

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 organized 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

Tiger King			
Web Recon 1 20	Web Recon 2 20	Showing off my Tigers 20	Eating Sweets 30
Browser Check 30	Admin Login 50	db query 1 50	db query 2 100
db query 3 100			

➤ Web recon 1

The screenshot shows a challenge card for "Web Recon 1" with a value of 20. The challenge description reads: "Joe developed the site and may have left some notes for you." Below the description is the URL "http://joe-tk.threatsim.com". The author is listed as "@nopresearcher". There are two buttons at the bottom: "Flag" and "Submit". The background of the card is white, while the surrounding interface has a dark green and grey theme.

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

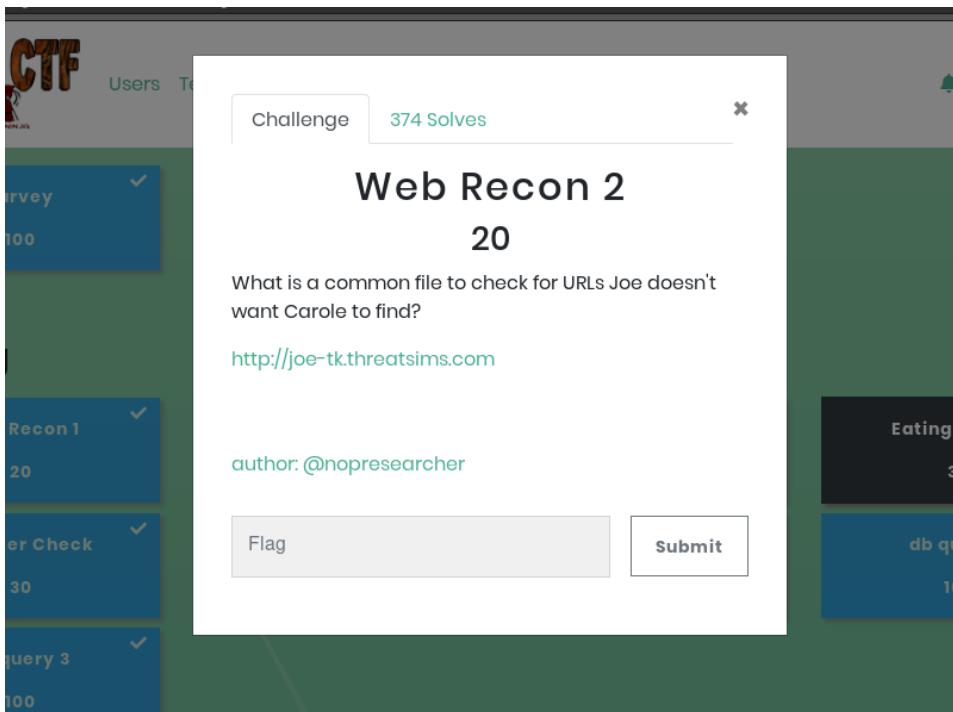
```

26     <li><a href="/tigers">Tigers</a></li>
27 </ul>
28 <ul class="nav pull-right">
29     <li><a href="/admin">Admin</a></li>
30     <li class="divider-vertical"></li>
31     </li>
32 </ul>
33 </div><!-- /.nav-collapse -->
34 </div><!-- /navbar-inner -->
35 </div><!-- /navbar -->
36
37
38 
39 <br>
40 
41 <!-- I don't know how many times I have to say it! derp{CaroleBaskinDidIt} -->
42
43
44
45
46
47
48 </body>
49 <br>
50 <br>
51 <br>
52 <br>
53 <br>
54 <br>

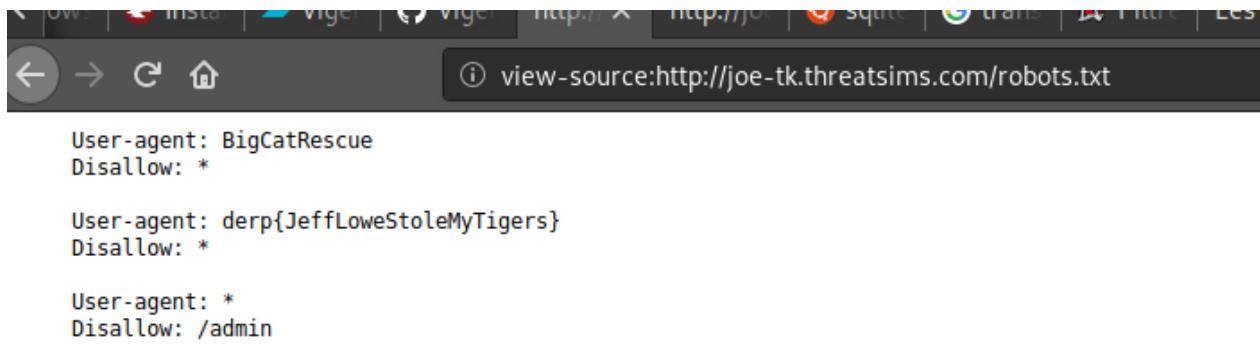
```

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.



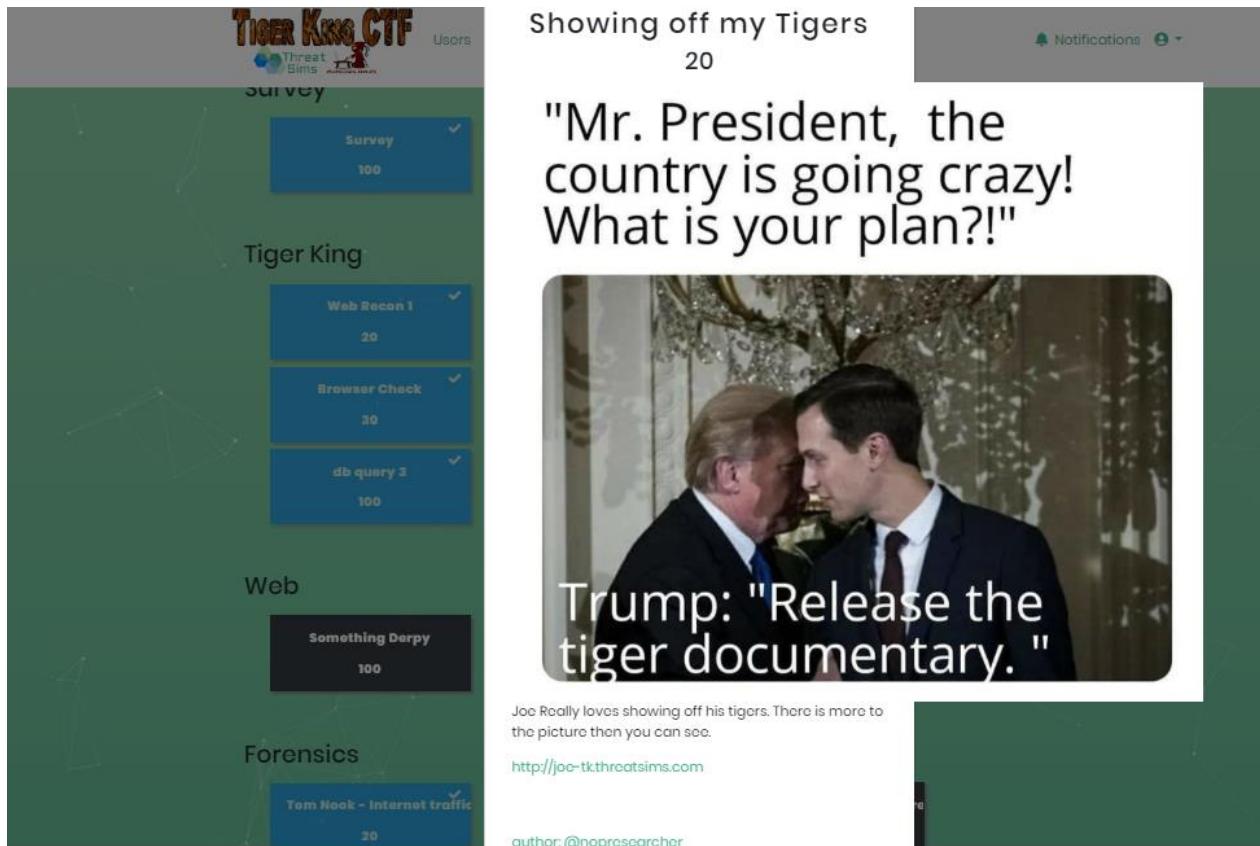
User-agent: BigCatRescue
Disallow: *

User-agent: derp{JeffLoweStoleMyTigers}
Disallow: *

User-agent: *
Disallow: /admin

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

➤ Showing off my Tigers



Showing off my Tigers
20

"Mr. President, the country is going crazy! What is your plan?!"



Trump: "Release the tiger documentary."

Joc Really loves showing off his tigers. There is more to the picture then you can see.
<http://joe-tk.threatsims.com>
author: @noipresearcher

