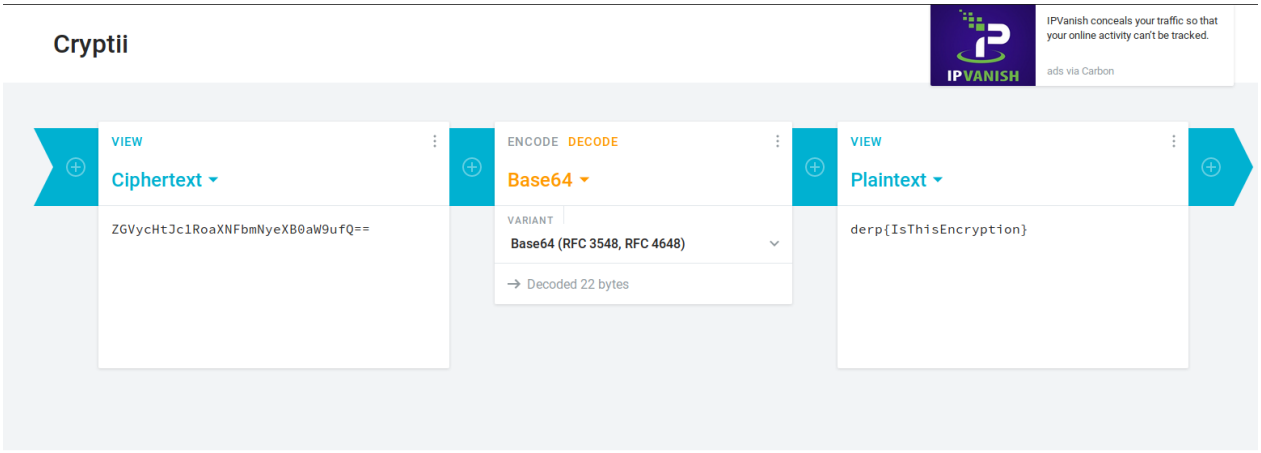


The flag is : **derp{ilovebeacon}**

➢ All about the base

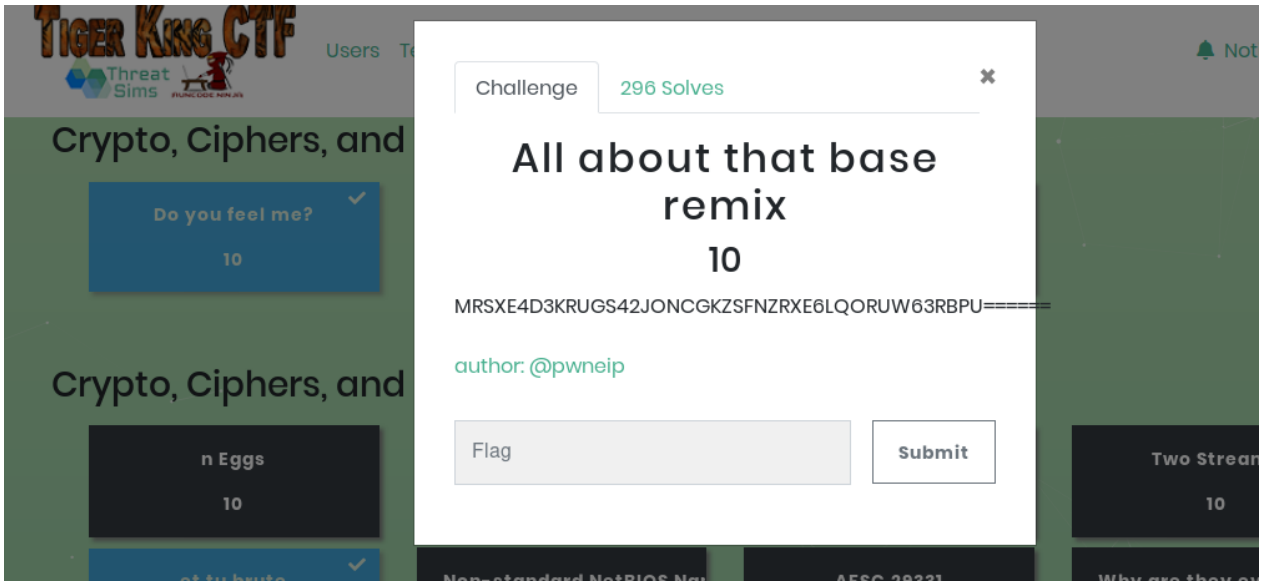


In this challenge we can see that this encoding is that of base64 and therefore to decipher the flag we used the site https://cryptii.com/ using the option base64 decode.

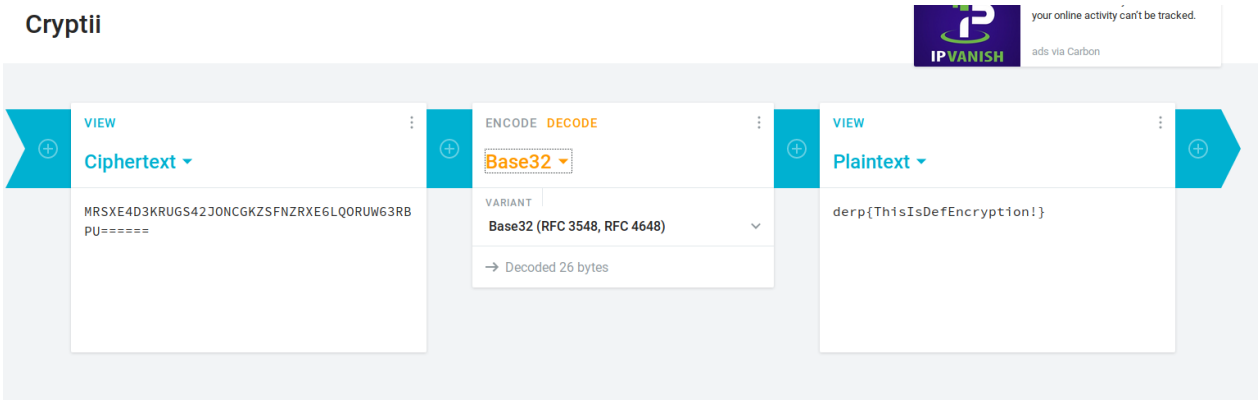


The flag is : **derp{IsThisEncryption}**

➢ All about the base remix

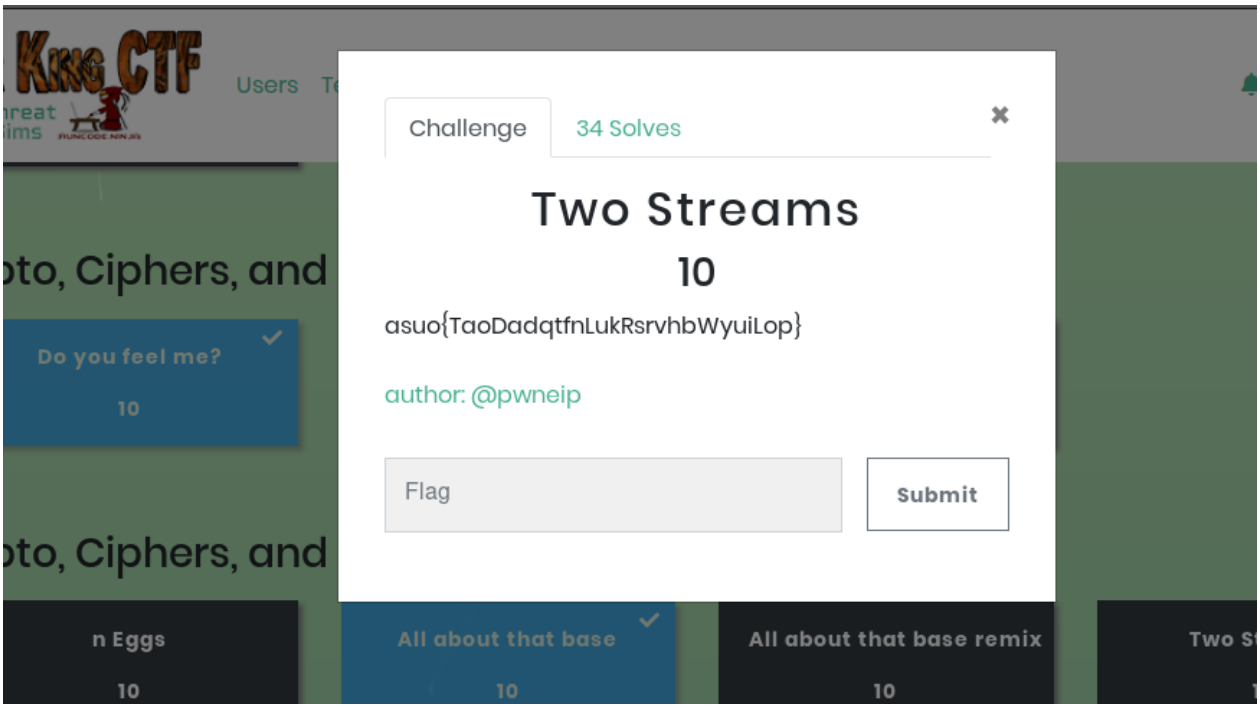


In this challenge we can see that this encoding is that of base32 and therefore to decipher the flag we used the site https://cryptii.com/ using the option base32 decode

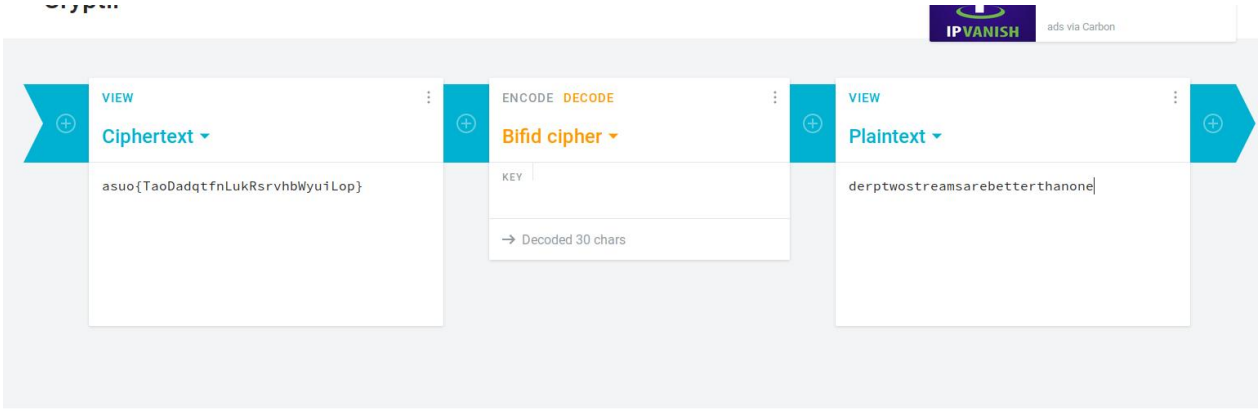


The flag is : **derp{ThisIsDefEncryption!}**

➢ Two streams

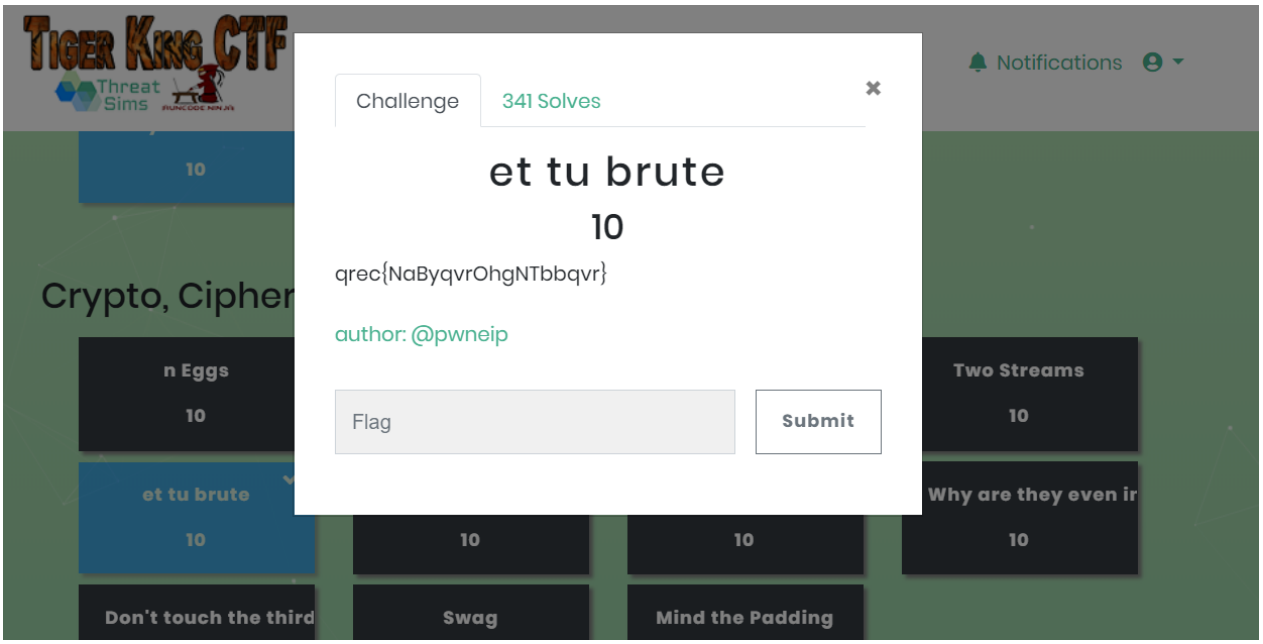


In this challenge we can see that this encoding is that of base32 and therefore to decipher the flag we used the site https://cryptii.com/ using the option base32 decode.



