Write up for siNUsoid CTF competition

SAURABH KUMAR [Organizer and Creator]

Email : geniwazir@gmail.com
Discord ID : 612694716969910272 [Geni_Wazir#0731]

Q1

Name of the file states that it have some relation to ROCKYOU and we all know that ROCKYOU.TXT is quite famous wordlist. And for a image password is required when we do stenography which makes it clear that we have to do stenography using ROCKYOU.TXT.

For this I am using `stegcracker` a python based tool for bruteforce using steghide.

The command will be as follow

To install stegcracker

```
pip3 install stegcracker
```

To use the format is

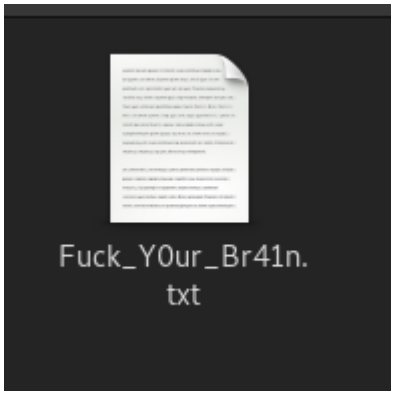
```
stegcracker <stego_image> <wordlist>
```

```
┌──(geniwazir㉿kali)-[ [redacted] ]
└─$ stegcracker R0ckyou.jpg [redacted] /rockyou.txt
StegCracker 2.1.0 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2021 - Luke Paris (Paradoxis)

StegCracker has been retired following the release of StegSeek, which
will blast through the rockyou.txt wordlist within 1.9 second as opposed
to StegCracker which takes ~5 hours.

StegSeek can be found at: https://github.com/RickdeJager/stegseek
Counting lines in wordlist..
Attacking file 'R0ckyou.jpg' with wordlist '[redacted]/rockyou.txt'..
Successfully cracked file with password: gangster
Tried 708 passwords
Your file has been written to: R0ckyou.jpg.out
gangster
```

Here we have successfully cracked the stenography password

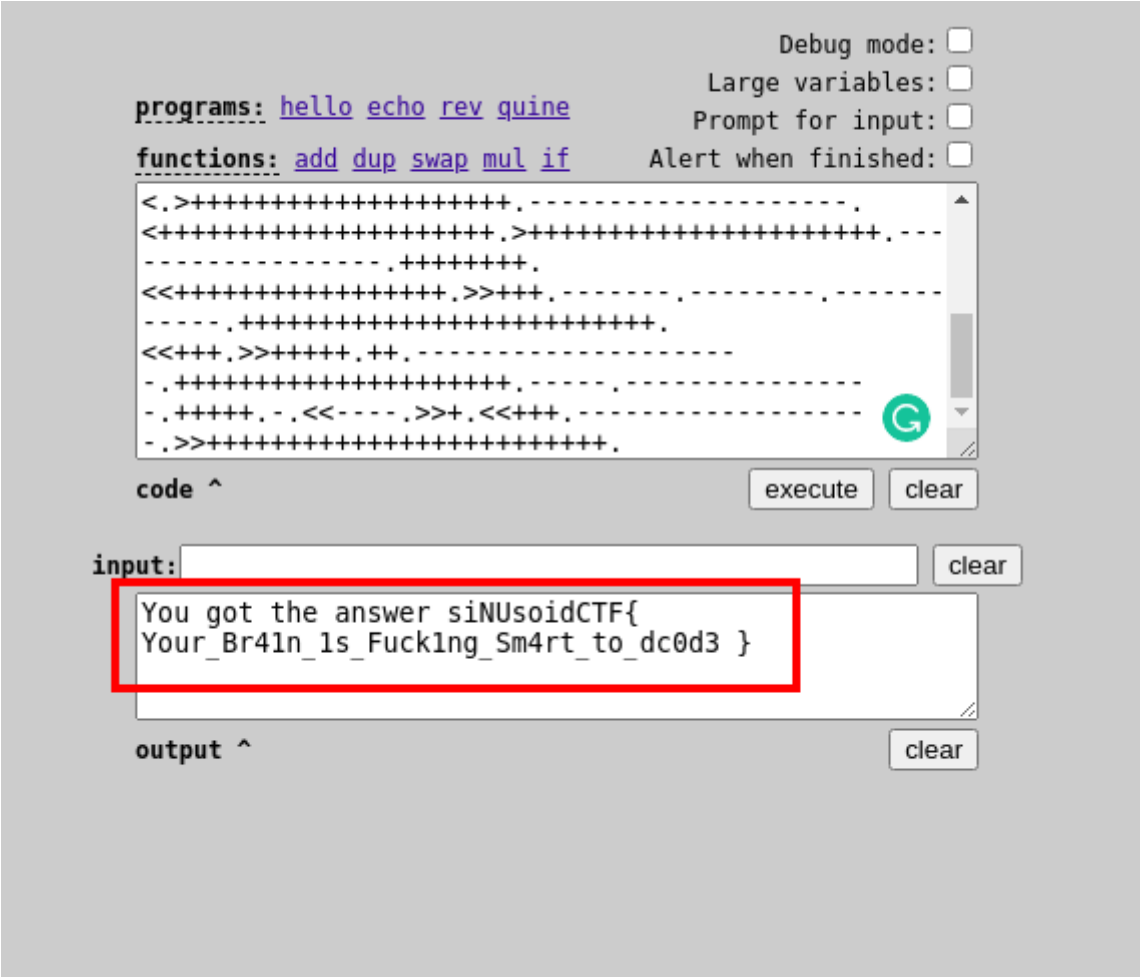
```
┌──(geniwazir㉿kali)-[ [redacted] ]
└─$ ls
R0ckyou.jpg  R0ckyou.jpg.out

┌──(geniwazir㉿kali)-[ [redacted] ]
└─$ cat R0ckyou.jpg.out
siNUsoidCTF{ Rocky0u_1s_Am3e2Ing_F0r_Brute_F0rc3 }

┌──(geniwazir㉿kali)-[ [redacted] ]
└─$
```