Tiger King CTF

Survey

Survey 100

Tiger King

Web Recon 1 20

Browser Check 30

db query 3 100

Web

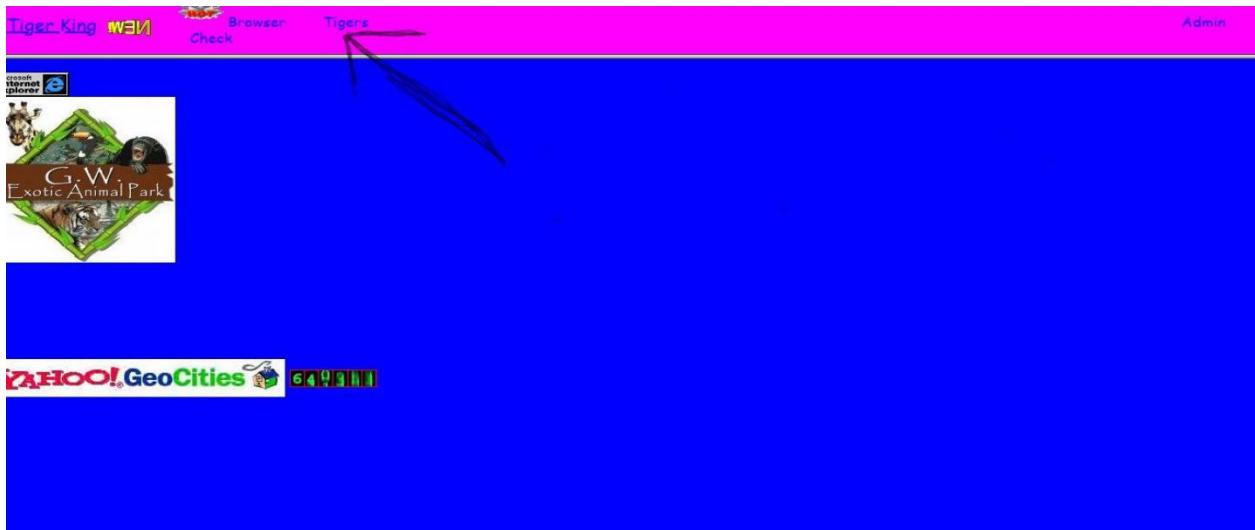
Something Derby 100

Forensics

Tom Hanks - Internet traffic 20

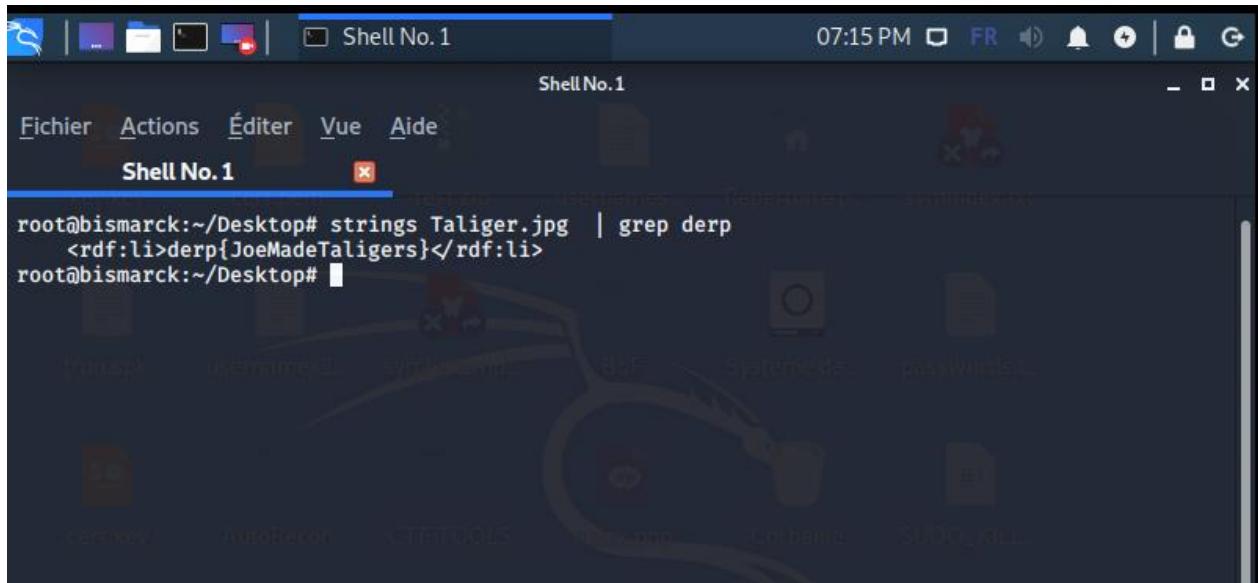
The title "**Showing off my tigers**" lets us think of "a concealment" and therefore a steganography challenge because to show means to show and the word off gives meaning to something inactive, to hide.

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.



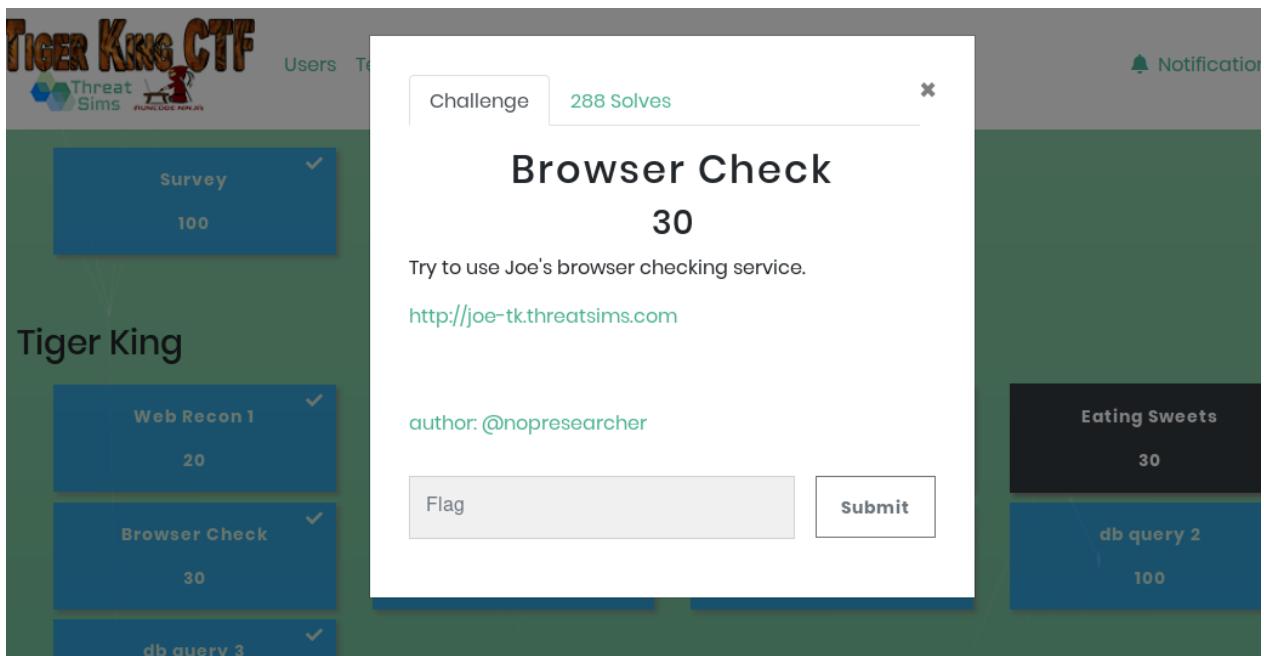
We went into Kali Linux to use the "**strings**" tool to extract all the strings from the image, which gave us a disordered string. Since the flags to be submitted start with

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



```
root@bismarck:~/Desktop# strings Taliger.jpg | grep derp
<rdf:li>derp{JoeMadeTaligers}</rdf:li>
root@bismarck:~/Desktop#
```

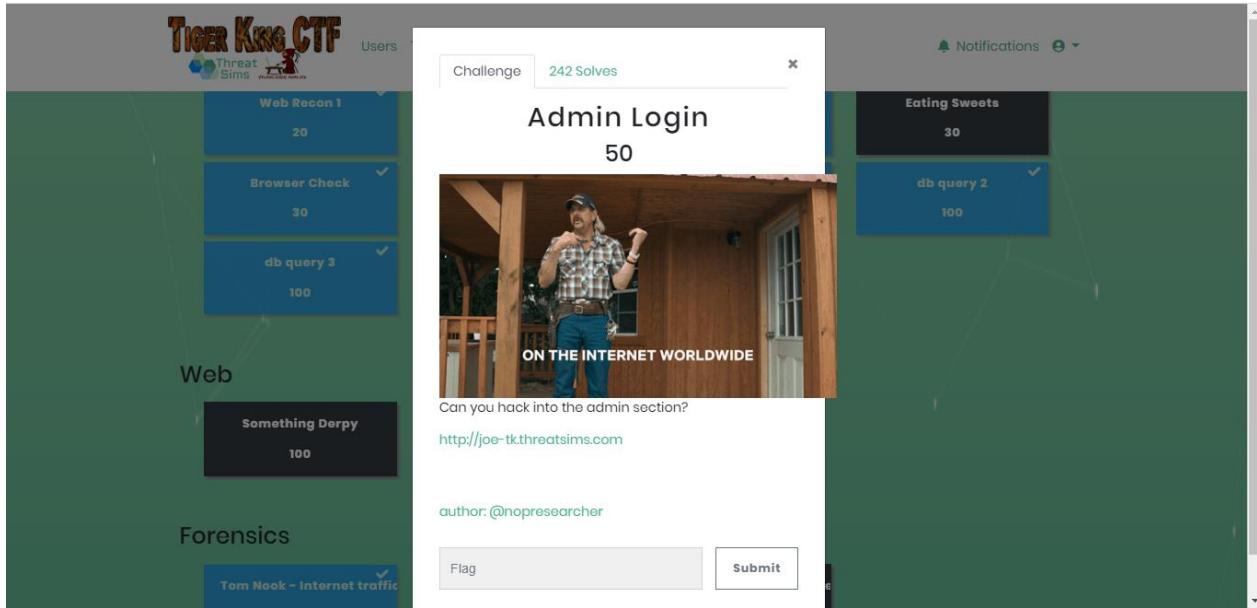
➤ Browser check



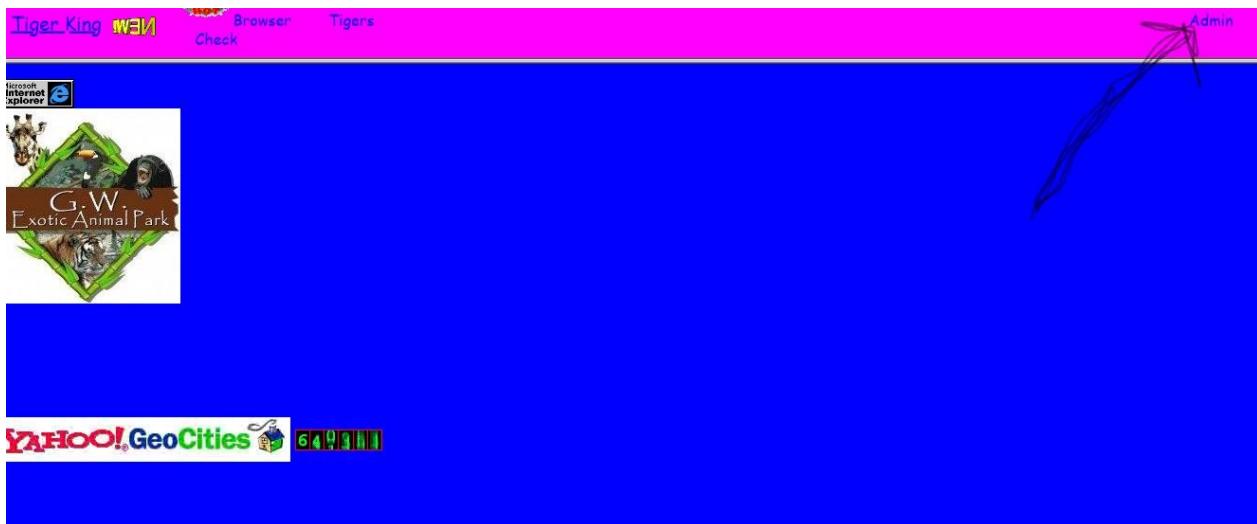
The screenshot shows the Tiger King CTF interface. On the left, there's a sidebar with various challenges: Survey (100 points), Web Recon 1 (20 points), Browser Check (30 points, highlighted in red), and db query 3. The main area displays the 'Browser Check' challenge details:

- Challenge:** 288 Solves
- Title:** Browser Check
- Value:** 30
- Description:** Try to use Joe's browser checking service.
- URL:** <http://joe-tk.threatsim.com>
- Author:** @nopresearcher
- Flag input field:** Flag
- Submit button:** Submit