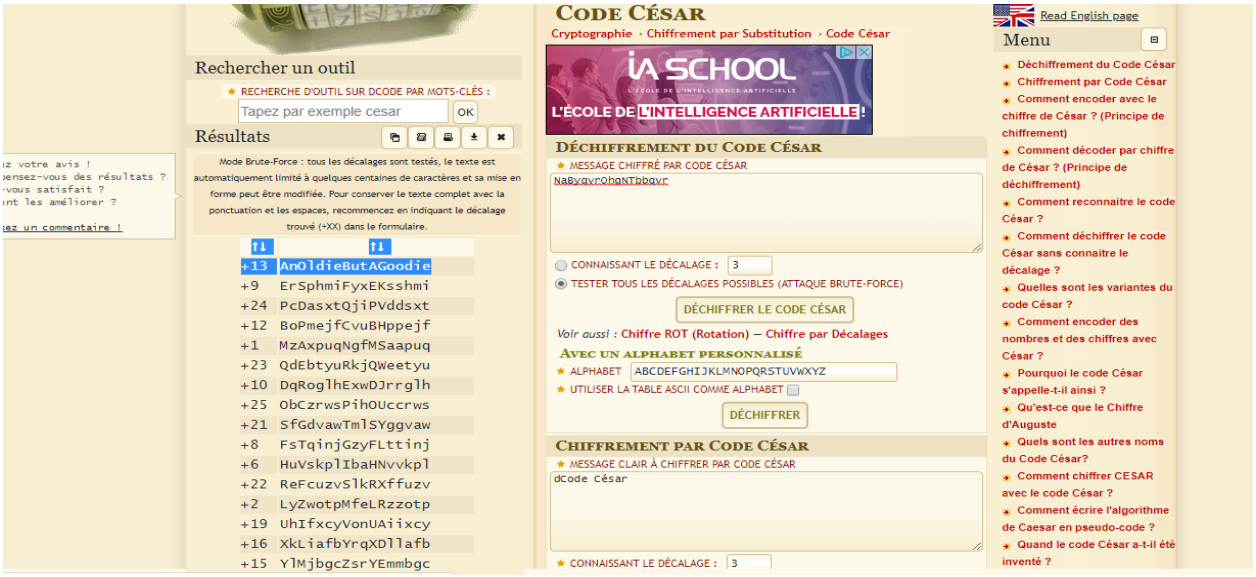
The flag is : **derp{TwoStreamsAreBetterThanOne}**

➢ Et tu brute



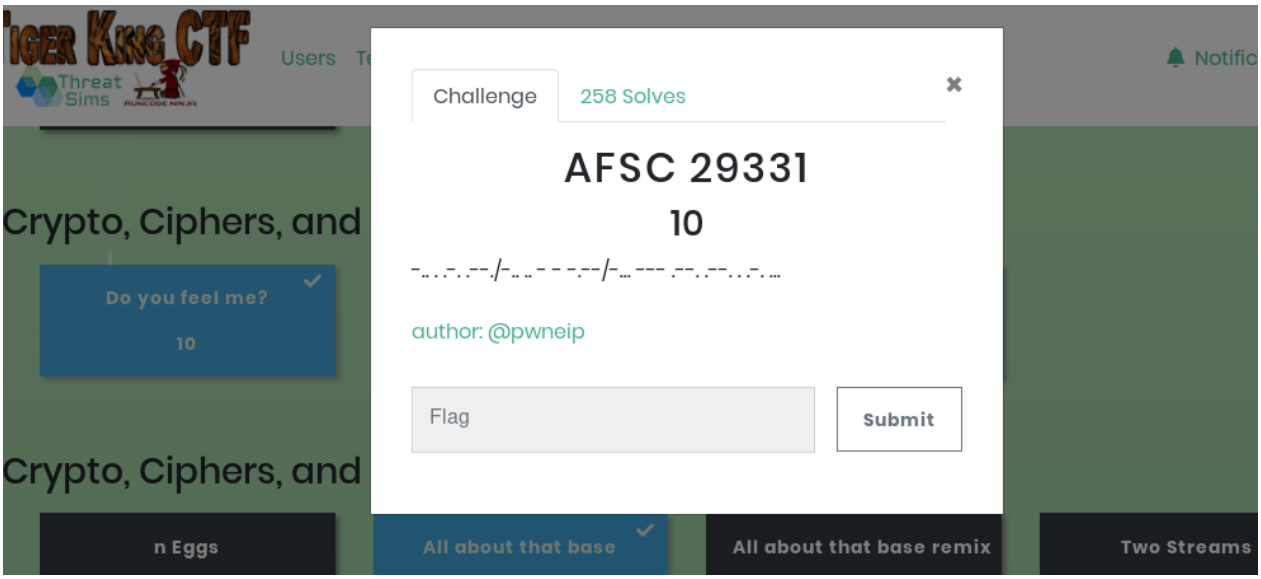
The title lets us think the challenge resolution will be based on a bruteforcing test. We have an encrypted word given by the challenge which is: **qrec {NaByqvrOhgNTbbqvr}**. As our flag starts with **derp** we are going to look for what encryption has transformed derp into **qrec**. The first encryption that came to mind is offset encryption (variant of Cesar encryption). We went to https://www.dcode.fr/chiffre-cesar where we entered the encrypted word **qrec**. We found that with an offset of **+13** we find the word **derp**.

We will do the same for our word in the **NaByqvrOhgNTbbqvr** brace and we get with the same offset (+13): **AnOldieButAgoodie**.

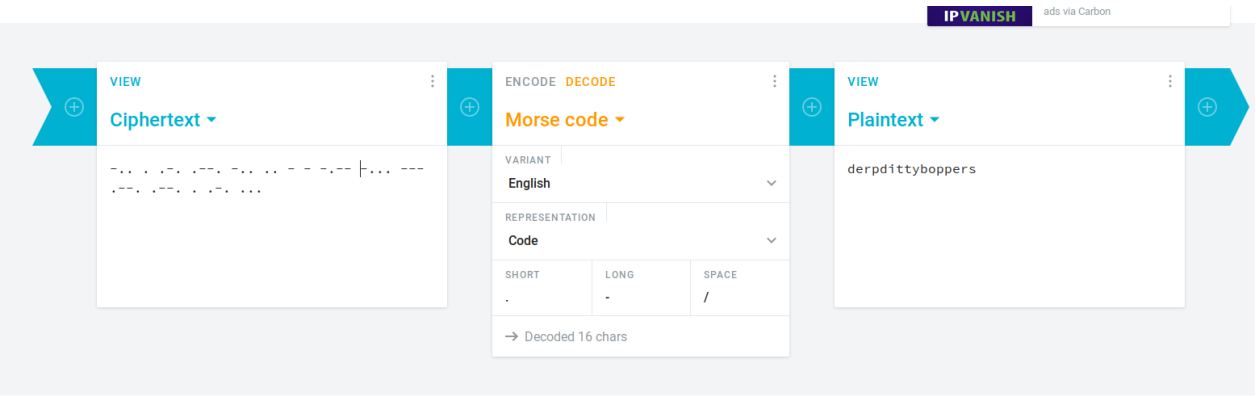


Finally, our encrypted flag **qrec{NaByqvrOhgNTbbqvr}** becomes with **Caesar encryption (offset of +13)**: **derp{AnOldieButAgoodie}**

➢ AFSC 29331



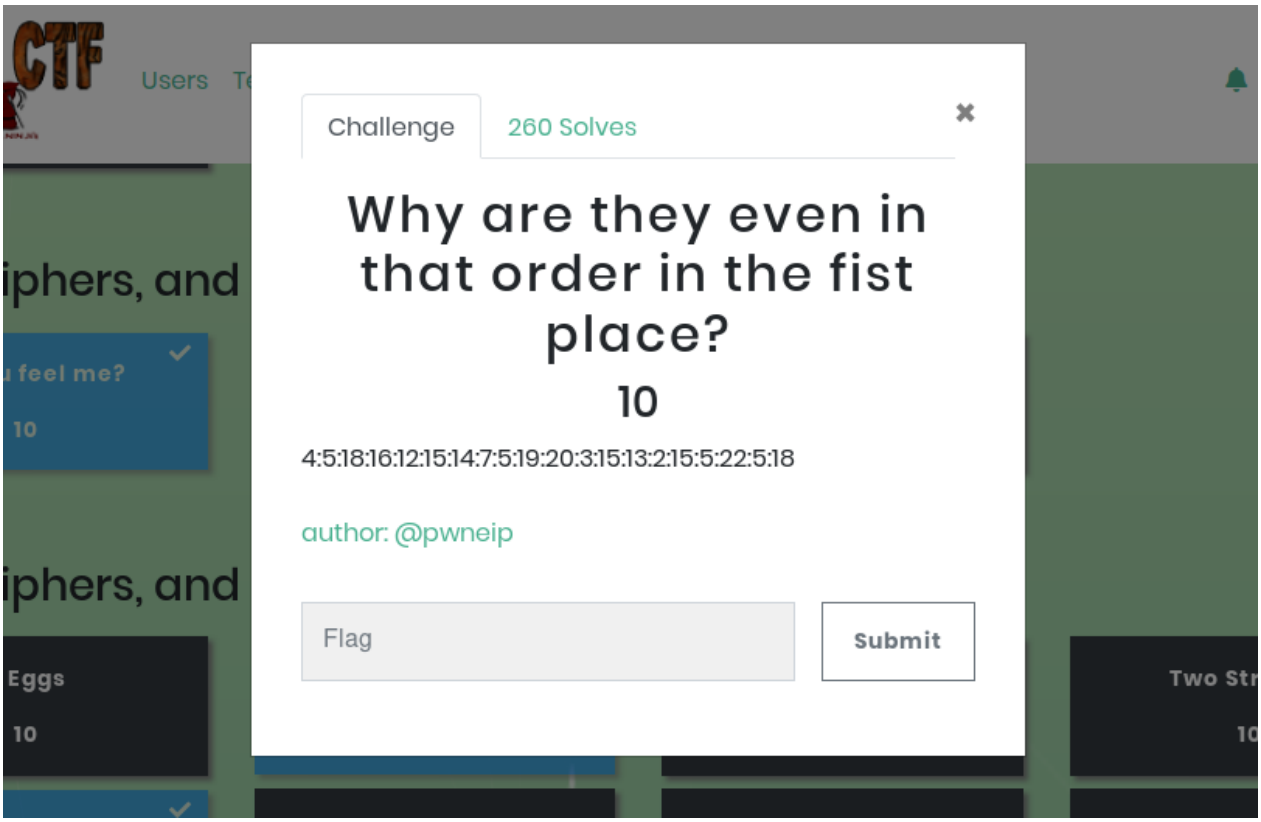
In this challenge after a little research on the text format we can see that this encoding is that of Morse code and therefore to decipher the flag we used the site https://cryptii.com/ using the Morse decode option