Q2

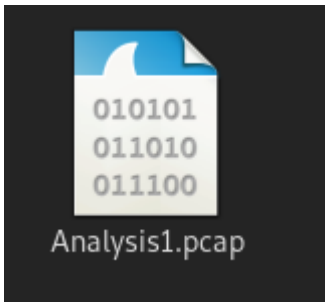



```
++++++++++[>+>+++>+++++++>++++++++++<<<<-]>>>>----------
-.++++++++++++++++++++++.------.<<++.>>----------.++++++++.+++++.<<.>>+-
----------.---.<<.>>-----.+++++++++++++.+++++.++++.------------------
-.+++++++++++++.<<.>>+.----------.<++++++++.+++++++.>++++++++++.-----.------
-.------.<------------------.>----------------.
<+++.>++++++++++++++++++++++++++++++++++++++++.<<.>++++++++++++++++++++.>------
-------.++++++.---.<++++++.--------------------------------.>.<---------------+-
--.>-----.---------------.<.>++++++++++++++++++++.--------------------.
<++++++++++++++++++++++.>++++++++++++++++++++++++.------------------.++++++++.
<<++++++++++++++++++.>>+++.-------.--------.----------
-.++++++++++++++++++++++++++++.<<+++.>>+++++.++.----------------------
-.++++++++++++++++++++++.------.----------------.+++++-.<------.>>+.<<+++.-----
----------.>>++++++++++++++++++++++++++.
```

This is a famous encryption method named as brain fuck encryption and the name of the file also provides the hint for that



Q3



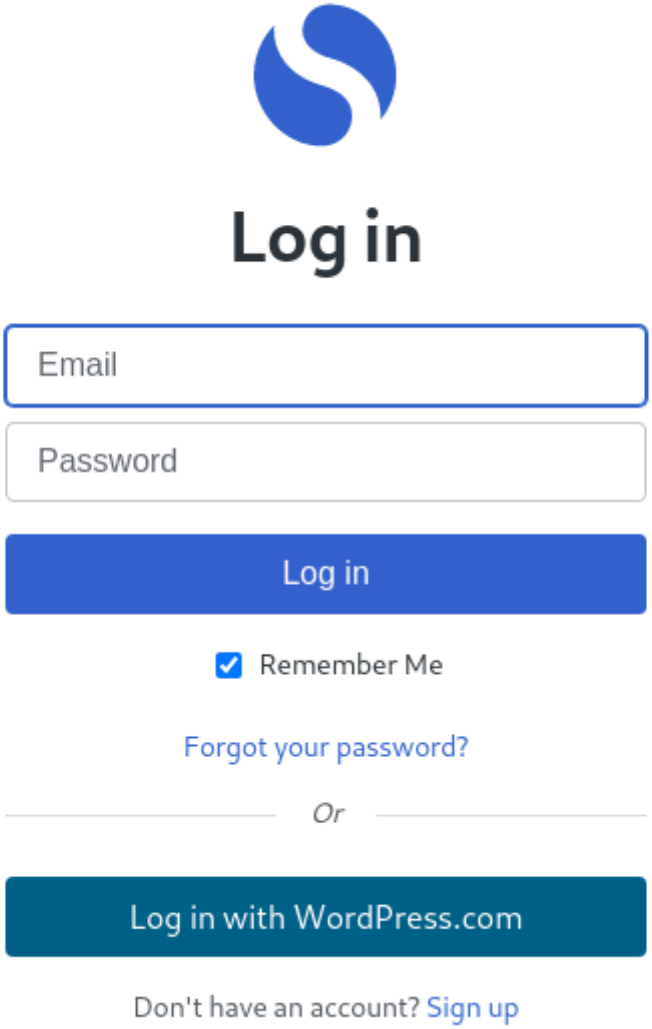
A pcap file was provided which means we will be requiring the wireshark and the name suggest that we need to do some sort of analysis.

367	9.054097	54.154.116.64	192.168.228.79	TLSv1.2	4093	Application Data, Application Data
368	9.054106	192.168.228.79	54.154.116.64	TCP	68	46136 → 443 [ACK] Seq=3030 Ack=306313 Win=518912 Len=0 TSval=3455552545 TSecr=226415335
369	14.179053	192.168.228.79	192.168.228.79	TCP	76	35206 → 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=4078175006 TSecr=0 WS=128
370	14.179073	192.168.228.79	192.168.228.79	TCP	76	80 → 35206 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=4078175006 TSecr=4078175006
371	14.179091	192.168.228.79	192.168.228.79	TCP	68	35206 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4078175006 TSecr=4078175006
372	14.179250	192.168.228.79	192.168.228.79	HTTP	622	GET /flag.txt HTTP/1.1
373	14.179276	192.168.228.79	192.168.228.79	TCP	68	80 → 35206 [ACK] Seq=1 Ack=555 Win=65024 Len=0 TSval=4078175006 TSecr=4078175006
374	14.179801	192.168.228.79	192.168.228.79	HTTP	502	HTTP/1.1 200 OK (text/plain)
375	14.179818	192.168.228.79	192.168.228.79	TCP	68	35206 → 80 [ACK] Seq=555 Ack=435 Win=65152 Len=0 TSval=4078175007 TSecr=4078175007
376	14.973241	192.168.228.79	192.168.228.79	HTTP	622	GET /flag.txt HTTP/1.1
377	14.973292	192.168.228.79	192.168.228.79	TCP	68	80 → 35206 [ACK] Seq=435 Ack=1109 Win=65024 Len=0 TSval=4078175800 TSecr=4078175800
378	14.973849	192.168.228.79	192.168.228.79	HTTP	501	HTTP/1.1 200 OK (text/plain)
379	14.973870	192.168.228.79	192.168.228.79	TCP	68	35206 → 80 [ACK] Seq=1109 Ack=868 Win=65152 Len=0 TSval=4078175801 TSecr=4078175801
380	18.478448	203.17.244.51	192.168.228.79	TCP	68	[TCP Dup ACK 269#1] 80 → 36055 [ACK] Seq=1 Ack=1 Win=11 Len=0 TSval=2989461767 TSecr=1276938800