➤ 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="" limit 0,1;
```

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



➤ db query 1

The screenshot shows the ThreatSims Tiger King CTF interface. On the left, there is a sidebar with challenges: "Survey" (100 points), "Tiger King" (20 points), and "Browser Check" (30 points). In the center, a challenge card for "db query 1" (50 points) is shown. The challenge text asks, "Is there anyone special looking for tigers?" and provides the URL "http://joe-tk.threatsims.com". It also credits the author as "@nopresearcher". There is a "Flag" input field and a "Submit" button. Below the challenge card, there are two more challenges: "Admin Login" (50 points) and "db query 2" (100 points). On the right, there is a "Notifications" section and a challenge card for "Eating Sweets" (30 points).

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.

The screenshot shows a web application interface. At the top, there is a search bar with the placeholder "Search" and a red "Search" button. Below the search bar, a message says "You know you love tigers!". The main content area displays a list of users in a table format:

joe	Age: 1223	Number of Tigers: 386
Antle	Age: 43	Number of Tigers: 386
kirkham	Age: 62	Number of Tigers: 0
dillon	Age: 22	Number of Tigers: 1
Carole	Age: 57	Number of Tigers: 0

At the bottom of the list, the text "Looking For Tigers" is displayed in green.

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

The screenshot shows the Tiger King CTF challenge interface for "db query 2". The challenge details are as follows:

- Name:** db query 2
- Value:** 100
- Description:** what is the length of the password field?
- Hint:** its sqlite3
- URL:** <http://joe-tk.threatsims.com>
- Author:** @nopresearcher

Below the challenge details, there is a "Flag" input field and a "Submit" button. To the right of the challenge box, there is a sidebar with other challenges listed:

- Eating Sweets (30)
- db query 2 (100)

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 –**

The screenshot shows a web application interface. At the top, there is a navigation bar with links for 'Logout', 'Joe Says thanks for logging in', and three 'hammer time' links. Below the navigation, a message says 'derp{JoeIsGladYouCameToSeeHisTigers} you should search for other tiger lovers!'. There is a 'Search' button and a text input field containing 'You know you love tigers!'. Below this is a search form with a 'Search by username here!' input field and a red 'Search' button. The main content area displays two user profiles:

User 1:
None
Age: 2
Number of Tigers: 3
Looking For Tigers

User 2:
Amtle
Age: 43
Number of Tigers: 386

At the bottom, there is a SQL command: `CREATE TABLE user (id INTEGER PRIMARY KEY, username VARCHAR(50) UNIQUE, age INTEGER, numTigers INTEGER, isLooking INTEGER, password VARCHAR(58))`. Below this, another profile is shown:

User 3:
Age: 2
Number of Tigers: 3
Looking For Tigers

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

The screenshot shows the Tiger King CTF interface. On the left, there's a sidebar with categories like Web Recon 1 (20), Browser Check (30), and db query 3 (100). The db query 3 section is highlighted with a blue background. In the center, a challenge box for "db query 3" is displayed, showing it has 104 solves. The challenge details ask for Joe's password and provide a URL: <http://joe-tk.threatsim.com>. The author is listed as @nopresearcher. Below the challenge box are "Flag" and "Submit" buttons. At the bottom of the page, there's a "Something Derby" button.

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.

The screenshot shows a list of users with their details. The users listed are dillon, dillon, joe, joe, kirkham, and kirkham. The user "joe" appears twice, with the second entry highlighted by a red box. The "joe" entries show the following details:
- First entry: Age: 2223, Number of Tigers: 386
- Second entry (highlighted): Age: 248b57c5cabbc9944d169d10bc4959a042d0bb81ab6cf9166f40a9d0f0fd614, Number of Tigers: 3
Both entries for "joe" are associated with the text "Looking For Tigers".

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

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

248b57c5cabbc9944d169d10bc4959a042d0bb81ab6cfc9166f40a9d0fd614

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(shai_bin)), QubesV3.1BackupDefaults

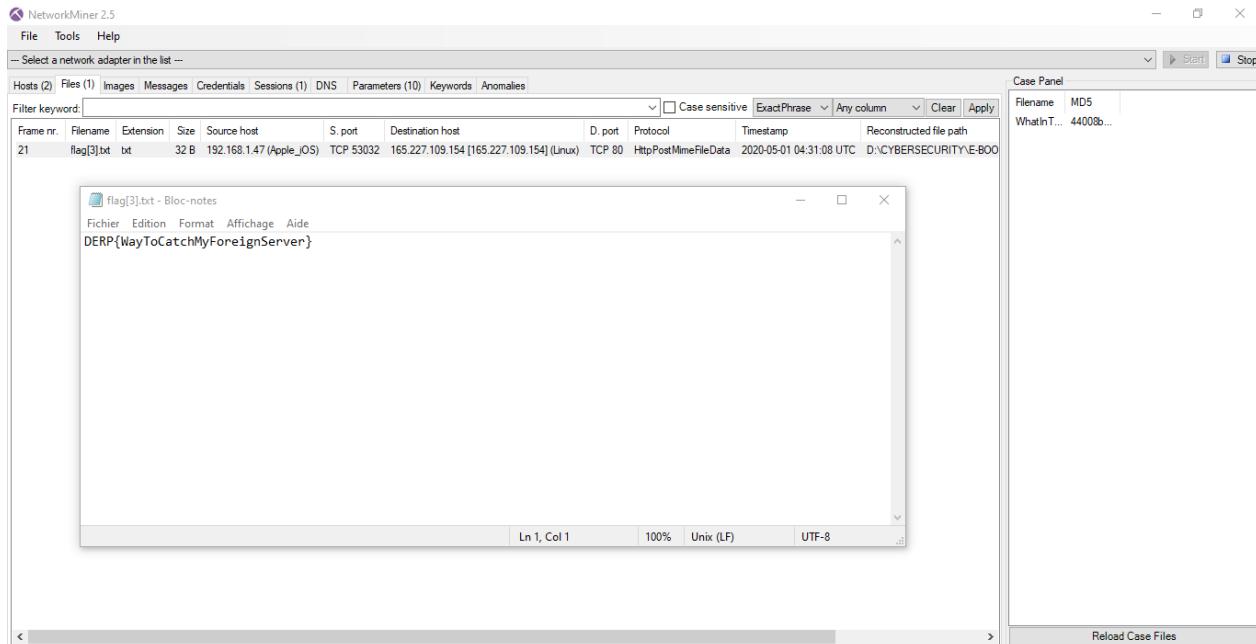
Hash	Type	Result
248b57c5cabbc9944d169d10bc4959a042d0bb81ab6cfc9166f40a9d0fd614	sha256	tigers

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

Forensics

➤ Tom Nook - Internet traffic - Part I

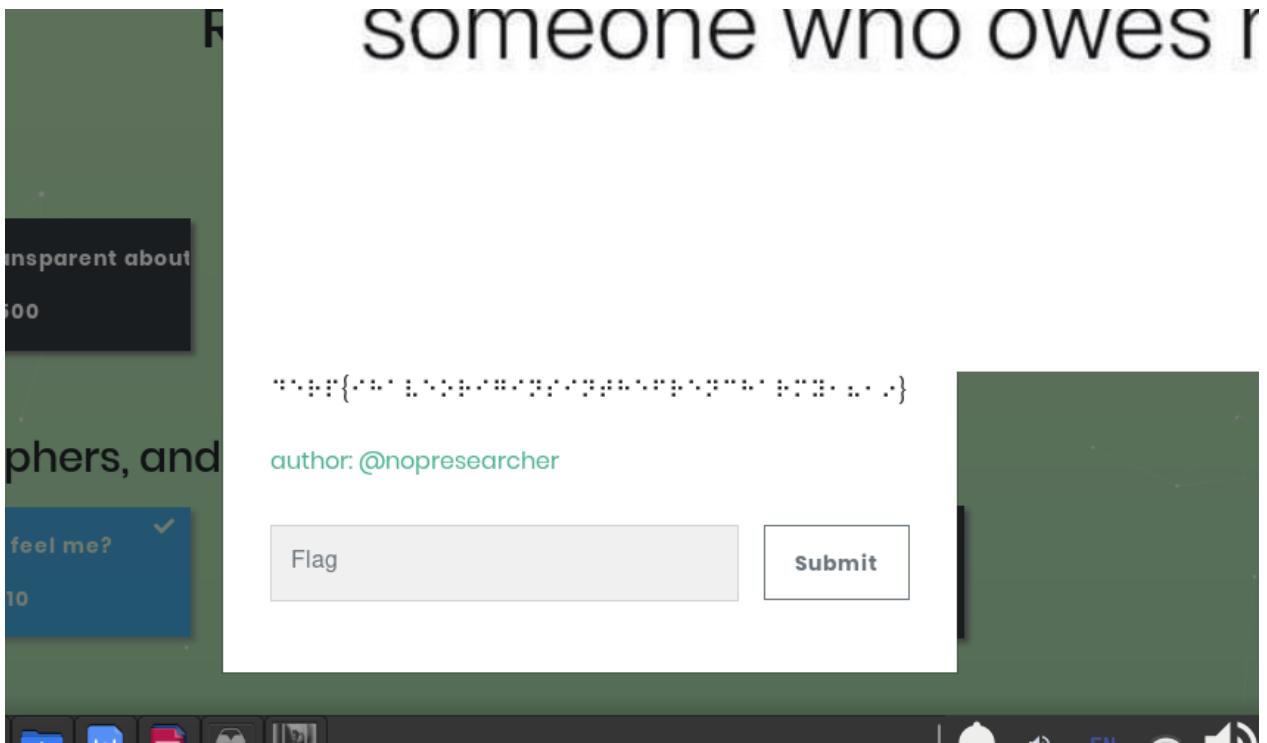
The test gives us a pcap file (**WhatInTheExfil.pcap**) to analyze. **NetworkMiner** which is an excellent tool to analyze pcap files. When we open, we see that our capture contains a file called flag.txt which is our wanted flag : **DERP{WayToCatchMyForeignServer}**.