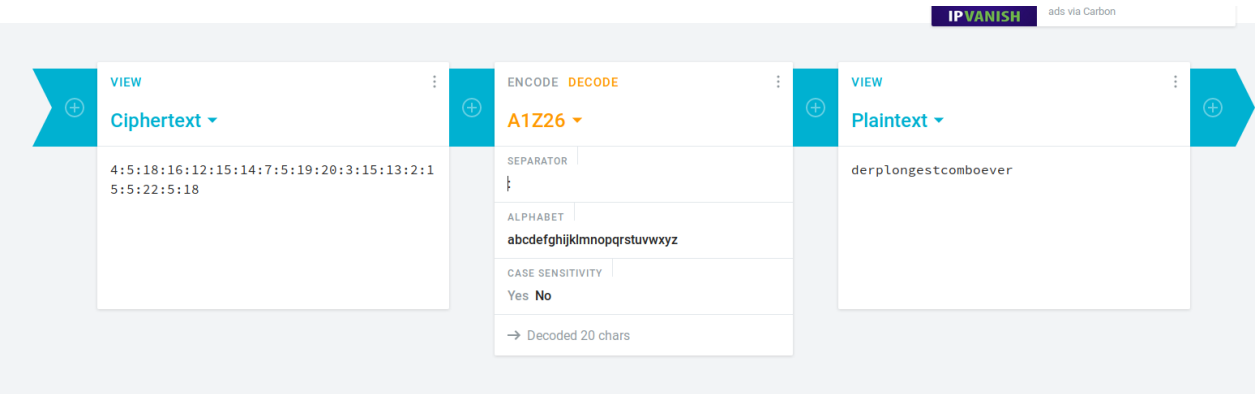
Vigenère cipher: Encrypt and decrypt online

The flag is : **derp{dittyboppers}**

➢ Why a the even in that

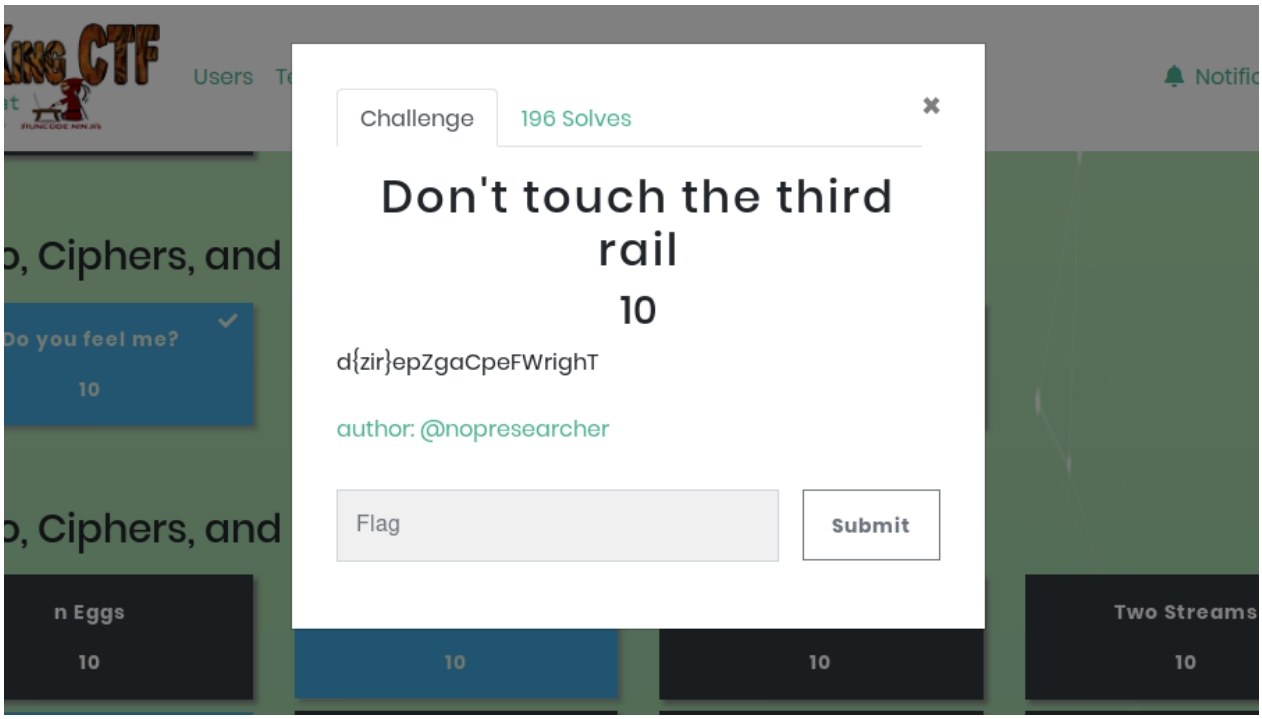


In this challenge after a little research on the text format we can see that this encoding is that of A1Z26 code and therefore to decipher the flag we used the site https://cryptii.com/ using the option A1Z26.

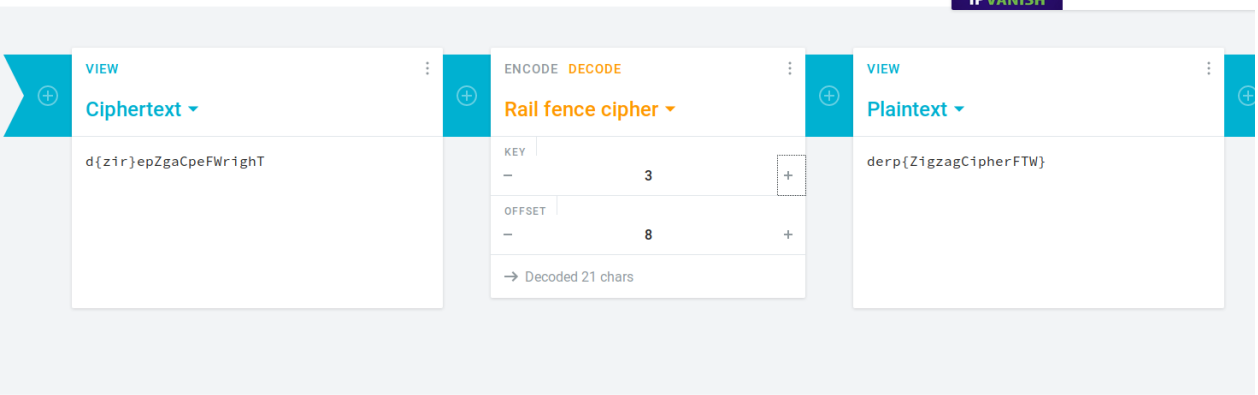


The flag is : **derp{longestcomboever}**

➢ Don t touche the third rail



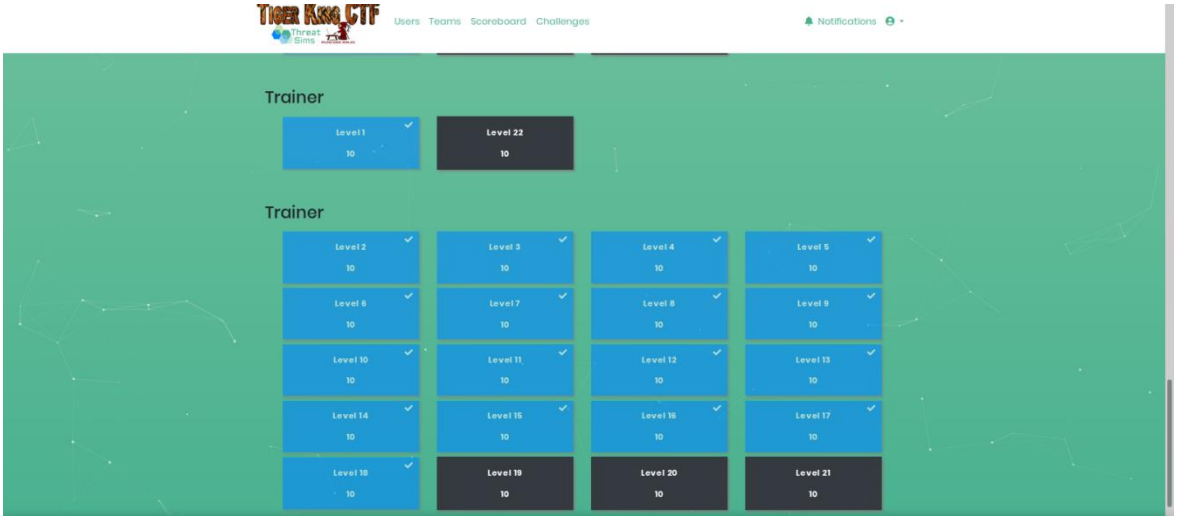
In this challenge by looking at the format of the text we can see that it contains all the characters except that it will shift after a little research it appears that this encoding is that of the rail fense code and therefore to decipher the flag we used the https://cryptii.com/ using the Rail Fense option



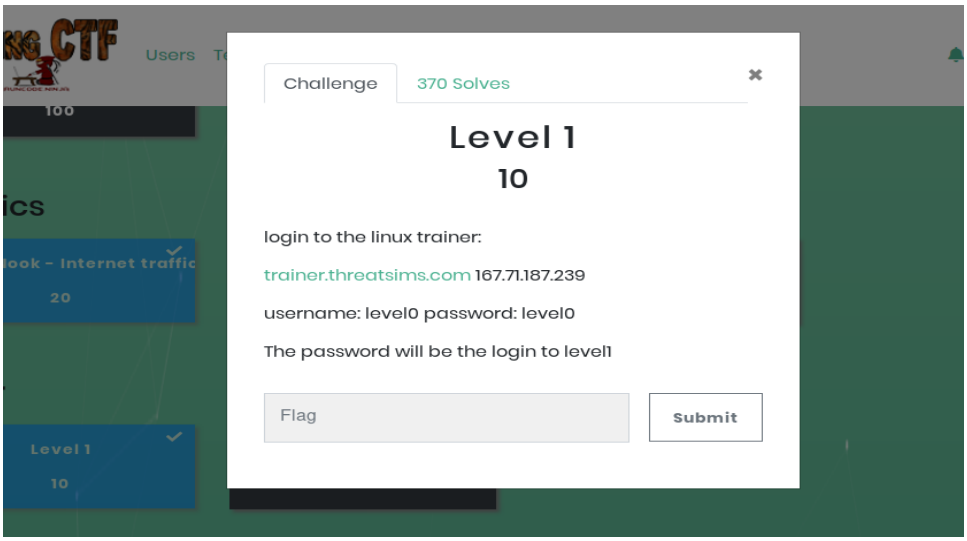
Vigenère cipher: Encrypt and decrypt online