We can see that there is a request to a file called flag.txt over HTTP and we can access its data which is this

```
▸ Frame 374: 502 bytes on wire (4016 bits), 502 bytes captured (4016 bits)
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 192.168.228.79, Dst: 192.168.228.79
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 35206, Seq: 1, Ack: 555, Len: 434
▸ Hypertext Transfer Protocol
▾ Line-based text data: text/plain (1 lines)
    have a look on https://app.simplenote.com may you find somthing or analys harder for the flag \n
```

The statement mentioned an URL which is this `https://app.simplenote.com` and if we visit the URL we are prompted with an login screen



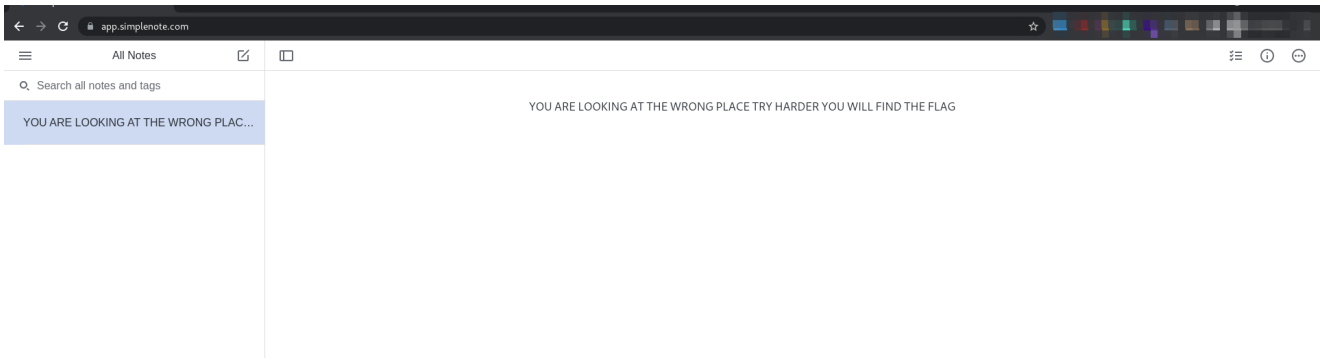
So going back to the wireshark and after doing some more analysis we found a POST request over HTTP which seems to be a jackpot

465	43.466858	203.17.244.51	192.168.228.79	TCP	68	[TCP Previous segment not captured] 80 → 36055 [ACK] Seq=2 Ack=1 Win=11 Len=0 TSval=2989486769 TS
466	45.111994	192.168.228.79	74.125.200.188	TCP	68	[TCP Dup ACK 1#1] 50676 → 5228 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2888377210 TSecr=2117194935
467	45.299802	74.125.200.188	192.168.228.79	TCP	68	[TCP Dup ACK 2#1] [TCP ACKed unseen segment] 5228 → 50676 [ACK] Seq=1 Ack=2 Win=265 Len=0 TSval=2
468	46.245896	192.168.228.79	44.228.249.3	HTTP	754	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
469	46.938227	44.228.249.3	192.168.228.79	TCP	68	80 → 50972 [ACK] Seq=2749 Ack=1165 Win=61568 Len=0 TSval=3170180684 TSecr=3340287214
470	46.938228	44.228.249.3	192.168.228.79	HTTP	344	HTTP/1.1 302 Found (text/html)
471	46.938271	192.168.228.79	44.228.249.3	TCP	68	50972 → 80 [ACK] Seq=1165 Ack=3025 Win=64128 Len=0 TSval=3340287906 TSecr=3170180685
472	46.947476	192.168.228.79	44.228.249.3	HTTP	585	GET /login.php HTTP/1.1
473	47.764532	44.228.249.3	192.168.228.79	TCP	68	80 → 50972 [ACK] Seq=3025 Ack=1682 Win=61056 Len=0 TSval=3170181374 TSecr=3340287916
474	47.764533	44.228.249.3	192.168.228.79	HTTP	2816	HTTP/1.1 200 OK (text/html)
475	47.764580	192.168.228.79	44.228.249.3	TCP	68	50972 → 80 [ACK] Seq=1682 Ack=5773 Win=63104 Len=0 TSval=3340288733 TSecr=3170181375
476	48.166988	44.228.249.3	192.168.228.79	HTTP	1408	[TCP Spurious Retransmission] Continuation

If we look to that packet we get username and password

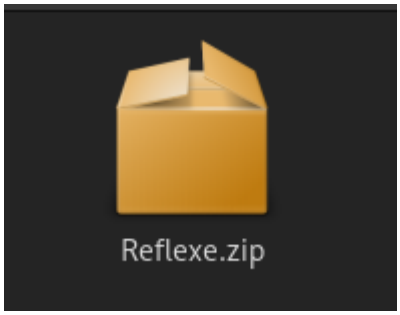
```
▸ Frame 468: 754 bytes on wire (6032 bits), 754 bytes captured (6032 bits)
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 192.168.228.79, Dst: 44.228.249.3
▸ Transmission Control Protocol, Src Port: 50972, Dst Port: 80, Seq: 479, Ack: 2749, Len: 686
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "uname" = "kejaga4460@erpipo.com"
  ▸ Form item: "pass" = "an0ther$tr0ngp@ssw0rd"
```

Trying this credentials to the login page that we got previous and we successfully login into the account.

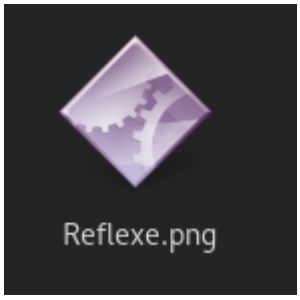


After figuring out the website we saw a trash section and when we visit that BOOOOM.