Crypto, Ciphers, and Encodings

Challenge	Description	Points
Do you feel me? ✓	10	
Isabella suspicion	30	
[v3n3:v] ✓	200	
n Eggs	10	
All about that base ✓	10	
All about that base remix	10	
Two Streams	10	
et tu brute ✓	10	
Non-standard NetBIOS Name	10	
AFSC 29331	10	
Why are they even in that	10	
Don't touch the third rail	10	
Swag	20	
Mind the Padding	200	

➤ 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.

About 27,600 results (0.58 seconds)

".." (and any subsequent words) was ignored because we limit queries to 32 words.

qaz.wtf > braille ▾

Unicode Braille Maker - Unicode Braille Converter

... ; Braille multiletter alternates that might work: As a rule of thumb, you can't use multiletter codes if the letters for that are in different syllables within the ...

www.pharmabraille.com > pharmaceutical-braille > the-... ▾

The Braille Alphabet – PharmaBraille

m, n, 134. n, 1345. o, 135. p, 1234. q, 12345. r, 1235. s, 1234. t, 12345. u, 1235. u, 136 ... y, 13456 ... 1, 3456 1 ... 3, 3456 14. 4, 3456 145. 5, 3456 15. 6, 3456 124. 7, 3456 1245. 8, 3456 125. 9, 3456 24 ... Character, Braille, Braille dots. , ., 2. , : , 23. : , 25. , 256 ? , 236.

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}**

➤ [VIGENERE]

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/>

SECURE CODE WARRIOR			
SYDNEY Level 6, 227 Elizabeth Street Sydney, NSW 2000 Australia	SYDNEY Level 10, 179 Elizabeth Street Sydney, NSW 2000 Australia	BOSTON 745 Atlantic Ave Boston, MA 02111 USA	PORLAND 920 Southwest 6th Ave Portland, OR 97204 USA
LONDON 1 Fore St Ave London, United Kingdom EC2Y 9DT	BRUGES Baron Ruzettelaan 5 bus 3 8310 Brugge Belgium	REYKJAVIK Katrínartún 4 105 Reykjavík Iceland	

And so to decipher the flag we use the site <https://cryptii.com/> using the vigner cipher option and we get

Cryptii

IPVanish conceals your traffic so that your online activity can't be tracked.
ads via Carbon

VIEW Ciphertext ▾

vcuc{wcusurgmvcznpamu}

ENCODE DECODE Vigenère cipher ▾

VARIANT Standard Vigenère cipher

KEY Sydney

KEY MODE Repeat

ALPHABET abcdefghijklmnopqrstuvwxyz

CASE STRATEGY Maintain case FOREIGN CHARS Include Ignore

→ Decoded 23 chars

VIEW Plaintext ▾

derp{securecodewarrior}

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

➤ n Eggs

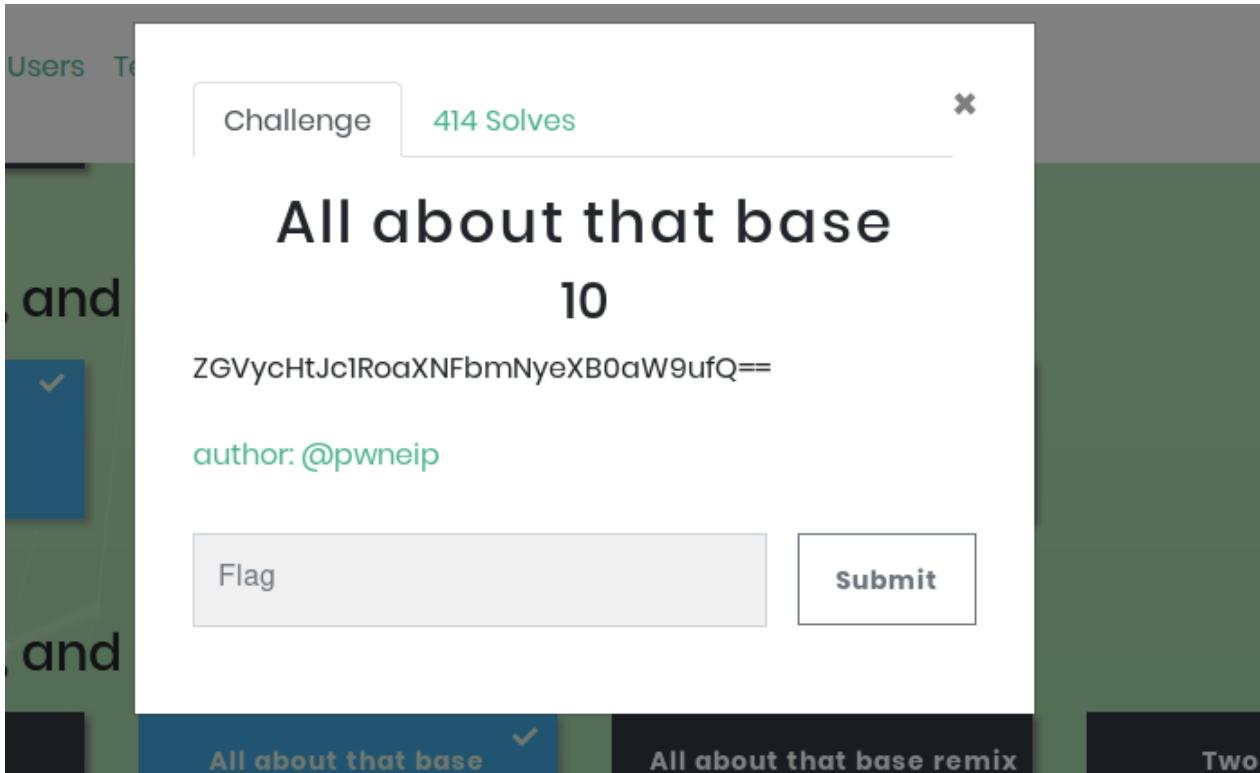
The screenshot shows the King CTF interface. On the left, there's a sidebar with a logo of a person at a desk, the text 'King CTF', 'reat ms', and 'Users'. Below this are two challenges: 'Do you feel me?' (10 points) and 'n Eggs' (10 points). On the right, the main challenge details for 'n Eggs' are shown. It has a difficulty rating of 10 and 299 solves. The challenge text reads: "AAABBAABAABAAABBBBB ABAAA ABABBABBBABABABAABAA AAAABAAAAAAAABAABBBAABBAB". The author is listed as '@pwneip'. There are 'Flag' and 'Submit' buttons. At the bottom, it says 'Non-standard NetBIOS Name' and 'AFSC 29331'. To the right, other challenges like 'Two Stream' (10 points) and 'Why are they e...' are partially visible.

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 screenshot shows the Cryptii platform interface. It has three main sections: 'Ciphertext' (containing the sequence 'AAABBAABAABAAABBBBB ABAAA ABABBABBBABABABAABAA AAAABAAAAAAAABAABBBAABBAB'), 'Encode Decode' (set to 'Bacon cipher'), and 'Plaintext' (containing the decoded message 'derpilovebacon'). The 'Encode Decode' section also shows a table for 'Unique codes for each letter' with rows for 'LETTER 1' (A) and 'LETTER 2' (B). An IPVanish advertisement is visible at the top right.

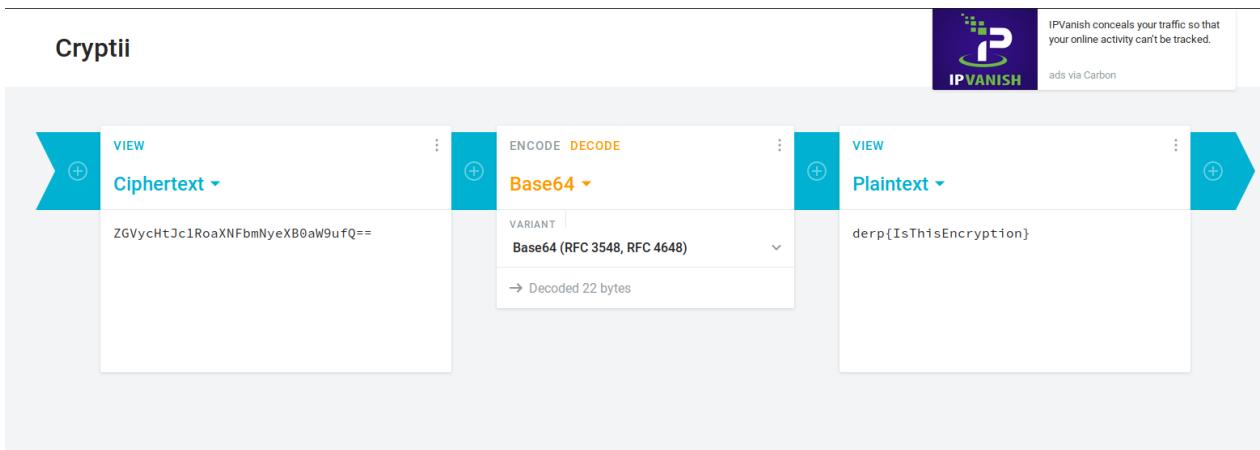
The flag is : **derp{ilovebeacon}**

➤ All about the base



The screenshot shows a challenge interface. At the top, it says "Challenge" and "414 Solves". The title of the challenge is "All about that base" with a difficulty rating of "10". Below the title is the encoded flag: "ZGVycHtJc1RoaXNFbmNyeXB0aW9ufQ==". The author of the challenge is listed as "author: @pwneip". There are two buttons at the bottom: "Flag" and "Submit". At the very bottom of the page, there are navigation links: "All about that base" (which is highlighted), "All about that base remix", and "Two".

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 screenshot shows the Cryptii web application interface. On the left, under "Ciphertext", the value "ZGVycHtJc1RoaXNFbmNyeXB0aW9ufQ==" is displayed. In the center, under "DECODE", the option "Base64" is selected. A dropdown menu for "VARIANT" shows "Base64 (RFC 3548, RFC 4648)". Below the dropdown, it says "Decoded 22 bytes". On the right, under "Plaintext", the decoded value "derp{IsThisEncryption}" is shown. The IPVanish logo is visible in the top right corner of the page.