The flag is : **derp{ZigzagChiperFTW}**

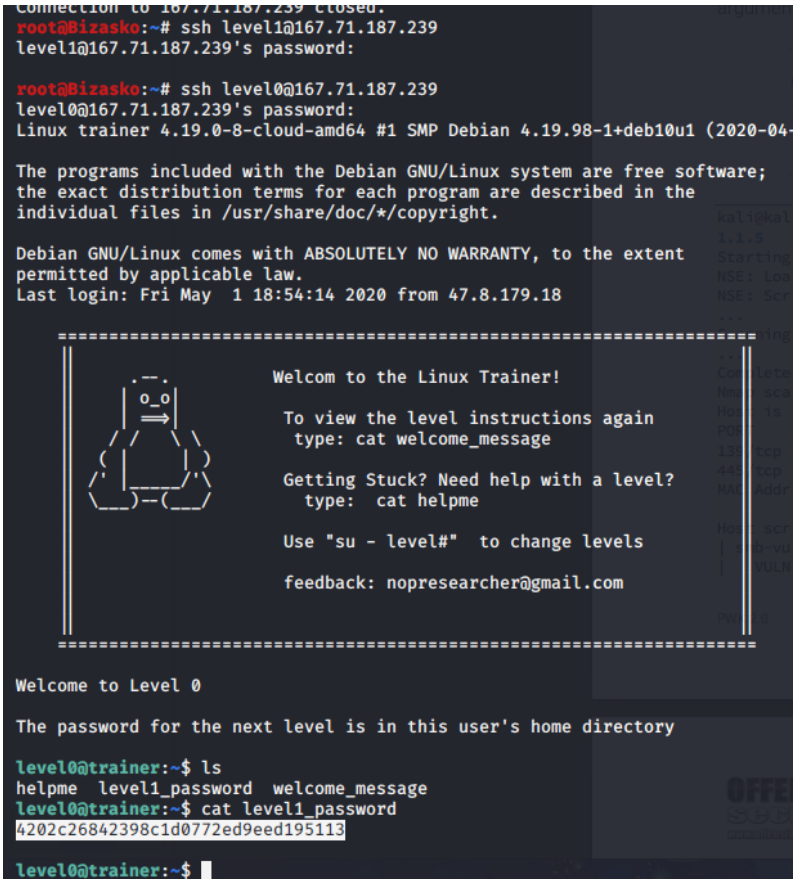
Trainer



Trainer 1



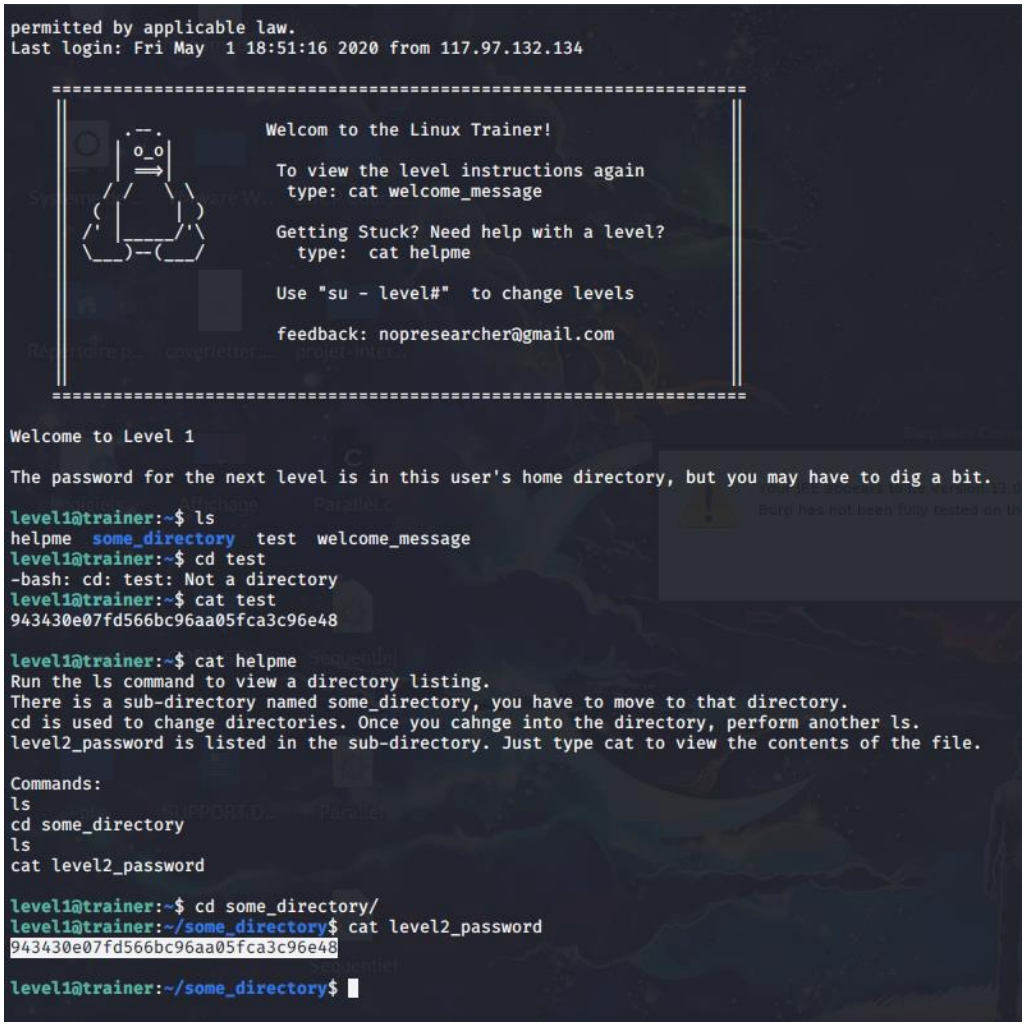
In this challenge, we used the accesses of the user level0 to have an access by ssh and afterwards we listed the contents of current directory with the command **ls** and we saw a file named **level1_password** which contains the flag and level1 user password.



The flag is : **42o2c2684239c1d0772ed9eed195113**

➢ Trainer 2

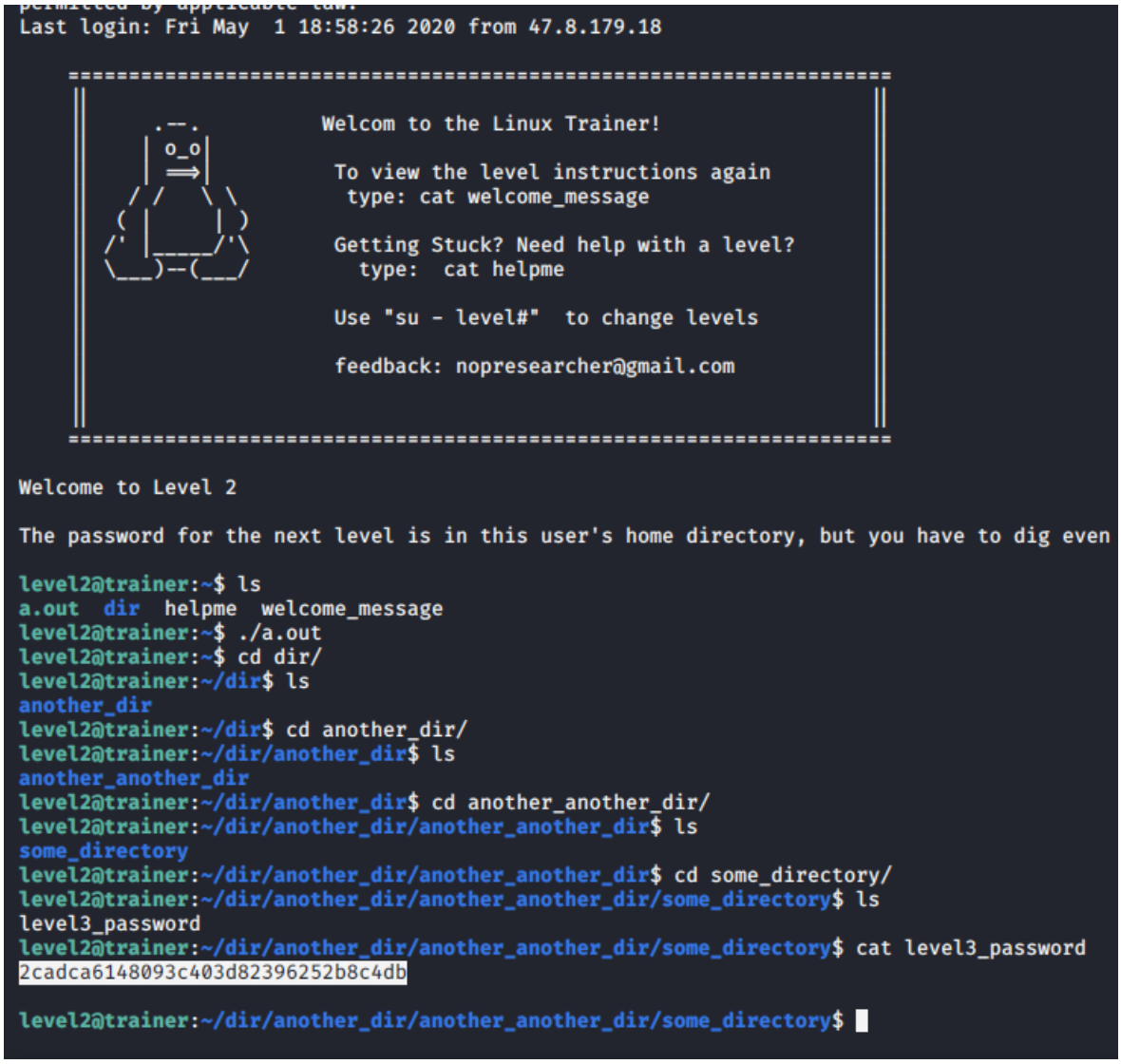
In this challenge, we used the password of the user level1 found in the previous flag to have access by ssh to its interface and then we listed the contents of the current directory with the command ls and we learned a file called helpme which we displayed with the **cat** command which allowed us to follow the instructions for having the flag displayed in a subfolder.



The flag is : **943430e07fd566bc96aa05fca3c96e48**

➢ Trainer 3

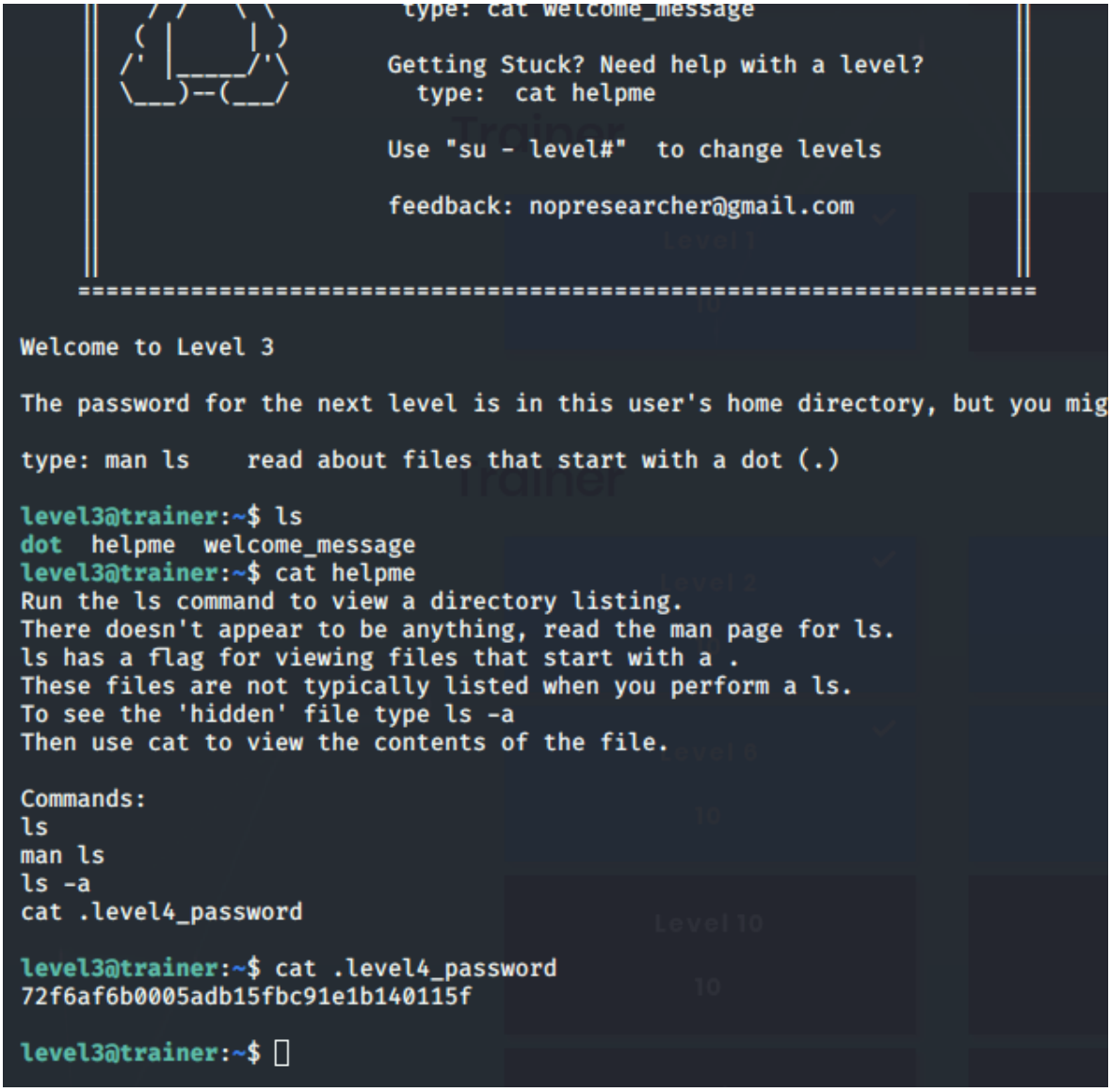
In this challenge, we used the password of the user level2 found in the previous flag to have access by ssh to its interface and then we listed the contents of current directory with the command **ls** and we learned a file called helpme which we displayed with the command **cat** which allowed us to follow the instructions for having the flag displayed in 3 subfolders.