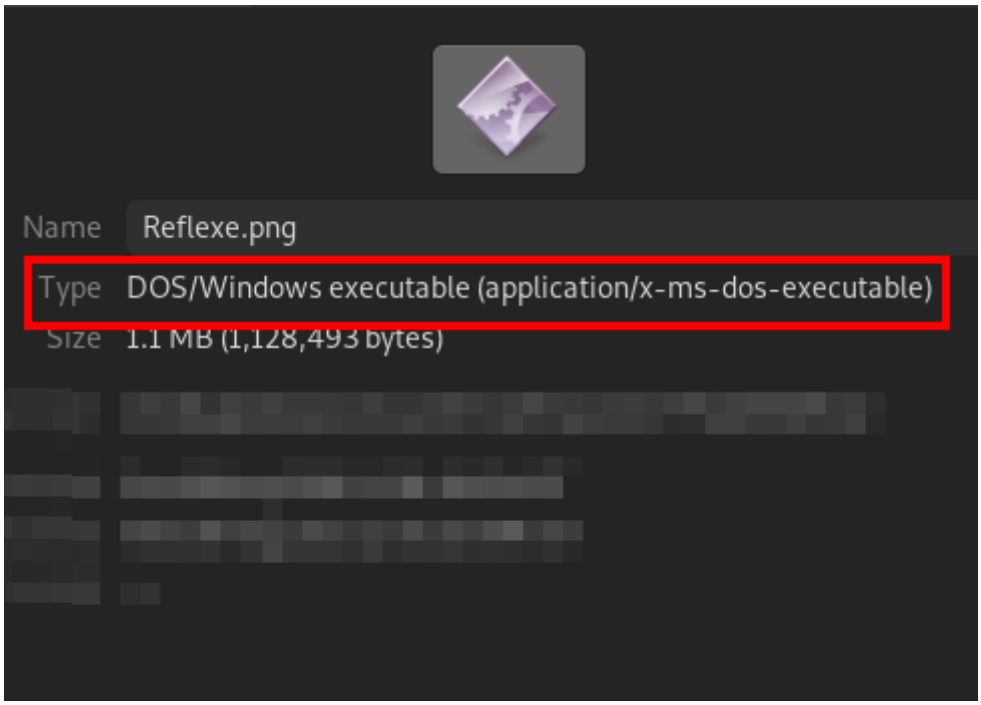
Q4



When we extract the zip file we get a file named `Rexlexe.png`



png is a image extension and so the user will think of something related to stenography and all. But if we check its property then it's an windows executable file buts its extension has been spoofed.



So now we know that it's an executable so first thing that we should do is *strings*.

```
┌──(geniwazir@kali)-[                                    ]
└─$ strings Refl<202e>gnp.exe
!This program cannot be run in DOS mode.
RichR
.text
`.rdata
@.data
.rsrc
@.reloc
u/;u
9=tXL
=tXL
5xXL
=lXL
=lXL
5dXL
%hXL
%dXL
t\HH
\SVW
D$(P
%hXL
%dXL
D$(P
WSPV
_^[]
~LWS
Iu-V
#E(VW
99u1
_^[]
98u[
uaSVW
t$8]4t
Pj0V
9^Xt=9^\tE
QQSVW
=4XL
54XL
54XL
98u#h
;Gxs
;Gds
^j|Xf
50XL
=DXL
=HXL
%@XL
(SVWh
8V:t
tC~)
uVWQ
@^_]
@_^[
=DXL
```

We got lot of data. So, to check if the flag is present in that we will use grep.

```
┌──(geniwazir@kali)-[
└─$ strings Refl<202e>gnp.exe | grep CTF
siNUsoidCTF{ R3vers1ng_is_crA2y_Th1nG_t0_d0 }
```

Q5

In this question 2 file are provided PYTHON and SECRET.

If we see the PYTHON file we get this


```
aW1wb3J0IHJlLCB1dWlkLCBiYXNlNjQKZGVmIGVuY3J5cHRpb24obXNnKToKICAgIG5ld19tc2cg
PSAiIgogICAgZW5jcnlwdCA9ICIiCiAgICBmb3IgY2hhcmFjdGVyIGluIG1zZzoKICAgICAgICBu
ZXdfbXNnID0gbmV3X21zZyArIGNocigob3JkKGNoYXJhY3RlcikrMSkpCiAgICBtZXNzYWdlX2J5
dGVzID0gbmV3X21zZy5lbmNvZGUoJ2FzY2lpJykKICAgIGJhc2U2NF9ieXRlcyA9IGJhc2U2NC5i
NjRlbmNvZGUobWVzc2FnZV9ieXRlcykKICAgIG5ld19tc2cgPSBiYXNlNjRfYnl0ZXMuZGVjb2Rl
KCdhc2NpaScpCiAgICB4ID0gKChyZS5maW5kYWxsKCcuLicsICclMDEyeCcgJSB1dWlkLmdldG5v
ZGUoKSkpKSoobGVuKG5ld19tc2cpLy81KQogICAgZm9yIGkgaW4gcmFuZ2UobGVuKG5ld19tc2cp
LTIpOgogICAgICAgIGVuY3J5cHQgPSBlbmNyeXB0ICsgbmV3X21zZ1tpXSArIHhbaV0KICAgIHBy
aW50KGVuY3J5cHQrJz09Jyk=
```


We can conclude that this is a base64 encoded massage and when we decode it we get a python code which is

```
import re, uuid, base64
def encryption(msg):
```

```
    new_msg = ""
    encrypt = ""
    for character in msg:
        new_msg = new_msg + chr((ord(character)+1))
    message_bytes = new_msg.encode('ascii')
    base64_bytes = base64.b64encode(message_bytes)
    new_msg = base64_bytes.decode('ascii')
    x = ((re.findall('..', '%012x' % uuid.getnode())))*(len(new_msg)//5)
    for i in range(len(new_msg)-2):
        encrypt = encrypt + new_msg[i] + x[i]
    print(encrypt+'==')
```

We can see what the code is doing it is taking msg as an argument and then looping through each character converting it into its equivalent ASCII value and incrementing it by 1 and again converting it back into the string with new ASCII value and storing it into new_msg. Then it is encoding the new_msg into base64 .
Now in the x variable in the code is just fetching the MAC address and multiplying it with the result of length of the new base64 encoded massage divided by 5 and taking its integer value (non decimal value).
Now looping through each character of bas64 encoded massage except the last two which are `==` and then adding 1 section of MAC that is 2 character every after one character of base64 encoded massage and storing is in encrypt variable.