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

➤ All about the base remix

The screenshot shows a challenge titled "All about that base remix" with a value of 10. The flag is MRSXE4D3KRUGS42JONCGKZSFNZRXE6LQORUW63RBPU===== and the author is @pwneip. There is a "Flag" input field and a "Submit" button. The challenge description includes the text "Do you feel me? 10" and "n Eggs 10". A sidebar on the right shows another challenge titled "Two Stream" with a value of 10.

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 screenshot shows the Cryptii interface with the ciphertext MRSXE4D3KRUGS42JONCGKZSFNZRXE6LQORUW63RBPU===== being decoded from Base32 to Plaintext. The resulting plaintext is derp{ThisIsDefEncryption!}.

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

➤ Two streams

The screenshot shows a challenge card for 'Two Streams' worth 10 points. The challenge details are as follows:

- Challenge:** Two Streams
- Points:** 10
- Flag:** asuo{TaoDadqtfnLukRsrvhbWyuiLop}
- Author:** @pwneip
- Buttons:** Flag, Submit

The background of the challenge card features a green banner with the text 'Foto, Ciphers, and' repeated twice. Below the banner, there are three boxes labeled 'Eggs' (10), 'All about that base' (10), and 'All about that base remix' (10). To the right, a partial view of another challenge card is visible.

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.

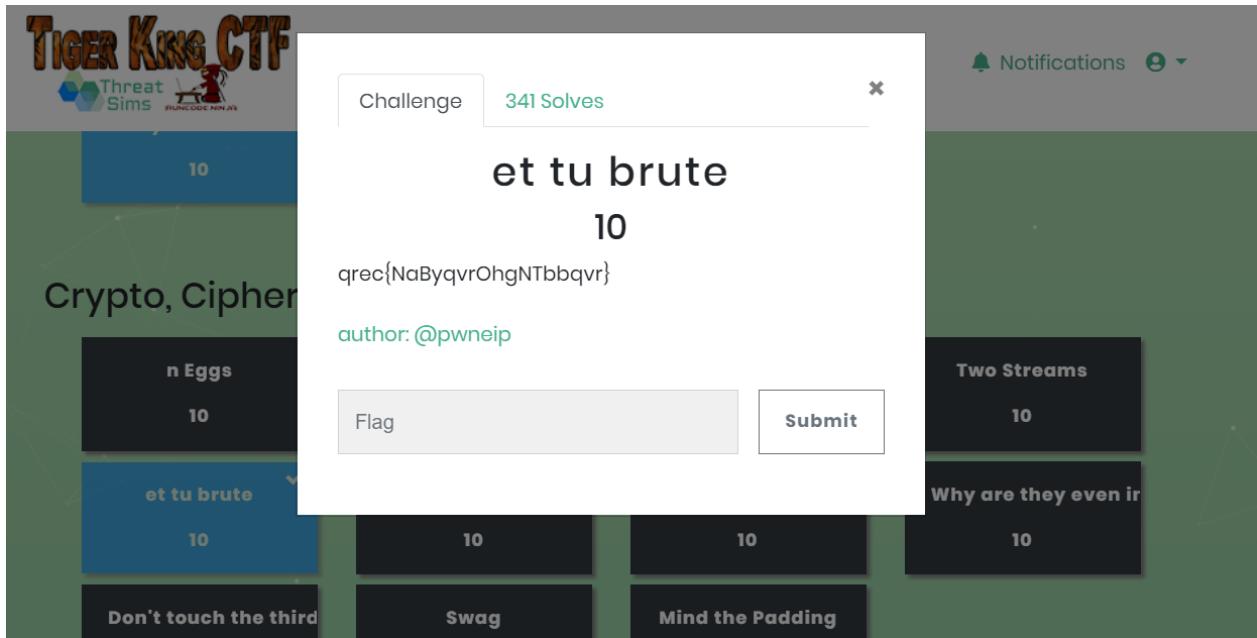
The screenshot shows the Cryptii interface with the following steps:

- Ciphertext:** asuo{TaoDadqtfnLukRsrvhbWyuiLop}
- Encode/Decode:** Bifid cipher
- Plaintext:** derp{TwoStreamsAreBetterThanOne}

The interface includes a sidebar with the IPVanish logo and an advertisement for Carbon.

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**.

A screenshot of a Caesar cipher decoder tool. On the left, there's a search bar and a results section with a list of decrypted words for different offsets. The word "derp" is highlighted with a blue circle and a hand-drawn arrow pointing to it. In the center, there's a banner for "IA SCHOOL - L'ÉCOLE DE L'INTELLIGENCE ARTIFICIELLE". Below it, there's a section titled "DÉCHIFFREMENT DU CODE CÉSAR" with a text input field containing "qrec". On the right, there's a sidebar with links related to Caesar cipher decoding and encoding.

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

The screenshot shows a web-based Caesar cipher decryption tool. The input text is 'NaByqvrOhgNTbbqvr'. The output text, highlighted in blue, is 'AnOldieButAgoodie'. Above the output, the offset is indicated as '+13'. The tool interface includes fields for 'CONNAISSANT LE DÉCALAGE:' (set to 3) and 'TESTER TOUS LES DÉCALAGES POSSIBLES (ATTAQUE BRUTE-FORCE)'. There are also sections for 'DÉCHIFFRER LE CODE CÉSAR' and 'CHIFFREMENT PAR CODE CÉSAR'.

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

➤ AFSC 29331

The screenshot shows a challenge titled 'AFSC 29331' with a value of 10. The challenge text is: '-...-.-./-...- - -/-... - - -.-.-.-.-...'. The author is listed as '@pwneip'. Below the text is a 'Flag' input field and a 'Submit' button. The challenge is part of a category labeled 'All about that base'.

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

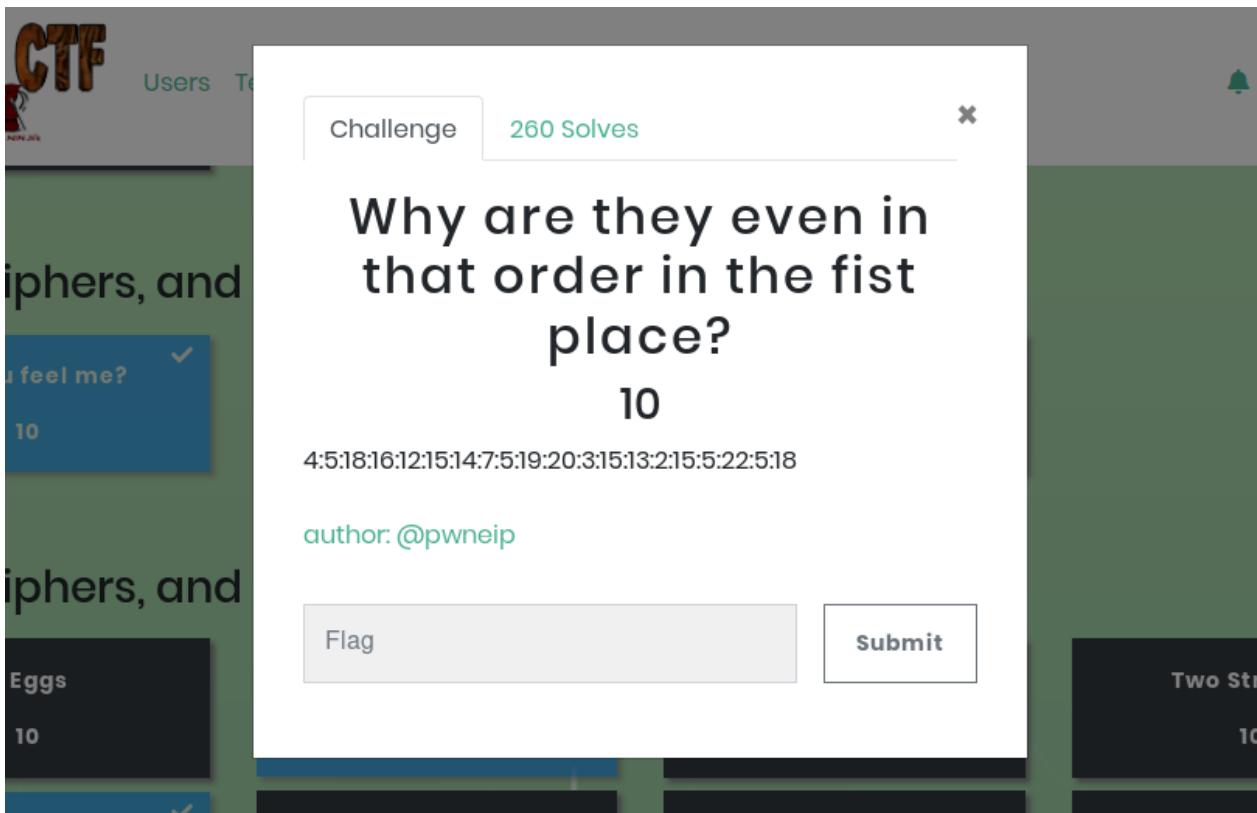
The screenshot shows a web application for decoding Morse code. It consists of three main sections arranged horizontally:

- Ciphertext:** A box containing the Morse code sequence: `.... . .-. .-.-. - . .- - -.- | ... - -- .-- .-.- . .- ..`.
- Morse code settings:** A central box titled "ENCODE DECODE" with "Morse code" selected. It includes dropdown menus for "VARIANT" (set to "English") and "REPRESENTATION" (set to "Code"). Below these are tables for "SHORT", "LONG", and "SPACE" symbols, each with a corresponding character or symbol. A message at the bottom states "→ Decoded 16 chars".
- Plaintext:** A box displaying the decoded text: "derpdittyboppers".

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

Challenge 196 Solves

Don't touch the third rail

10

d{zir}epZgaCpeFwricht

author: @nopresearcher

Flag Submit

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 fence code and therefore to decipher the flag we used the <https://cryptii.com/> using the Rail Fence option

Ciphertext

Rail fence cipher

KEY: 3
OFFSET: 8
→ Decoded 21 chars

Plaintext

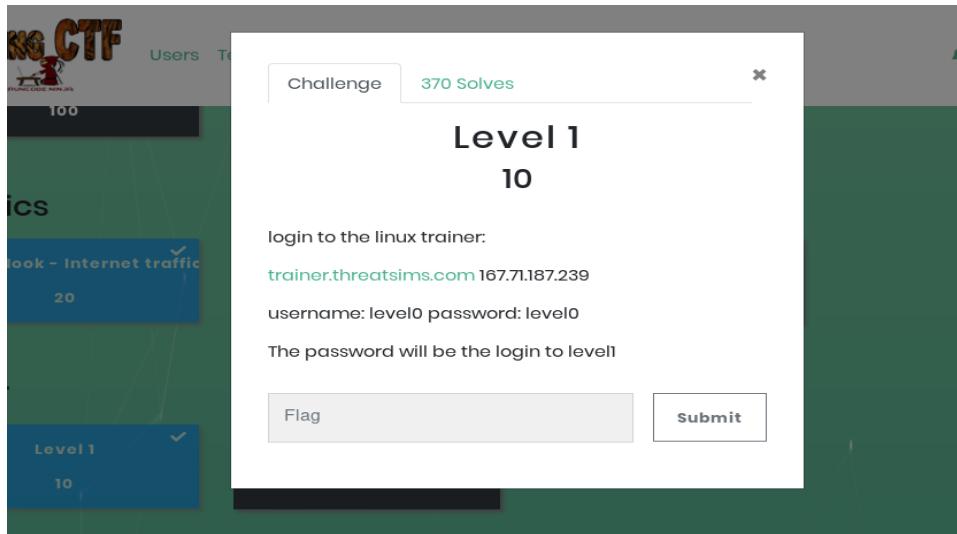
Vigenère cipher: Encrypt and decrypt online

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

Trainer

Trainer			
Level 1 10	Level 22 10		
Level 2 10	Level 3 10	Level 4 10	Level 5 10
Level 6 10	Level 7 10	Level 8 10	Level 9 10
Level 10 10	Level 11 10	Level 12 10	Level 13 10
Level 14 10	Level 15 10	Level 16 10	Level 17 10
Level 18 10	Level 19 10	Level 20 10	Level 21 10

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.