The flag is : **2cadca6148093c403d82396252b8c4db**

➢ Trainer 4

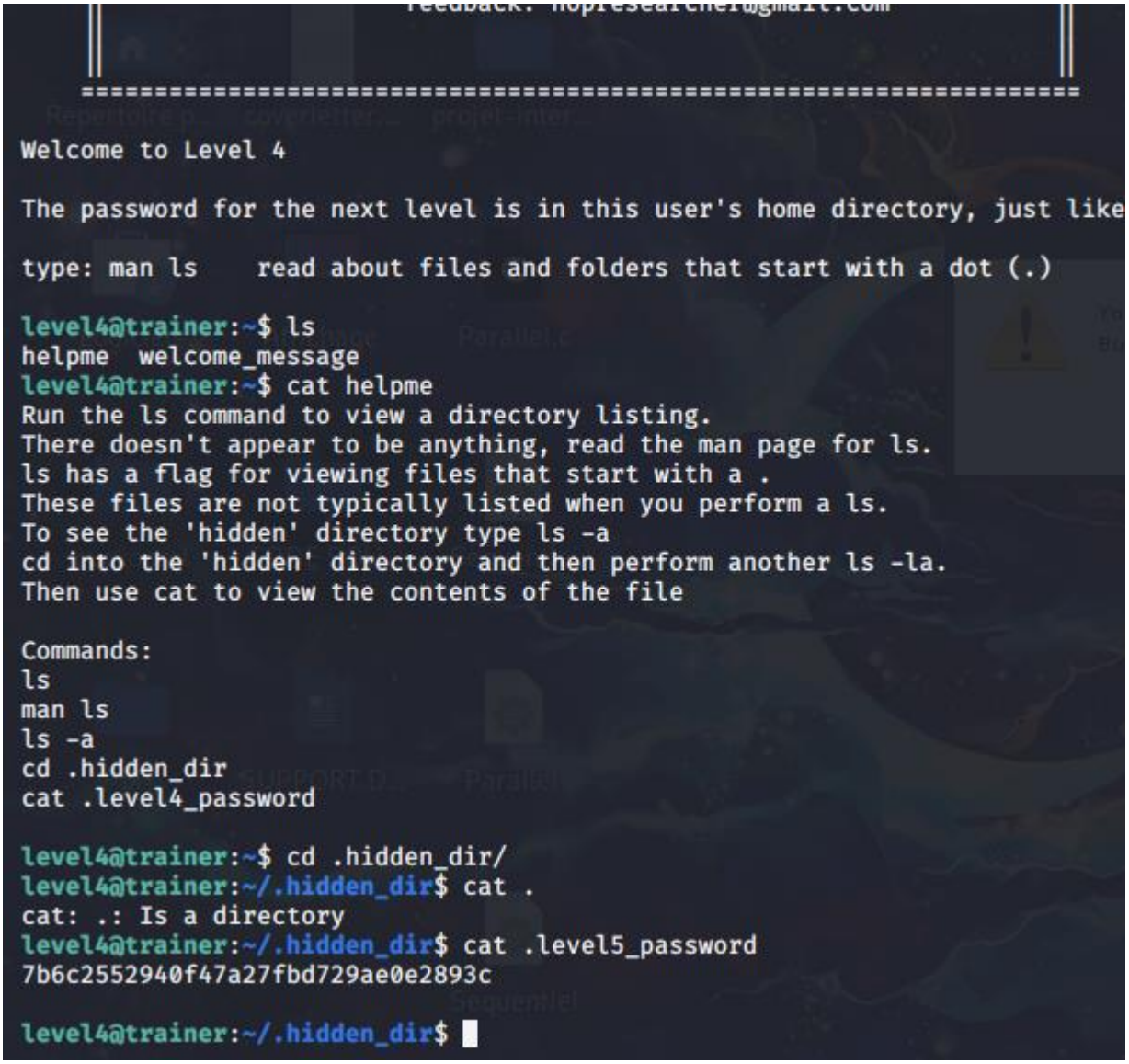
In this challenge, we used the password of the user level3 found in the previous flag to have access by ssh to its interface and then we listed the contents of the current directory with the command **ls -a** and we we have seen a cache file named **.level4_password** which contains the flag and password of the user level4.



The flag is : **72f6af6b0005adb15fbc91e1b140115f**

➢ Trainer 5

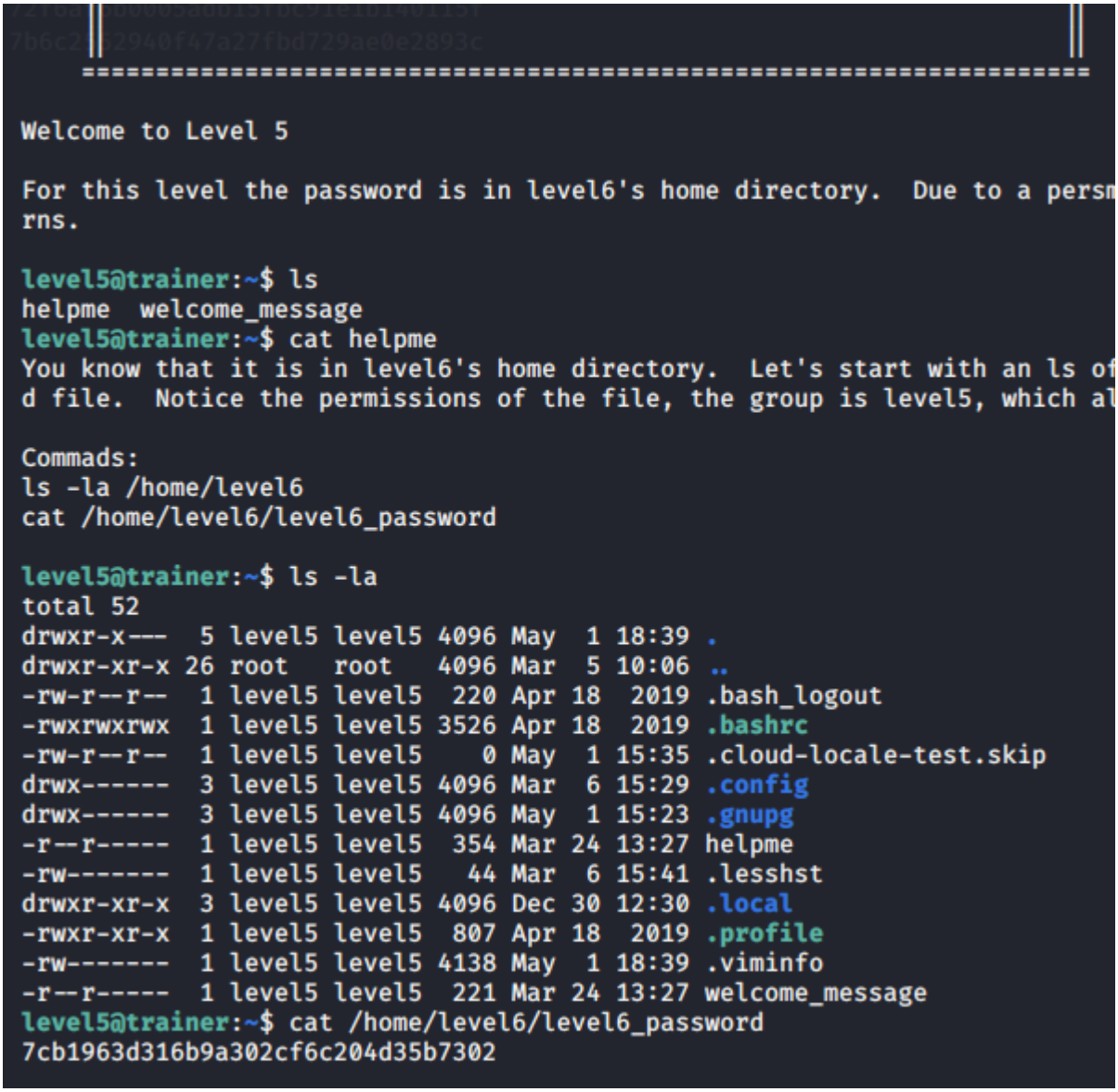
In this challenge, we used the password of the user level4 found in the previous flag to have access by ssh to its interface and then we listed the contents of the current directory with the command **ls -a** and we we have seen a folder **.hidden_dir** which contains a cache file named **.level5_password** which contains the flag and password of the user level5.



The flag is : **7b6c2552940f47a27fbd729ae0e2893c**

➢ Trainer 6

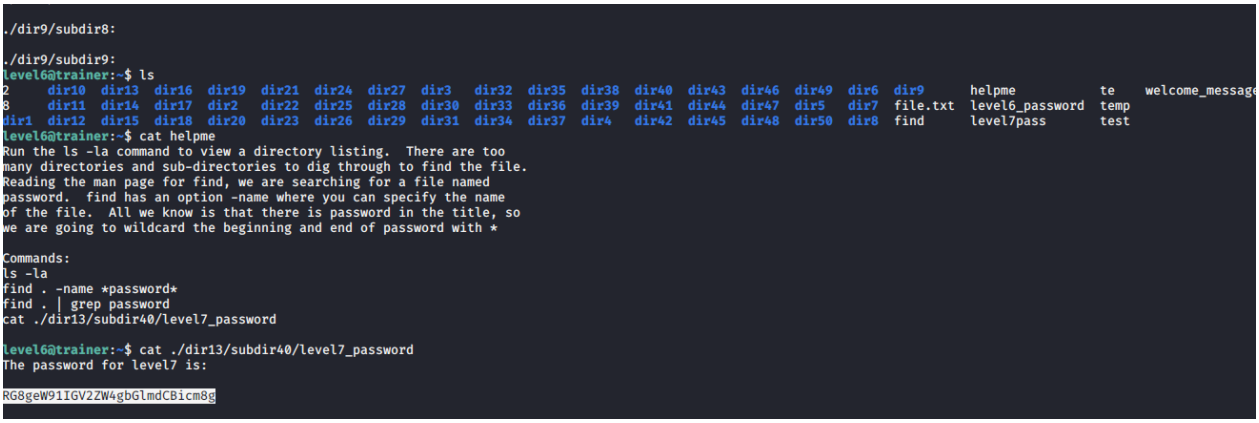
In this challenge, we used the password of the user level5 found in the previous flag to have access by ssh to its interface and after verification we found that the current folder does not contain any folder likely to contain the flag afterwards we tried to see if we have access rights to the folder of user6 with the command **ls -la Home/level6** and we have seen a cache file called **.level6_password** which contains the flag and the password of user level6.