This was the logic behind the encryption method and based on that we have to write a code to decrypt the massage that we are provided in the SECRET file

Content of SECRET file


```
debG29pbcP9aV8dn8cRebw29abcm9aV8dE8cVebU29dbc89aI8dU8cZebv29ZbcH9aN8dx8ceebm
29obcx9ab8dz8cZebg29Ybcn9aM8d08cYebH29Nbcm9aY8dm8c1ebt29ebcm9aB8dp8cYebn29Nb
cl9aY8dH8cUebx29YbcG9aR8dz8cNebW29Rbcs9aI8dX8c==
```


Here is the code created by me

```
import base64

new_msg =
"debG29pbcP9aV8dn8cRebw29abcm9aV8dE8cVebU29dbc89aI8dU8cZebv29ZbcH9aN8dx8ceeb
m29obcx9ab8dz8cZebg29Ybcn9aM8d08cYebH29Nbcm9aY8dm8c1ebt29ebcm9aB8dp8cYebn29N
```

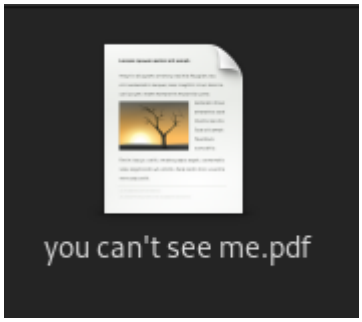
```
bcl9aY8dH8cUebx29YbcG9aR8dz8cNebW29Rbcs9aI8dX8c"
new_msg = new_msg[::3]

decrypt = ""

base64_message = new_msg + '='
base64_bytes = base64_message.encode('ascii')
message_bytes = base64.b64decode(base64_bytes)
message = message_bytes.decode('ascii')

for character in message:
    decrypt = decrypt + chr((ord(character)-1))

print(decrypt)
```

When we run the code we get this

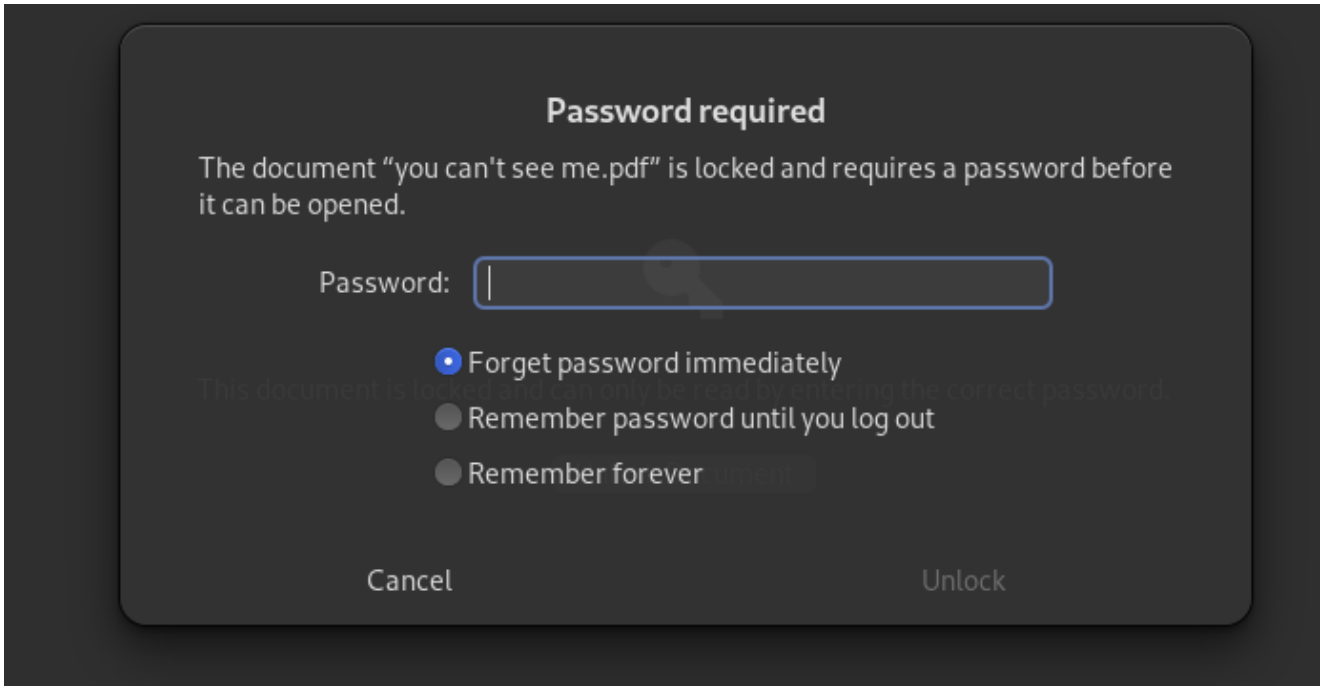
```
siNUsoidCTF{ Encrpyi0n5_ar3_really_hard_t0_cr4ck
[Finished in 38ms]
```

Just by adding a space and } we get out flag `siNUsoidCTF{ Encrpyi0n5_ar3_really_hard_t0_cr4ck }`