```
connection to 167.71.187.239 closed.
root@Bizaske:~# ssh level1@167.71.187.239
level1@167.71.187.239's password:

root@Bizaske:~# ssh level0@167.71.187.239
level0@167.71.187.239's password:
Linux trainer 4.19.0-8-cloud-amd64 #1 SMP Debian 4.19.98-1+deb10u1 (2020-04-14) 
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May  1 18:54:14 2020 from 47.8.179.18

=====
|   .--.
|   o_o |
|   / \  |
|   (   ) |
|   \_ \_/
=====
Welcom to the Linux Trainer!
To view the level instructions again
type: cat welcome_message
Getting Stuck? Need help with a level?
type: cat helptme
Use "su - level#" to change levels
feedback: noprerearcher@gmail.com

=====
|   .--.
|   o_o |
|   / \  |
|   (   ) |
|   \_ \_/
=====
Welcome to Level 0

The password for the next level is in this user's home directory

level0@trainer:~$ ls
helptme level1_password welcome_message
level0@trainer:~$ cat level1_password
4202c26842398c1d0772ed9eed195113
level0@trainer:~$
```

The flag is : **4202c26842398c1d0772ed9eed195113**

➤ 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.

```
permitted by applicable law.
Last login: Fri May 1 18:58:26 2020 from 47.8.179.18

=====
|   .--.
|   | o_o | |
|   || == |
|   | \_ ) |
|   | / ( \
|   \_ \_) |
=====
Welcom to the Linux Trainer!
To view the level instructions again
type: cat welcome_message
Getting Stuck? Need help with a level?
type: cat helpme
Use "su - level#" to change levels
feedback: noprerearcher@gmail.com

=====
Welcome to Level 2

The password for the next level is in this user's home directory, but you have to dig even

level2@trainer:~$ ls
a.out  dir  helpme  welcome_message
level2@trainer:~$ ./a.out
level2@trainer:~$ cd dir/
level2@trainer:~/dir$ ls
another_dir
level2@trainer:~/dir$ cd another_dir/
level2@trainer:~/dir/another_dir$ ls
another_another_dir
level2@trainer:~/dir/another_dir$ cd another_another_dir/
level2@trainer:~/dir/another_dir/another_another_dir$ ls
some_directory
level2@trainer:~/dir/another_dir/another_another_dir$ cd some_directory/
level2@trainer:~/dir/another_dir/another_another_dir/some_directory$ ls
level3_password
level2@trainer:~/dir/another_dir/another_another_dir/some_directory$ cat level3_password
2cadca6148093c403d82396252b8c4db

level2@trainer:~/dir/another_dir/another_another_dir/some_directory$ █
```

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 have seen a cache file named **.level4_password** which contains the flag and password of the user level4.

The terminal window shows the following session:

```
type: cat welcome_message
Getting Stuck? Need help with a level?
type: cat helpme
Use "su - level#" to change levels
feedback: noprerearcher@gmail.com
=====
Welcome to Level 3
The password for the next level is in this user's home directory, but you might
type: man ls      read about files that start with a dot (.)
level3@trainer:~$ ls
dot  helpme  welcome_message
level3@trainer:~$ cat helpme
Run the ls command to view a directory listing.
There doesn't appear to be anything, read the man page for ls.
ls has a flag for viewing files that start with a .
These files are not typically listed when you perform a ls.
To see the 'hidden' file type ls -a
Then use cat to view the contents of the file.
Commands:
ls
man ls
ls -a
cat .level4_password
level3@trainer:~$ cat .level4_password
72f6af6b0005adb15fbc91e1b140115f
level3@trainer:~$
```

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 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 terminal window shows a challenge from 'Trainer 5'. The text in the terminal is as follows:

```
Feedback: n0p3r3arch3r@gmail.com
=====
Repository: https://github.com/n0p3r3arch3r/project-intermediate
=====
Welcome to Level 4

The password for the next level is in this user's home directory, just like
type: man ls      read about files and folders that start with a dot (.)

level4@trainer:~$ ls
helpme welcome_message
level4@trainer:~$ cat helpme
Run the ls command to view a directory listing.
There doesn't appear to be anything, read the man page for ls.
ls has a flag for viewing files that start with a .
These files are not typically listed when you perform a ls.
To see the 'hidden' directory type ls -a
cd into the 'hidden' directory and then perform another ls -la.
Then use cat to view the contents of the file

Commands:
ls
man ls
ls -a
cd .hidden_dir
cat .level4_password

level4@trainer:~$ cd .hidden_dir/
level4@trainer:~/._hidden_dir$ cat .
cat: ..: Is a directory
level4@trainer:~/._hidden_dir$ cat .level5_password
7b6c2552940f47a27fdbd729ae0e2893c

level4@trainer:~/._hidden_dir$
```

The flag is : `7b6c2552940f47a27fdbd729ae0e2893c`

➤ 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.

```
7210a || 70005a001010c>1c101401101  
7b6c2 || 2940F47a27fbd729ae0e2893c  
=====  
  
Welcome to Level 5  
  
For this level the password is in level6's home directory. Due to a persn  
rns.  
  
level5@trainer:~$ ls  
helpme welcome_message  
level5@trainer:~$ cat helpme  
You know that it is in level6's home directory. Let's start with an ls of  
d file. Notice the permissions of the file, the group is level5, which al  
  
Commads:  
ls -la /home/level6  
cat /home/level6/level6_password  
  
level5@trainer:~$ ls -la  
total 52  
drwxr-x--- 5 level5 level5 4096 May  1 18:39 .  
drwxr-xr-x 26 root   root   4096 Mar  5 10:06 ..  
-rw-r--r--  1 level5 level5  220 Apr 18  2019 .bash_logout  
-rwxrwxrwx  1 level5 level5 3526 Apr 18  2019 .bashrc  
-rw-r--r--  1 level5 level5    0 May  1 15:35 .cloud-locale-test.skip  
drwx-----  3 level5 level5 4096 Mar  6 15:29 .config  
drwx-----  3 level5 level5 4096 May  1 15:23 .gnupg  
-r--r----  1 level5 level5  354 Mar 24 13:27 helpme  
-rw-----  1 level5 level5   44 Mar  6 15:41 .lesshist  
drwxr-xr-x  3 level5 level5 4096 Dec 30 12:30 .local  
-rwxr-xr-x  1 level5 level5  807 Apr 18  2019 .profile  
-rw-----  1 level5 level5 4138 May  1 18:39 .viminfo  
-r--r----  1 level5 level5  221 Mar 24 13:27 welcome_message  
level5@trainer:~$ cat /home/level6/level6_password  
7cb1963d316b9a302cf6c204d35b7302
```

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.

```
./dir9/subdir8:  
./dir9/subdir9:  
level6@trainer:~$ ls  
2 dir10 dir13 dir16 dir19 dir21 dir24 dir27 dir3 dir32 dir35 dir38 dir40 dir43 dir46 dir49 dir6 dir9 helpme te welcome_message  
8 dir11 dir14 dir17 dir2 dir22 dir25 dir28 dir30 dir33 dir36 dir39 dir41 dir44 dir47 dir5 dir7 file.txt level6_password temp  
dir1 dir12 dir15 dir18 dir20 dir23 dir26 dir29 dir31 dir34 dir37 dir4 dir42 dir45 dir48 dir50 dir8 find level7pass test  
level6@trainer:~$ cat helpme  
Run the ls -la command to view a directory listing. There are too  
many directories and sub-directories to dig through to find the file.  
Reading the man page for find, we are searching for a file named  
password. find has an option -name where you can specify the name  
of the file. All we know is that there is password in the title, so  
we are going to wildcard the beginning and end of password with *  
  
Commands:  
ls -la  
find . -name *password*  
find . | grep password  
cat ./dir13/subdir40/level7_password  
level6@trainer:~$ cat ./dir13/subdir40/level7_password  
The password for level7 is:  
RG8geW91IGV2ZW4gbGlmdCBlc8g
```

The flag is : **RG8geW91IGV2ZW4gbGlmdCBlc8g**

➤ 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**.

```

level7@trainer:~$ ls
cat helptme password_directory welcome_message
level7@trainer:~$ cat helptme
cd into the password_directory and run ls -la to view a directory listing. There are 100 files all the same size, one of them contains the password to the next level. There are multiple ways to solve this solution.

grep:
Use grep to search through all the files

Commands:
cd password_directory
grep pass level8_password*

find, xargs and grep:
Use find to get a list of every single file in the directory. This can then be piped to xargs. Xargs is used to build the arguments to be passed to a utility. The utility that the arguments are going to be passed to is grep. Grep is used to search each file for the phrase password. So to put it all together, find will output a list of files found in the directory, xargs will then pass them one at a time to grep to be searched through for the phrase password. One benefit of using xargs is that the file name is printed when password is found

Commands:
cd password_directory
find . | xargs grep password
cat ./level8_password68

find and grep:
Find has a primary -exec that will execute an utility on each result. For each result, which is a file grep will search for the phrase password. When using the -exec primary a {} is used as a placeholder for each filename that will then be passed to grep. The last piece of -exec is that it must end with a ;, a \ is used to escape the semicolon. The only issue is that it will not list the file name where the phrase was found. Using the -H option with grep will print the filename with the result.

Commands:
cd password_directory
find . -exec grep -H password {} ;
cat ./level8_password68

level7@trainer:~$ cat ./level8_password68
cat ./level8_password68: No such file or directory
level7@trainer:~$ cd password_directory/
level7@trainer:~/password_directory$ cd password_directory/
-bash: cd: password_directory/: No such file or directory
level7@trainer:~/password_directory$ cat ./level8_password68
The password for level8 is:
bGV0J3MgZmluzCBzb21ldGhpbmcg

```

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.

```

Use "su - level9" to change levels
feedback: noprerearcher@gmail.com
=====
Welcome to Level 8
For this level the password is in an executable hidden in one of these sub-directories. When you run the executable it will print out the flag. To run the executable type: ./executable
type: man find
After finding and executing the binary, you should try to run the command strings on the it and review the output. type: strings <executable>
level8@trainer:~$ ls
desktop dir11 dir14 dir17 dir2 dir22 dir25 dir28 dir30 dir33 dir36 dir39 dir41 dir44 dir47 dir5 dir7 helpme      s
dir1   dir2   dir5 dir18 dir20 dir23 dir26 dir29 dir31 dir34 dir37 dir4   dir42 dir45 dir48 dir49 dir50 dir6 log
dir10 dir13 dir16 dir19 dir21 dir24 dir27 dir3   dir32 dir35 dir38 dir40 dir43 dir45 dir46 dir49 dir5   dir9 ''$\001'P'$'\020''@h9@8' welcome_message
level8@trainer:~$ cat helptme
For this level you are looking for an executable program that will print out the password to the next level. Just like in Level 6 there are multiple layers of directories. If you read the man page for find, you will see there is a -executable flag for find. This will search for directories and files that are executable by the user running them.

Commands:
find . -executable
./level9_password

level8@trainer:~$ ./d
desktop dir11/ dir17/ dir2/ dir22/ dir25/ dir28/ dir30/ dir33/ dir36/ dir39/ dir41/ dir44/ dir47/ dir5/ dir7/
dir1/ dir12/ dir15/ dir18/ dir20/ dir23/ dir26/ dir29/ dir31/ dir34/ dir37/ dir4/ dir42/ dir45/ dir48/ dir50/ dir8/
dir10/ dir13/ dir16/ dir19/ dir21/ dir24/ dir27/ dir3/ dir32/ dir35/ dir38/ dir40/ dir43/ dir46/ dir49/ dir6/ dir9/
level8@trainer:~$ ./desktop
The password is: 96ab15e954f126ea04c35de2d771c2b
level8@trainer:~$ 

```

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 terminal window shows the following text:

```
Welcome to Level 9
To view the level instructions again
type: cat welcome_message
Getting Stuck? Need help with a level?
type: cat helme
Use "su - level9" to change levels
feedback: noprereacher@gmail.com
```

On the right, there is a password entry form with the number '10' above it, a 'Flag' input field, and a 'Submit' button.