The flag is : **7cb1963d31b9a302cf6c204d35b7302**

➢ Trainer 7

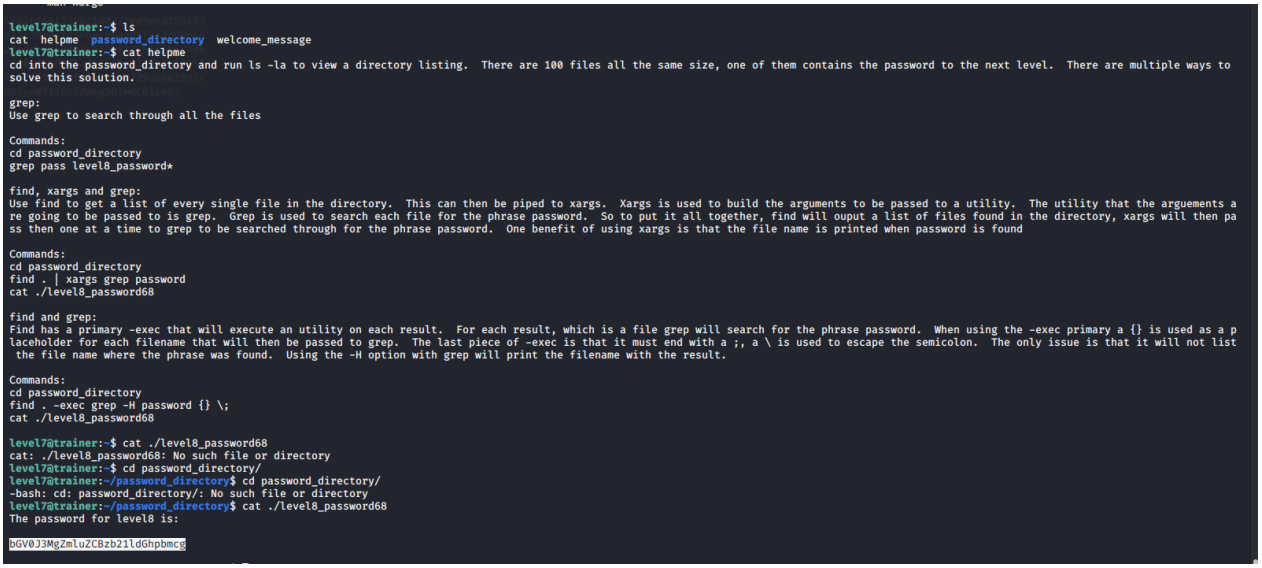
In this challenge, we used the password of the user level6 found in the previous flag to have access by ssh to its interface and after verification we found that the current folder does not contain several folders likely to contain the flag we tried to do a targeted search using the command **find** and **grep** to perform a search and a filtering of the results and we saw a cache file named **level7_password** which contains the flag and password of the user level6.



The flag is : **RG8geW91IGV2ZW4gbGlmdCBIcm8g**

➢ Trainer 8

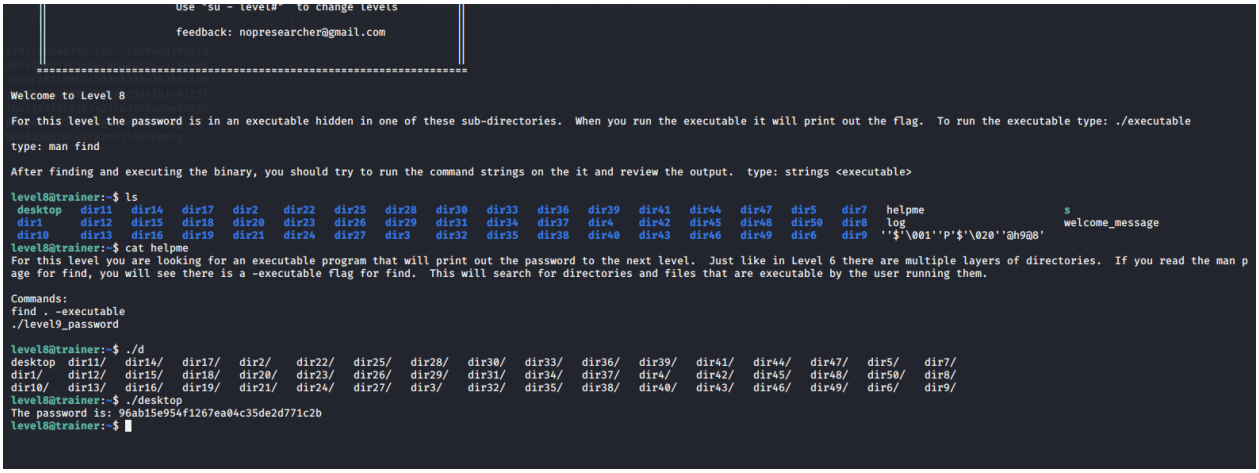
In this challenge, we used the password of the user level7 found in the previous flag to have access by ssh to its interface and then we listed the contents of the current directory with the command **ls** and we learned a file named helpme that we displayed with the command **cat** which allowed us to follow the instructions to have the flag displayed which ended up being **level8_password68**.



The flag is : **bGV0J3MgZCBzb21ldGhpbmcg**

➢ Trainer 9

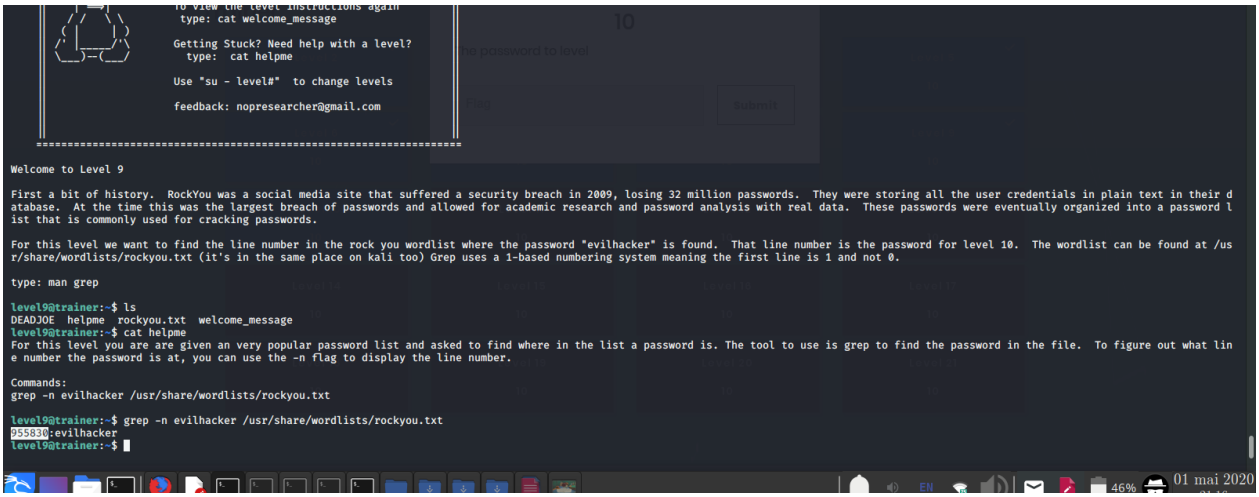
In this challenge, we used the password of the user level8 found in the previous flag to have access by ssh to its interface and then we listed the contents of current directory with the command ls and we learned a script called desktop that we executed. This allowed us to display the flag.



The flag is :**96ab15e954f126ea04c35de2d771c2b**

➢ Trainer 10

In this challenge, we used the password of the user level9 found in the previous flag to have access by ssh to its interface. Given the instruction it was a question of finding the line of the word evilhacker in the dictionary rockyou.txt which happens to be the flag. and we have the command **grep -n evilhacker rockyou.txt.** This allowed us to display the flag.



The flag is : **955830**

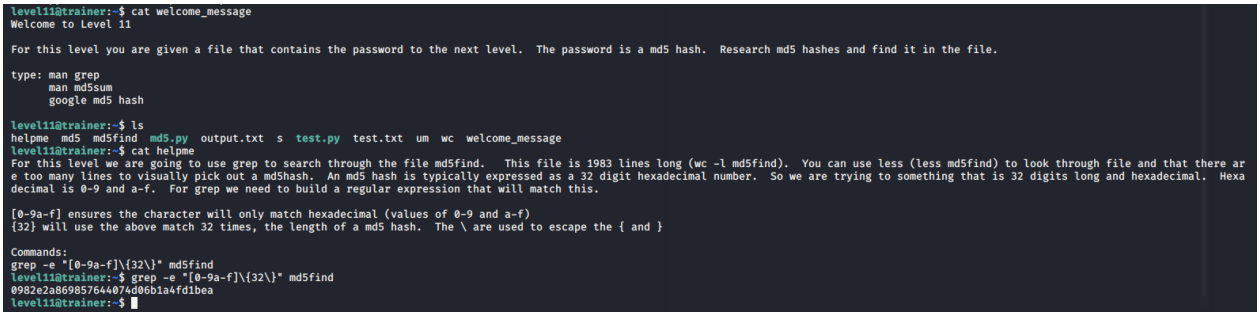
➢ Trainer 11

In this challenge, we used the password of the user level10 found in the previous flag to have access by ssh to its interface. Given the instruction it was a question of finding the number of times the ip address 112.85.42.94 appeared in a log file called fail2ban.log which happens to be the flag. and we made the command **grep " Ban 112.85.42.94 " fail2ban.log | wc -l.** This allowed us to display the flag.

The flag is : **192**

➢ Trainer 12

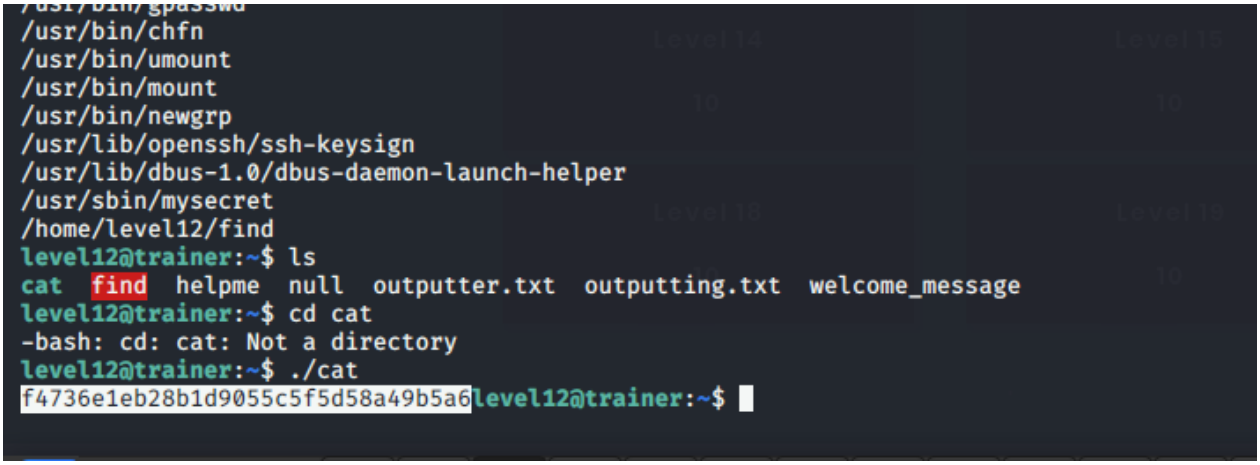
In this challenge, we used the password of the user level11 found in the previous flag to have access by ssh to its interface. Given the instruction it was a question of finding a flag in hash format md5 in a file called md5find. and we made the command **grep -e " [0-9a-f]\{32\}" md5fing.** This allowed us to display the flag.