Q6

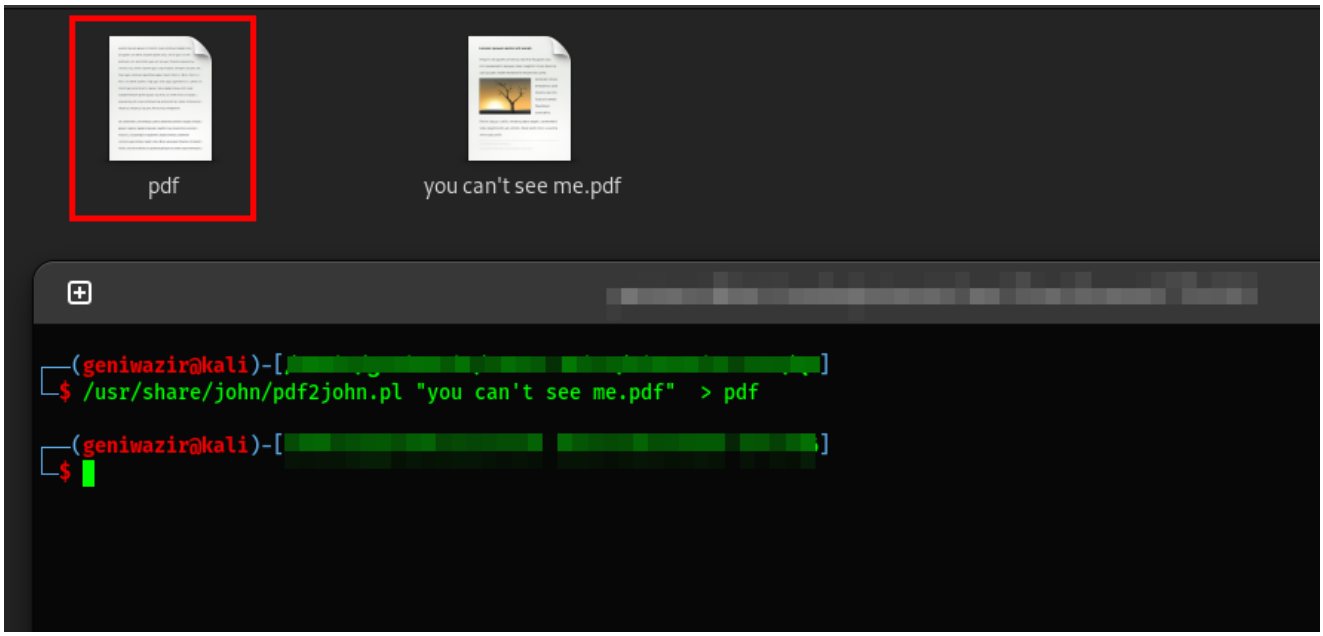


The hint `John sent me this file` clearly give us an indication that we will be requiring John for this file as it is password protected



So we will run John with the ROCKYOU.TXT wordlist to crack the password.

For that first we need the hash

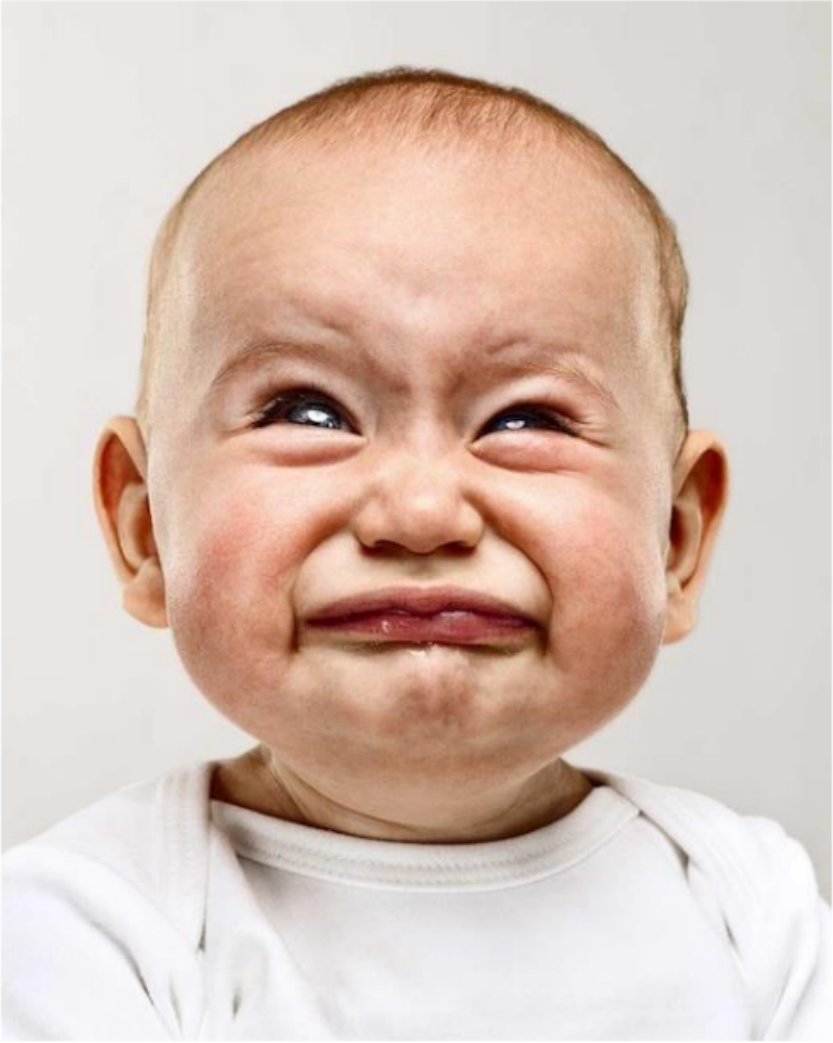


Now we have the hash into the PDF file so we will start the cracking process.

```
┌──(geniwazir㉿kali)-[/[redacted]]
└─$ john --format=pdf pdf /home/[redacted]/rockyou.txt
Created directory: /home/[redacted]/.john
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 6 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
elizabeth        (you can't see me.pdf)
1g 0:00:00:00 DONE 2/3 (2021-11-21 11:04) 2.325g/s 146093p/s 146093c/s 146093C/s crystal..bigben
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed
```