```
-----
```

For this level we want to find the line number in the rock you wordlist where the password "evilhacker" is found. That line number is the password for level 10. The wordlist can be found at /usr/share/wordlists/rockyou.txt (it's in the same place on kali too) Grep uses a 1-based numbering system meaning the first line is 1 and not 0.

type: man grep

```
level9@trainer:~$ ls
DEADJOE helme rockyou.txt welcome_message
level9@trainer:~$ cat helme
For this level you are given an very popular password list and asked to find where in the list a password is. The tool to use is grep to find the password in the file. To figure out what line number the password is at, you can use the -n flag to display the line number.

Commands:
grep -n evilhacker /usr/share/wordlists/rockyou.txt
level9@trainer:~$ grep -n evilhacker /usr/share/wordlists/rockyou.txt
955830:evilhacker
level9@trainer:~$
```

At the bottom, there is a file manager icon bar and a status bar showing '01 mai 2020' and battery level '46%'.
01 mai 2020

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 terminal window shows the following text:

```
Welcome to Level 10
For this level you are given a log file from the program fail2ban. Fail2ban is used monitor log files for suspicious activity like too many failed logins. It is commonly deployed for use with Apache or SSH. After a configured number of attempts it will create an iptables (Linux firewall) rule to block the IP from communicating with the device for a period of time.

The log file is located in your home directory and is called fail2ban.log. The password to level 11 is the number of times 112.85.42.94 was banned.

type: man grep
type: man wc
```

For this challenge we need to use grep like we did on the previous level to find the IP 112.85.42.94. However, grepping for just the IP results in over 1100 lines. Looking at the logs we see that the IP is listed numerous times for each time it is found in a log, when it is banned and unbanned. We need to find out how many times it was banned. On the log entries where it is banned we notice that the word Ban is in front of the IP. We can simply grep for 'Ban 112.85.42.94' and get every log entry where the IP was banned. Now, we have the problem of too many log entries to count. Let's use a | (pipe) and the command wc (word count). A | (pipe) is used to pass the output of one command into another command like word count. We need the -l flag on wc so that it will count the number of lines.

Commands:
grep 'Ban 112.85.42.94' fail2ban.log | wc -l

```
level10@trainer:~$ grep 'Ban 112.85.42.94' fail2ban.log | wc -l
192
```

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}” md5find**. This allowed us to display the flag.

```
level11@trainer:~$ cat welcome_message
Welcome to Level 11

For this level you are given a file that contains the password to the next level. The password is a md5 hash. Research md5 hashes and find it in the file.

type: man grep
      man md5sum
      google md5 hash

level11@trainer:~$ ls
helpme  md5  md5find  md5.py  output.txt  s  test.py  test.txt  um  wc  welcome_message
level11@trainer:~$ cat helpme
For this level we are going to use grep to search through the file md5find. This file is 1983 lines long (wc -l md5find). You can use less (less md5find) to look through file and that there are too many lines to visually pick out a md5hash. An md5 hash is typically expressed as a 32 digit hexadecimal number. So we are trying to something that is 32 digits long and hexadecimal. Hexadecimal is 0-9 and a-f. For grep we need to build a regular expression that will match this.

[0-9a-f] ensures the character will only match hexadecimal (values of 0-9 and a-f)
{32} will use the above match 32 times, the length of a md5 hash. The \ are used to escape the { and }

Commands:
grep -e "[0-9a-f]{32}" md5find
level11@trainer:~$ grep -e "[0-9a-f]{32}" md5find
0982e28a6985764a074d06b1a4fd1bea
level11@trainer:~$
```

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.

```

/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/umount
/usr/bin/mount
/usr/bin/newgrp
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/mysecret
/home/level12/find
level12@trainer:~$ ls
cat find helme null outputter.txt outputting.txt welcome_message
level12@trainer:~$ cd cat
-bash: cd: cat: Not a directory
level12@trainer:~$ ./cat
f4736e1eb28b1d9055c5f5d58a49b5a6level12@trainer:~$

```

The flag is : **f4736e1eb28b1d9055c5f5d58a49b5a6**

➤ 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.

```

level13@trainer:~$ ls
helpme mypassword.txt welcome_message
level13@trainer:~$ cat mypassword.txt
f4736e1eb28b1d9055c5f5d58a49b5a6level13@trainer:~$ cat helpme
For this level we are going to use the env command. The user has their AWS credentials set as environment variables. It is very common to see credentials in environment variables in cloud deployments and containers.

Commands:
env
env | grep ID
level13@trainer:~$ env
SHELL=/bin/bash
HISTTIMEFORMAT=%Y-%m-%d %T
AWS_DEFAULT_REGION=us-west-2
PWD=/home/level13
LOGNAME=level13
XDG_SESSION_TYPE=tty
HOME=/home/level13
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=01;33:cd=40;33:01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.tbz=01;31:*.tz=01;31:*.deb=01;31:*.tar.gz=01;31:*.tar.bz2=01;31:*.tar.xz=01;31:*.tar.lz=01;*.tar.lzo=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.rz=01;31:*.cab=01;31:*.swm=01;31:*.dwm=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.jpg=01;35:*.mng=01;35:*.mpg=01;35:*.avi=01;35:*.mpe=01;35:*.mkv=01;35:*.webm=01;35:*.ogg=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.mov=01;35:*.mp4=01;35:*.rmvb=01;35:*.rm=01;35:*.flc=01;35:*.avi=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ovr=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*
AWS_SECRET_ACCESS_KEY=2F3dadasswJalxs2UtnFEMis/GKMDENG/bPxRfIcy
SSH_CONNECTION=196.182.146.89 49452 167.71.187.239 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=level13
SHELL=/bin/bash
AWS_ACCESS_KEY_ID=0ea027e3835aa87a4a47465321c5fe75
XDG_SESSION_ID=3467
XDG_RUNTIME_DIR=/run/user/1014
SSH_CLIENT=196.182.140.88 49452 22
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
MAIL=/var/mail/level13
SSH_TTY=/dev/pts/76
/_usr/bin/env
level13@trainer:~$ env | grep ID
AWS_ACCESS_KEY_ID=0ea027e3835aa87a4a47465321c5fe75
XDG_SESSION_ID=3467
level13@trainer:~$

```

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.



```
permitted by appricotice.com
Last login: Fri May 1 19:32:32 2020 from 187.190.166.175
=====
Welcome to the Linux Trainer!
=====
To view the level instructions again
type: cat welcome_message
Getting Stuck? Need help with a level?
type: cat helptext
Use "su - level#" to change levels
feedback: noprereacher@gmail.com
=====
Level 15
10
Please password to level
Play Submit
=====
Welcome to Level 14
For this level you are going to familiarize yourself with the kernel version. We are just looking for the Kernel and Major version (the first two sets of numbers) example: if the version is 2.62 .26.1 the password will be 2.62
Understanding Kernel versions can help when search for exploits with tools like searchsploit or exploitdb (Sorry, there isn't any kernel exploits for this box, I hope)
type: man uname
      google linux kernel
=====
level14@trainer:~$ ls
helptext  welcome_message
level14@trainer:~$ cat helptext
For this level we are going to use the uname command. There are multiple different ways to get this information. You can also look in the /proc directory as well.
Commands:
uname -a
cat /proc/version
ls /boot
level14@trainer:~$ uname -a
Linux trainer 4.19.0-8-cloud-amd64 #1 SMP Debian 4.19.98-1+deb10u1 (2020-04-27) x86_64 GNU/Linux
level14@trainer:~$
```

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.

```
Welcome to Level 15
For this level you are going to familiarize yourself with the distro version. We are just looking for the distro name. example: Fedora 31, then the password would be Fedora
Understanding distro versions can help when searching for exploits with tools like searchsploit or exploitdb (Sorry, there isn't any exploits for this distro, I hope)
type: man hostnamectl
      man lsb_release
      google linux distro
```

Trainer

Level 17
10

The password to level

```
level15@trainer:~$ ls
helpme  welcome_message
level15@trainer:~$ cat helpme
For this level there are multiple different ways to get this information. You can also look in the /etc directory as well.
When searching through searchsploit or exploitdb there are exploits that are only present in a specific version of that software packaged in that distro.
```

Commands:

```
lsb_release -a
hostnamectl
```

```
cat /etc/issue
cat /etc/*-release
```

```
level15@trainer:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
level15@trainer:~$ hostnamectl
  Static hostname: trainer
    Icon name: computer-vm
      Chassis: server
      Machine ID: h23d5936bb6b6596488fb97c5de15439
        Boot ID: 20371398e48f4b10814bcd8163767476
      Virtualization: kvm
  Operating System: Debian GNU/Linux 10 (buster)
            Kernel: Linux 4.19.0-8-cloud-amd64
          Architecture: x86-64
level15@trainer:~$ ^C
```

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.

```
2160  /'` \` type: cat welcome_message
2161  b6bc  { [ ] ) } Getting Stuck? Need help with a level?
2162  681e  \`_`-(_` type: cat helpme
2163  54f1207fe040c350e20 Use "su - level#" to change levels
2164  92 feedback: nopreresearcher@gmail.com
2165  6982  069857644074d06b1a4f010ed
2166  4731  02801d90556575058400506
2167  =====
2168
Welcome to Level 16
For this level you are going to familiarize yourself with the aliases. They can be very useful and can be used for a variety of actions to speed up your work.
S.

type: google alias

level16@trainer:~$ ls
helpme  welcome_message
level16@trainer:~$ cat helpme
For this level it is pretty straight forward, just use the common alias to see the available aliases. Aliases are super useful, however in this case they are pass allows you to specify your password as an argument to the command. This can be useful when scripting tasks for CTFs, however it exposes the password. B
sshpass.