The flag is : **0982e2869857644074d06b1a4fd1bea**

➢ Trainer 13

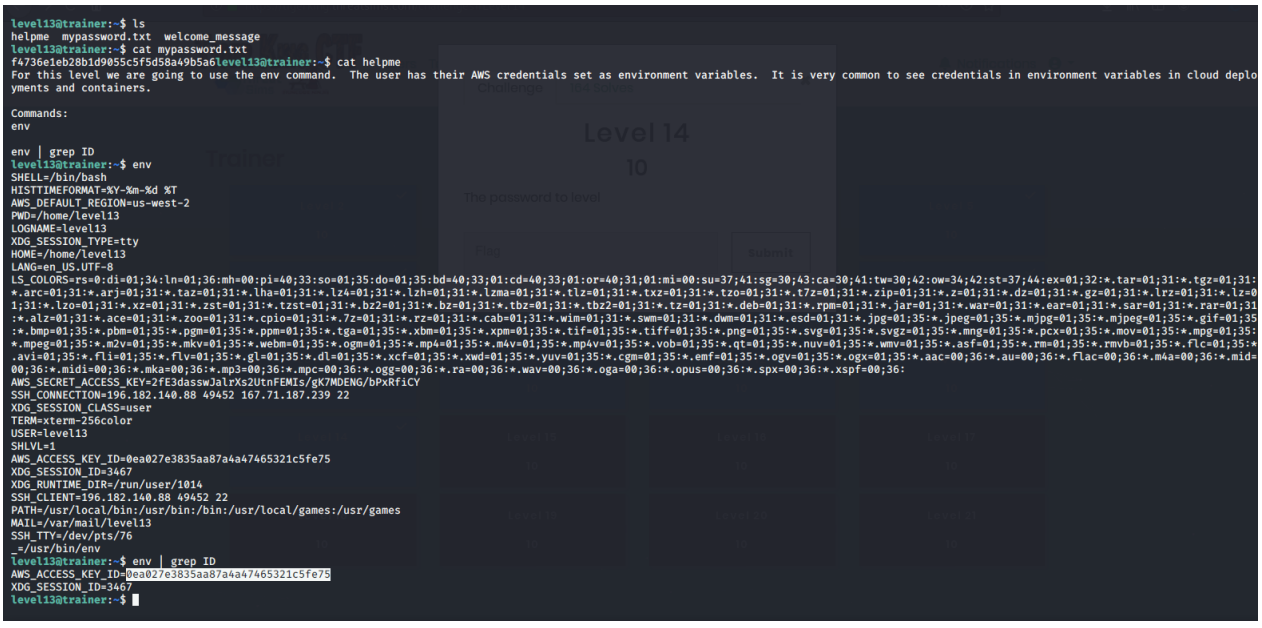
In this challenge, we used the password of the user level13 found in the previous flag to have access by ssh to its interface and then we listed the contents of the current directory with the command ls and we learned a script called cat that we executed. This allowed us to display the flag.



The flag is : **f4736e1e1eb28b1d9055c5f5d58a49b5a6**

➢ Trainer 14

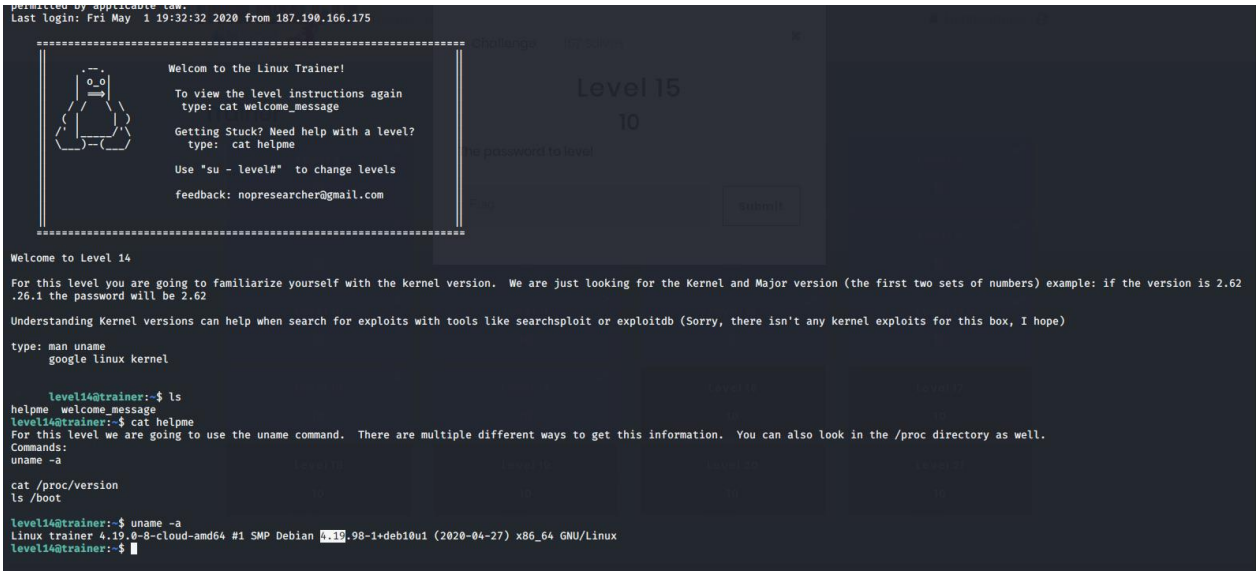
In this challenge, we used the password of the user level14 found in the previous flag to have access by ssh to its interface and then we listed the contents of the current directory with the command **ls** and we learned a file called helpme which we displayed with the command **cat** which allowed us to follow the instructions to have the flag displayed.



The flag is : **0ea027e3835aa87a4a47465321c5fe75**

➢ Trainer 15

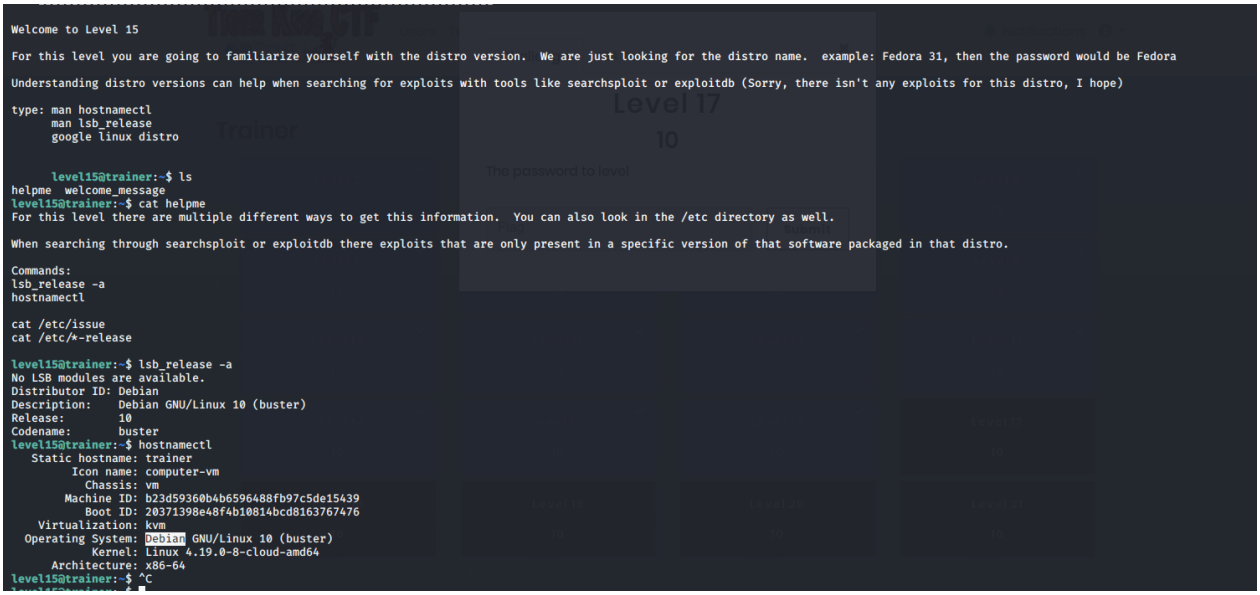
In this challenge, we used the password of the user level15 found in the previous flag to have access by ssh to its interface. Given the instruction it was a question of finding the version of the kernel which happens to be the flag, we made the command **uname -a** This one allowed us to display the flag.



The flag is : **4.19**

➢ Trainer 16

In this challenge, we used the password of the user level16 found in the previous flag to have access by ssh to its interface. Given the instruction it was a question of finding the distribution which is found the flag, we made the command **lsb_release -a** This one allowed us to display the flag.



The flag is : **Debian**

➢ Trainer 17

In this challenge, we used the password of the user level17 found in the previous flag to have access by ssh to its interface and then we listed the contents of current directory with the command **ls** and we learned a file called helpme which we displayed with the command **cat** which allowed us to follow the instructions for having the flag.

The flag is : **6b39034a8045ed996a436f8d09031522**

➢ Trainer 18

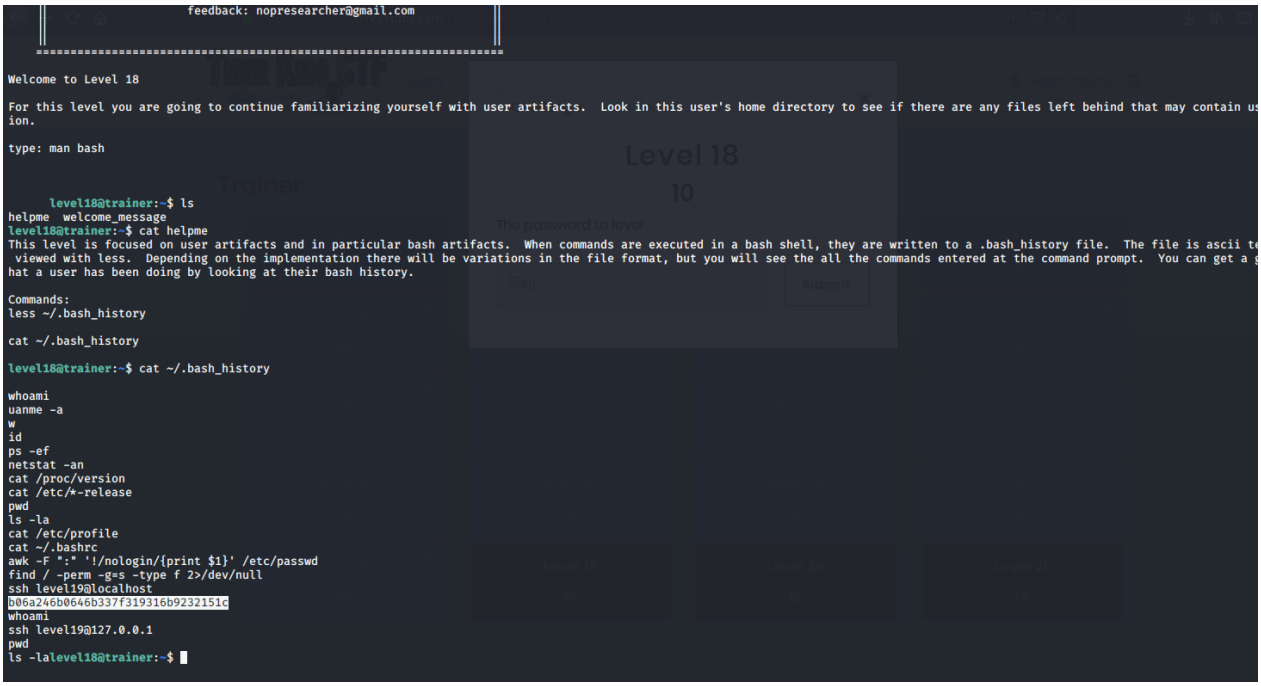
In this challenge, we used the password of the user level18 found in the previous flag to have access by ssh to its interface and then we listed the contents of current directory with the command **ls** and we learned a file called helpme which we displayed with the command **cat** which allowed us to follow the instructions for having the flag.

```
less .viminfo

cat .viminfo

level17@trainer:~$ cat .vim
cat: .vim: Is a directory
level17@trainer:~$ cat .viminfo
# This viminfo file was generated by Vim 8.1.
# You may edit it if you're careful!

# Viminfo version
|1,4

# Value of 'encoding' when this file was written
*encoding=utf-8

# hlsearch on (H) or off (h):
~h
# Command Line History (newest to oldest):
:wq
|2,0,1583196421,,"wq"

# Search String History (newest to oldest):

# Expression History (newest to oldest):

# Input Line History (newest to oldest):

# Debug Line History (newest to oldest):

# Registers:
"a      LINE    0
        ssh level18@localhost
|3,0,10,1,1,0,1583196413,"ssh level18@localhost"
""b     LINE    0
        9a42b1822710d790a393800f2896a8f7
|3,1,11,1,1,0,1583196418,"9a42b1822710d790a393800f2896a8f7"

# File marks:
'0  11  0  ~/password
|4,48,11,0,1583196421,"~/password"
'1  12  0  ~/password
|4,49,12,0,1583196243,"~/password"
'2  11  0  ~/password
|4,50,11,0,1583196243,"~/password"
```

The flag is : **9a42b1822710d790a393800f2896a8f7**

➢ Trainer 19

In this challenge, we used the password of the user level19 found in the previous flag to have access by ssh to its interface and then we listed the contents of current directory with the command **ls** and we learned a file called helpme which we displayed with the command **cat** which allowed us to follow the instructions for having the flag.