Commands:
alias

cat .bashrc

level16@trainer:~$ alias
alias bc='bc -l'
alias devbox='sshpass -p 6b39034a8045ed996a436f8d09031522 ssh level17@trainer.threatsim.com'
alias grep='grep --color=auto'
alias ls='ls -color:auto'
alias mkdir='mkdir -pv'
```

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
Attention, vous utilisez la commande less pour visualiser ce fichier.
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!
RG8geW91LGV2Zw4gbGlmdCBicm8g
# Viminfo version
|1,4

# Value of 'encoding' when this file was written
*encoding=utf-8
0982e2a88985f644074d06b1a4fd1bea
F4736e1eb28b1d9055c5f5d58a49b5a6
# hlsearch on (H) or off (h): 75
~h
# Command Line History (newest to oldest):
:wq
|2,0,1583196421,, "wq"
9a42b1822710d790a393800f2896a8f7

# 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.

Welcome to Level 18

For this level you are going to continue familiarizing yourself with user artifacts. Look in this user's home directory to see if there are any files left behind that may contain user artifacts.

type: man bash

Level 18

10

The password to level

Flag:

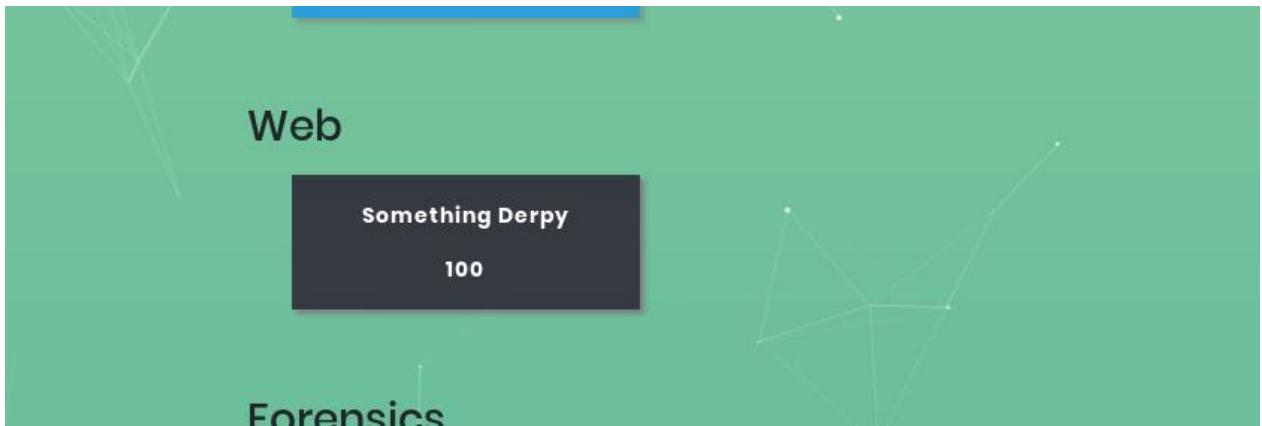
Submit

```
level18@trainer:~$ ls
helpme welcome_message
level18@trainer:~$ cat helpme
This level is focused on user artifacts and in particular bash artifacts. When commands are executed in a bash shell, they are written to a .bash_history file. The file is ASCII text and can be viewed with less. Depending on the implementation there will be variations in the file format, but you will see all the commands entered at the command prompt. You can get a sense of what a user has been doing by looking at their bash history.
Commands:
less ~/.bash_history
cat ~/.bash_history

level18@trainer:~$ cat ~/.bash_history
whoami
uname -a
id
ps -ef
netstat -an
cat /proc/version
cat /etc/*-release
pwd
ls -la
cat /etc/profile
cat ~/.bashrc
awk '{F = "/tmp/login/{print $1}'; /etc/passwd
find / -perm -4e -type f 2>/dev/null
ssh level19@localhost
b05a246b0646b337f319316b9232151c
whoami
ssh level19@127.0.0.1
pwd
ls -la
level18@trainer:~$
```

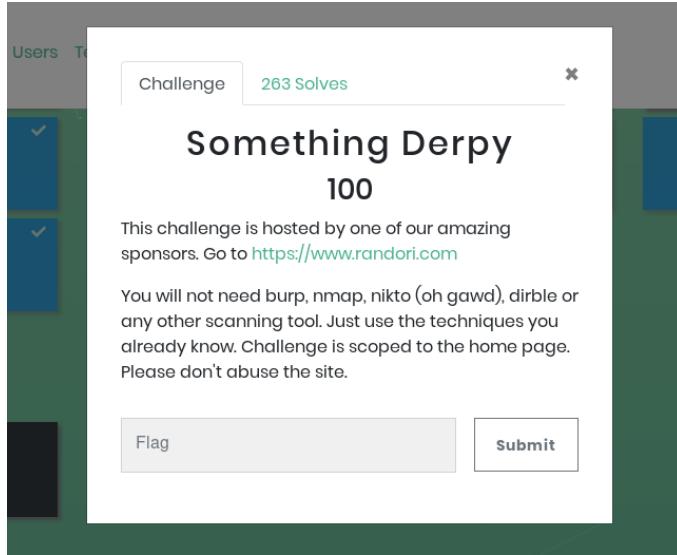
The flag is : **b06a246b0646b337f319316b9232151c**

WEB



➤ Something Derpy

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



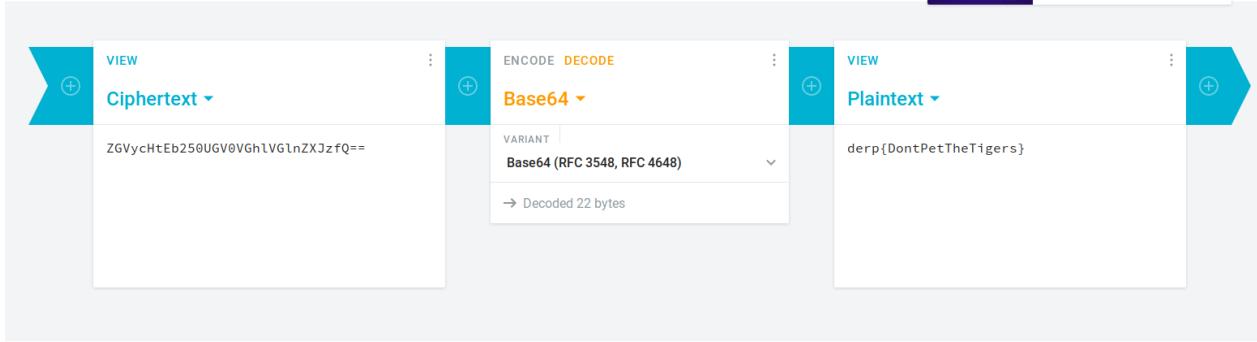
challenge partner. The inspiration of the source page allowed us to come across a message accompanied by an image.

```
529 <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>
530 </nav>
531
532 <section class="c-footer__legal">
533   <center>
534     <a href="/bart"></a> <!-- something is derpy with this image, will fix later -->
535   </center>
536   <div class="o-grid u-clearfix">
537     <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>
538     <li class="c-footer__legal-item"><a href="https://www.randori.com/terms-of-service/">Terms of Use</a></li>
539     <li class="c-footer__legal-item">Scopy; 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:

```
G%cK
/+((
ys49<
U?A)—M
h_7c
k4kW
/,,|
I-:~
h4q-
.mgi
IEND
ZGVycHtEb250UGV0VGh1VGlnZXJzfQ=
root@Bizasko:~/Téléchargements#
```

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

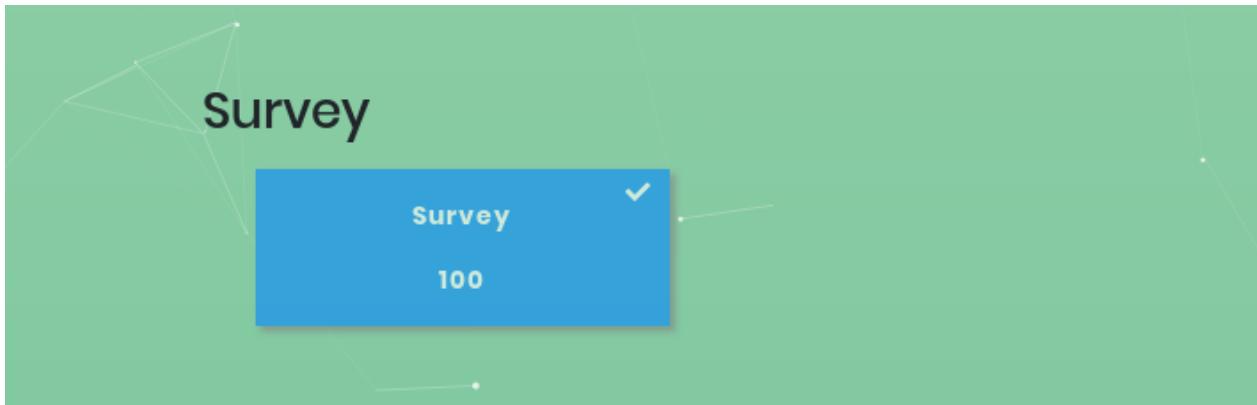


The screenshot shows the Cryptii interface with three main sections: Ciphertext, Encode/Decode, and Plaintext.

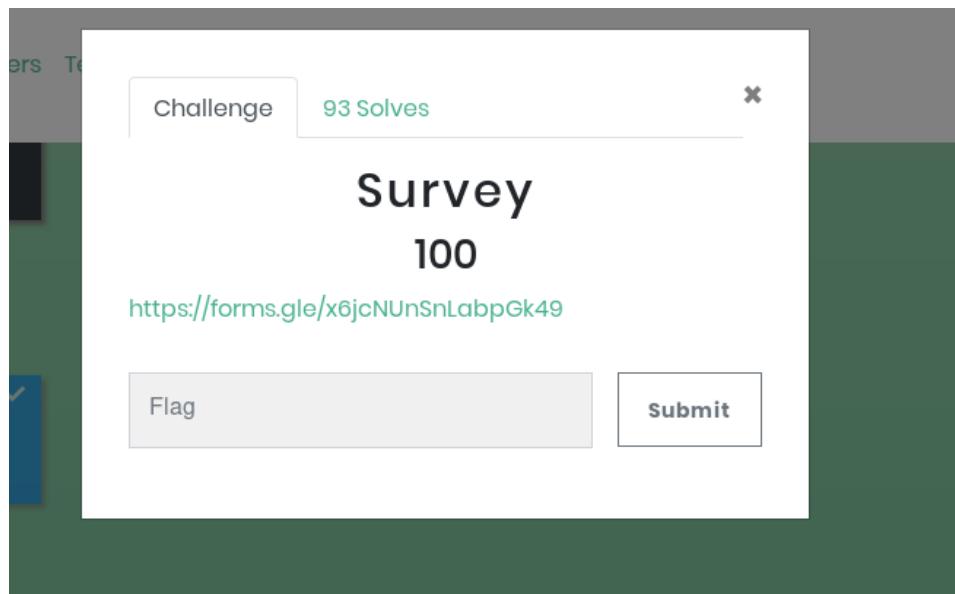
- Ciphertext:** Shows the encoded text `ZGVycHtEb250UGV0VGhlVGlnZXJzfQ==`.
- Encode/Decode:** Set to **Base64**.
 - VARIANT:** Set to **Base64 (RFC 3548, RFC 4648)**.
 - Decoded:** Shows the decoded text `derp{DontPetTheTigers}`.
- Plaintext:** Shows the decoded text `derp{DontPetTheTigers}`.

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

Servvey



➤ Servvey



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

The screenshot shows a Google Form interface. At the top, it says "DerpCon CTF Survey". Below that, it says "Flag for scoreboard: `derp{ThanksForTakingTheSurvey}`". There is a blue link "Envoyer une autre réponse". At the bottom left, there is a note in French: "Ce formulaire a été créé dans Threat Simulations. [Signaler un cas d'utilisation abusive](#)". The Google Forms logo is at the bottom right.

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