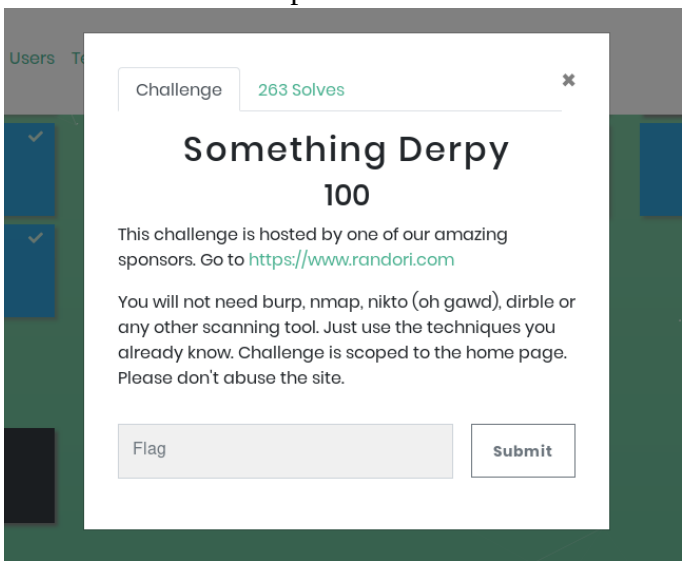


The flag is : **b06a246b0646b337f319316b9232151c**

WEB

➢ Something Derpy

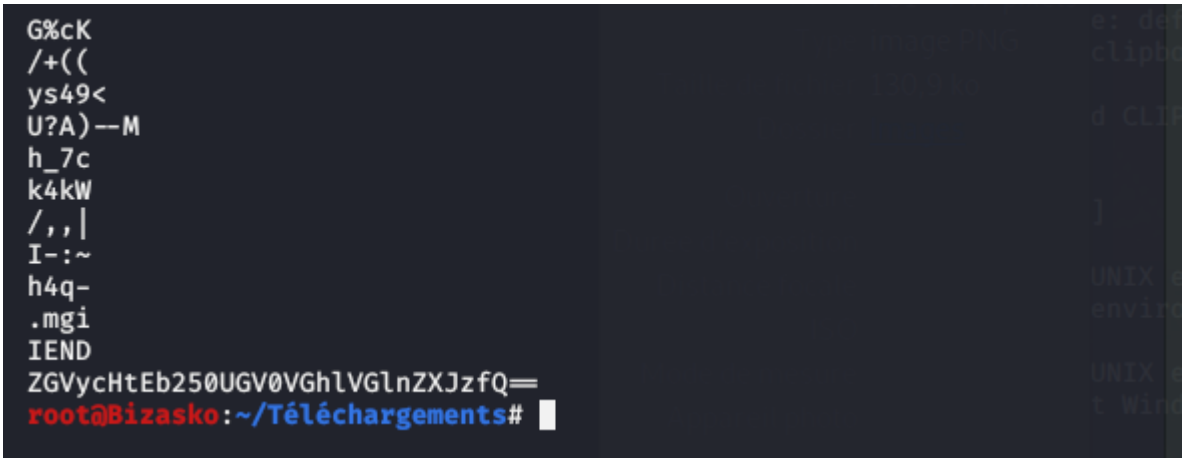
In this challenge, when we click on the description link we come across the website of a



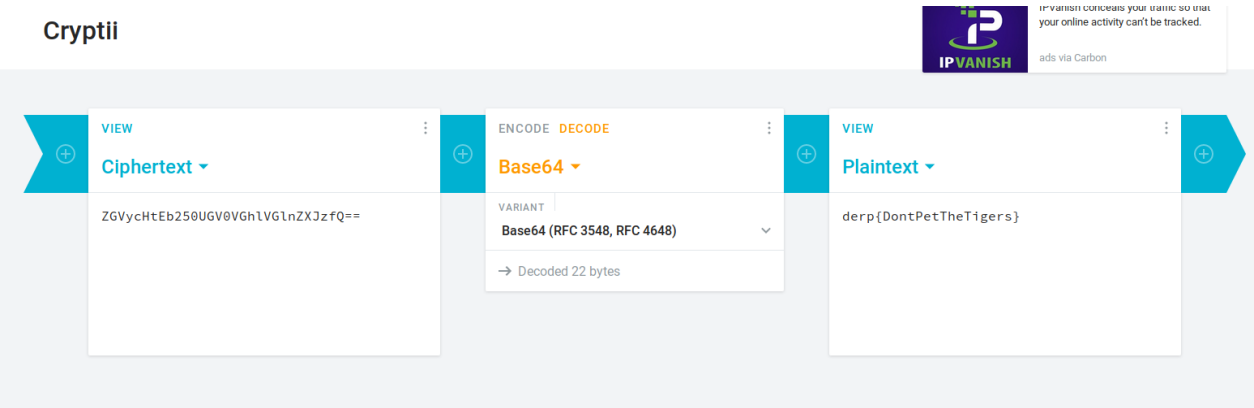
challenge partner. The inspiration of the source page allowed us to come across a message accompanied by an image.

```
<li class="c-footer__item o-grid__cell u-1/2 u-1/4--tablet"><a class="c-button" href="/free-trial/">Free Trial</a></li></ul>    </div>
  </nav>

  <section class="c-footer__legal">
    <center>
      <a href="/bart"><img src="/wp-content/themes/randori/assets/images/bart.png" alt="Bart"></a> <!-- something is derpy with this image, will fix later -->
    </center>
    <div class="o-grid u-clearfix">
      <ul id="menu-legal" class="o-grid__cell c-footer__legal-list"><li class="c-footer__legal-item"><a href="https://www.randori.com/privacy-policy/">Privacy Policy</a></li>
<li class="c-footer__legal-item"><a href="https://www.randori.com/terms-of-service/">Terms of Use</a></li>
<li class="c-footer__legal-item">&copy; 2020 Randori all rights reserved</li></ul>    </div>
```

We downloaded the image to analyze it and more closely with the **strings** command of kali linux and we got:

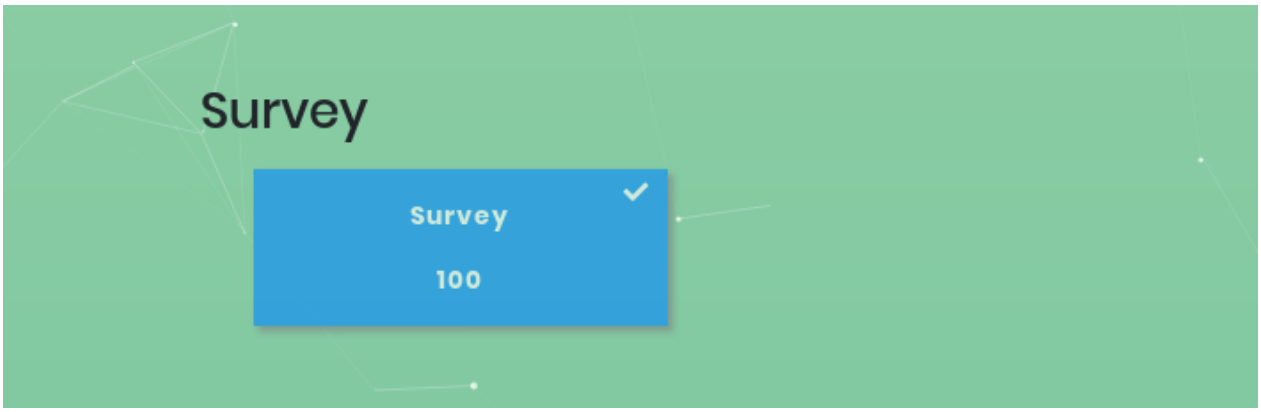


A base 64 code that we will try to decipher with the cryptii platform

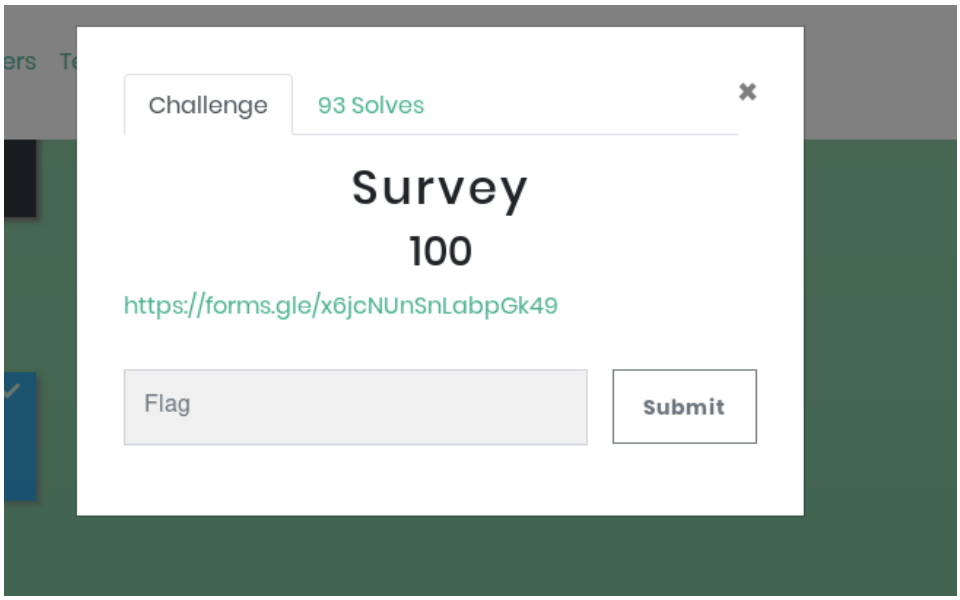


The flag is : **derp{DontPetTheTigers}**

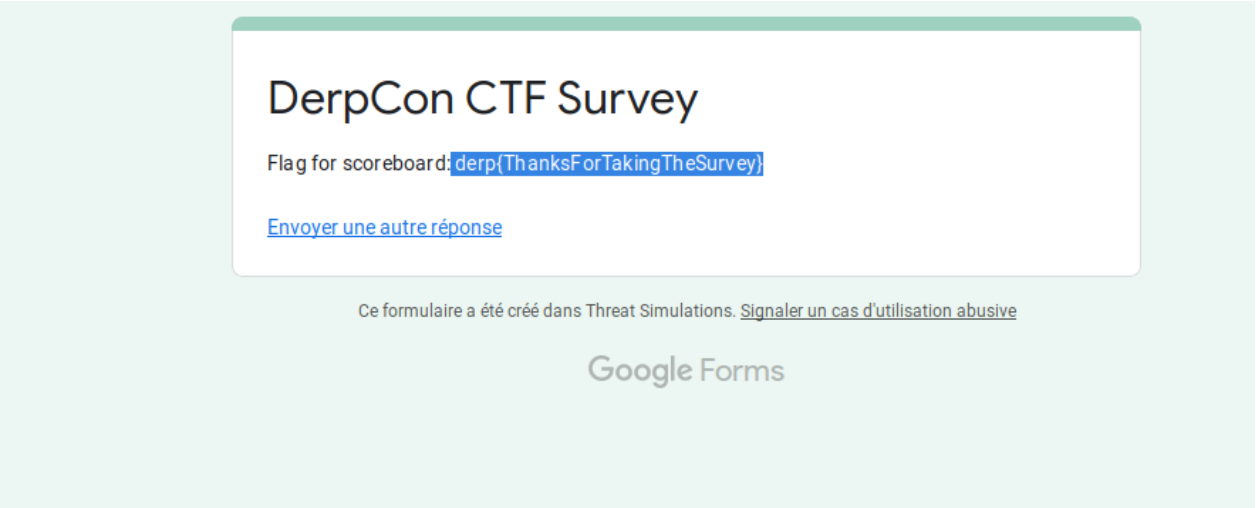
Servey



➢ Servey



This challenge consisted in filling in a google form to send it and as a sending success message we have the flag.



The flag is : **derp{ThankFortakingTheSurvey}**