Now we have the password so, moving on to the pdf file we get this

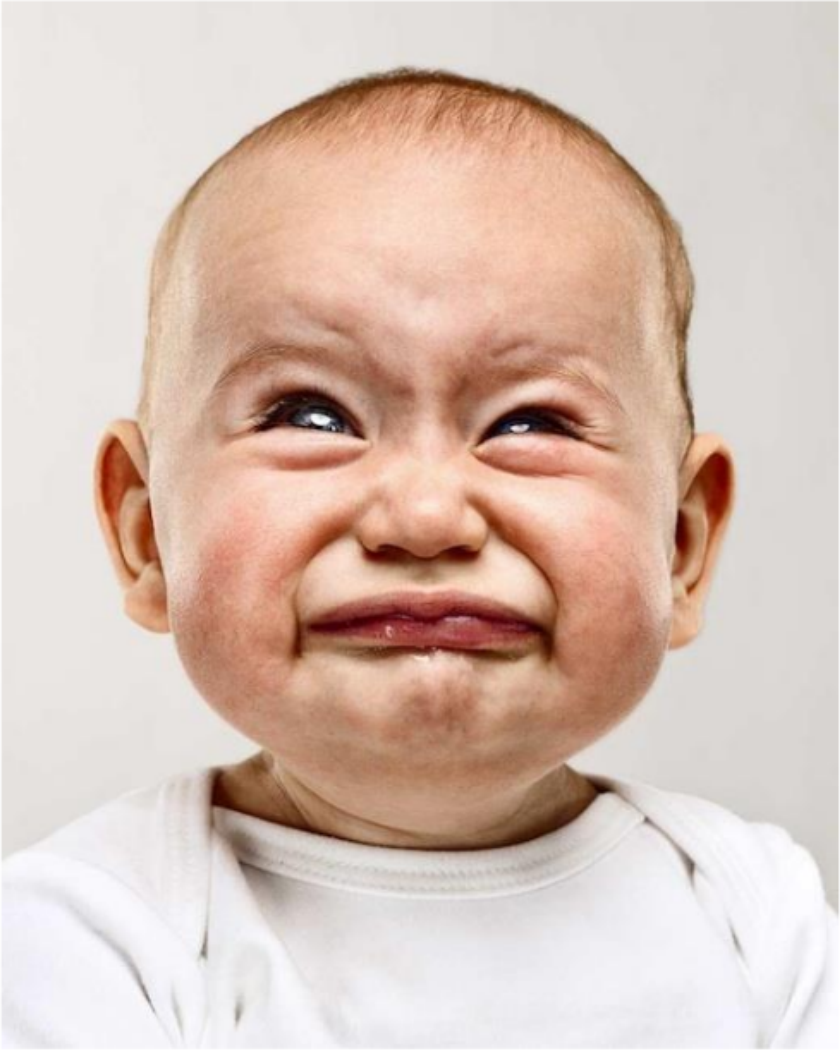
YOU WILL FIND NOTHING HERE
TRY HARDER AND HARDER



But if we try to select everything in the page the what we get is this

YOU WILL FIND NOTHING HERE

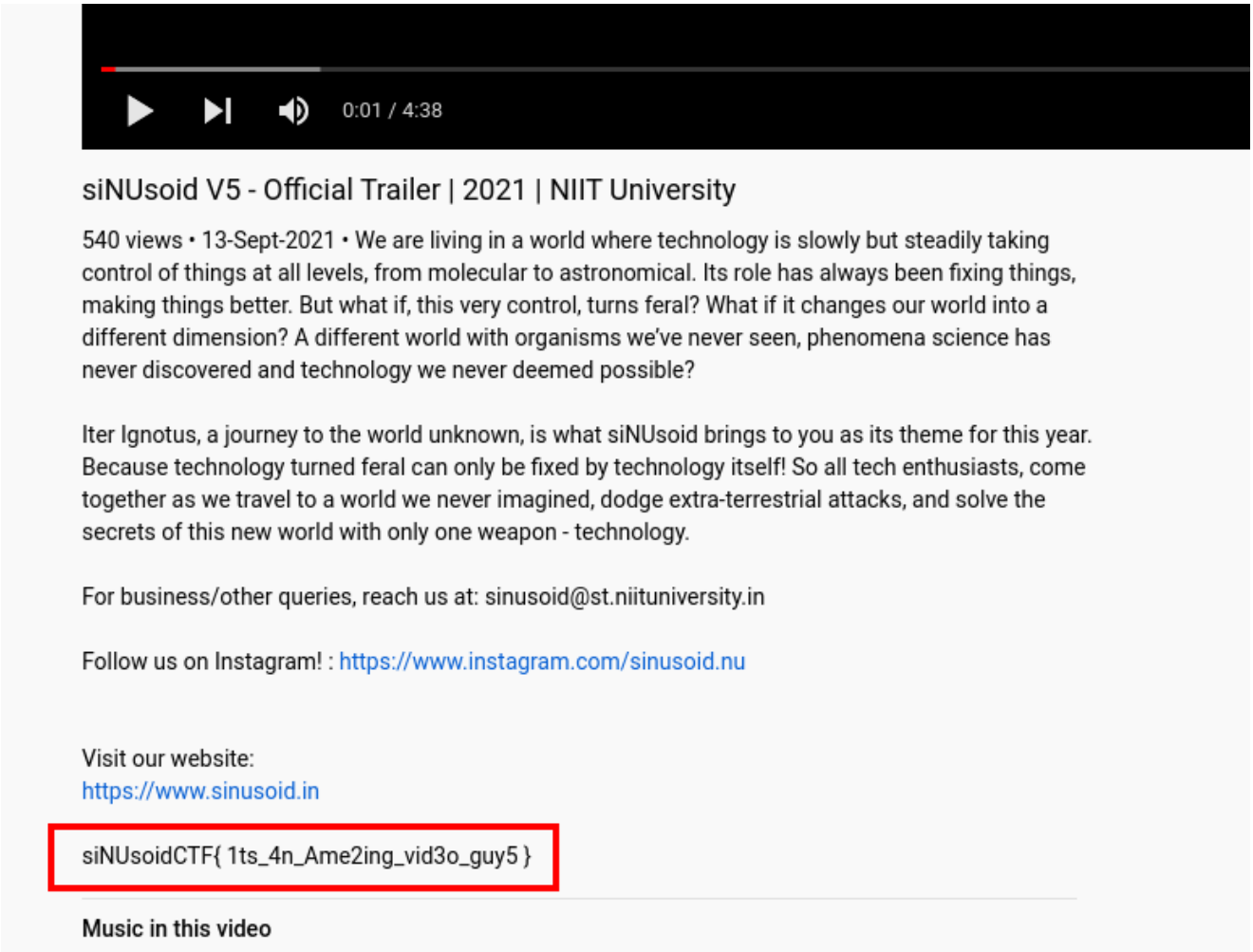
TRY HARDER AND HARDER



siNUsoidCTF{ I_Wa5_h1dd3N_heR3_y0u_5aw_M3 }

Q7

It was quite easy question just have to visit the video url and check the description of the video



Q8

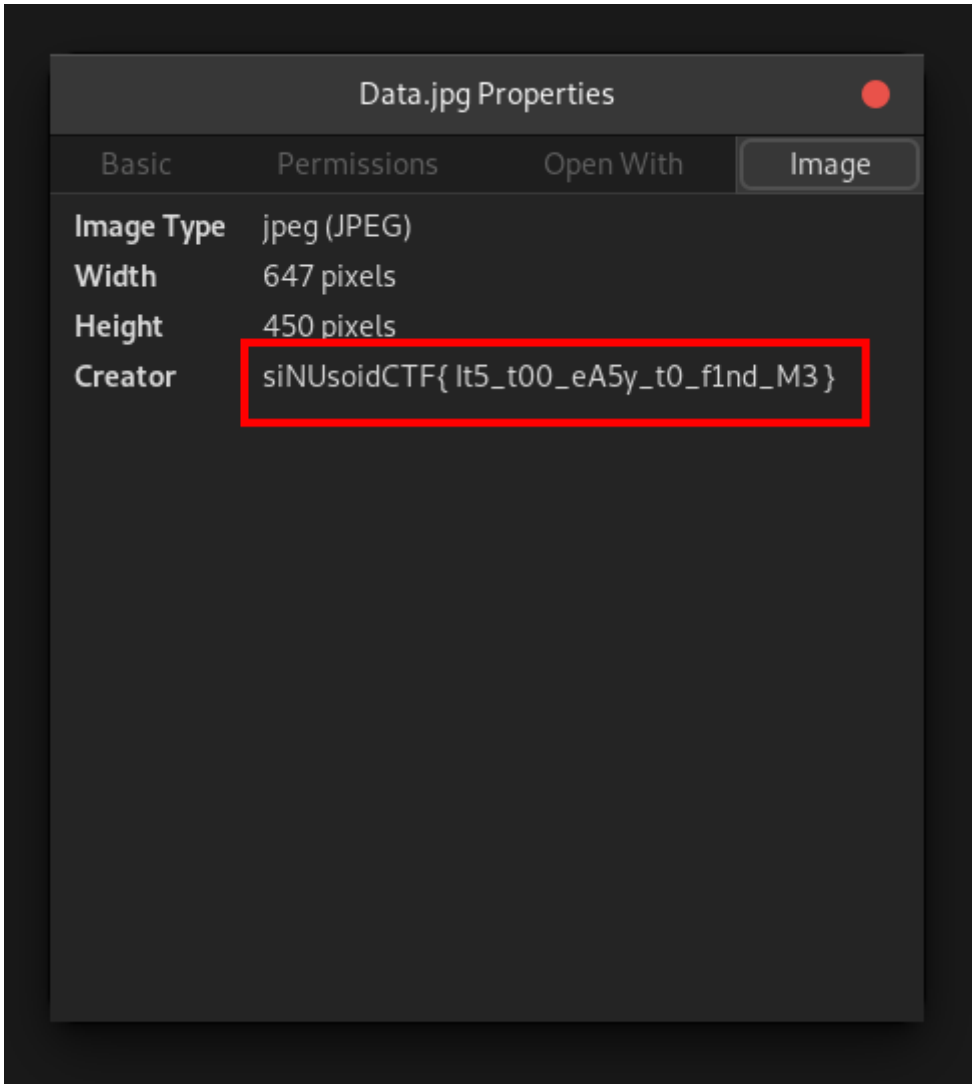
The Question description asks you to follow the rule. And the rules were written on the official website of the siNUsoid that is over this URL `https://sinusoid.in/ctf.html`

In the front end you will find nothing so visit the source-code and as the question says follow to rules so visit the rules div section and BOOOOOOM!!!!.

```
<div class="sinuv-gap-4"></div>
<h3 class="sinuv-decorated-h-2"><span><span class="text-main-1">Rules</span> & Regulations</span></h3>
<ol>
<li>Players have to compete individually.</li>
<li>Brute forcing of flags is not tolerated or recommended.</li>
<li>Automatic vulnerability scanners such as SQLmap are prohibited.</li>
<li>Flags should be submitted as they are found rather than hoarding flags and submitting them all at once.</li>
<li>If any participant is found sharing answers they will immediately be disqualified.</li>
<li>Students have a specific time to solve the questions of the CTF.</li>
<li>The student who reaches the final destination first wins.</li>
<li>Students are required to solve tricky questions which can be made using ciphers and encryptions as well.</li>
<li>The decision of the organizers in declaring the results will be final. No queries in this regard will be entertained.</li>
<li>The top 3 members have to submit POCs for all the questions they have solved.</li>
<li>POCs have to be submitted within 30 min after the event ends. If you fail to do so you will be directly disqualified.</li>
<li style="display: none;">siNUsoidCTF{ Flag_5h0uld_Be_1n_th1s_FoRm4t }</li>
</ol>
<div class="sinuv-gap-4"></div>
```

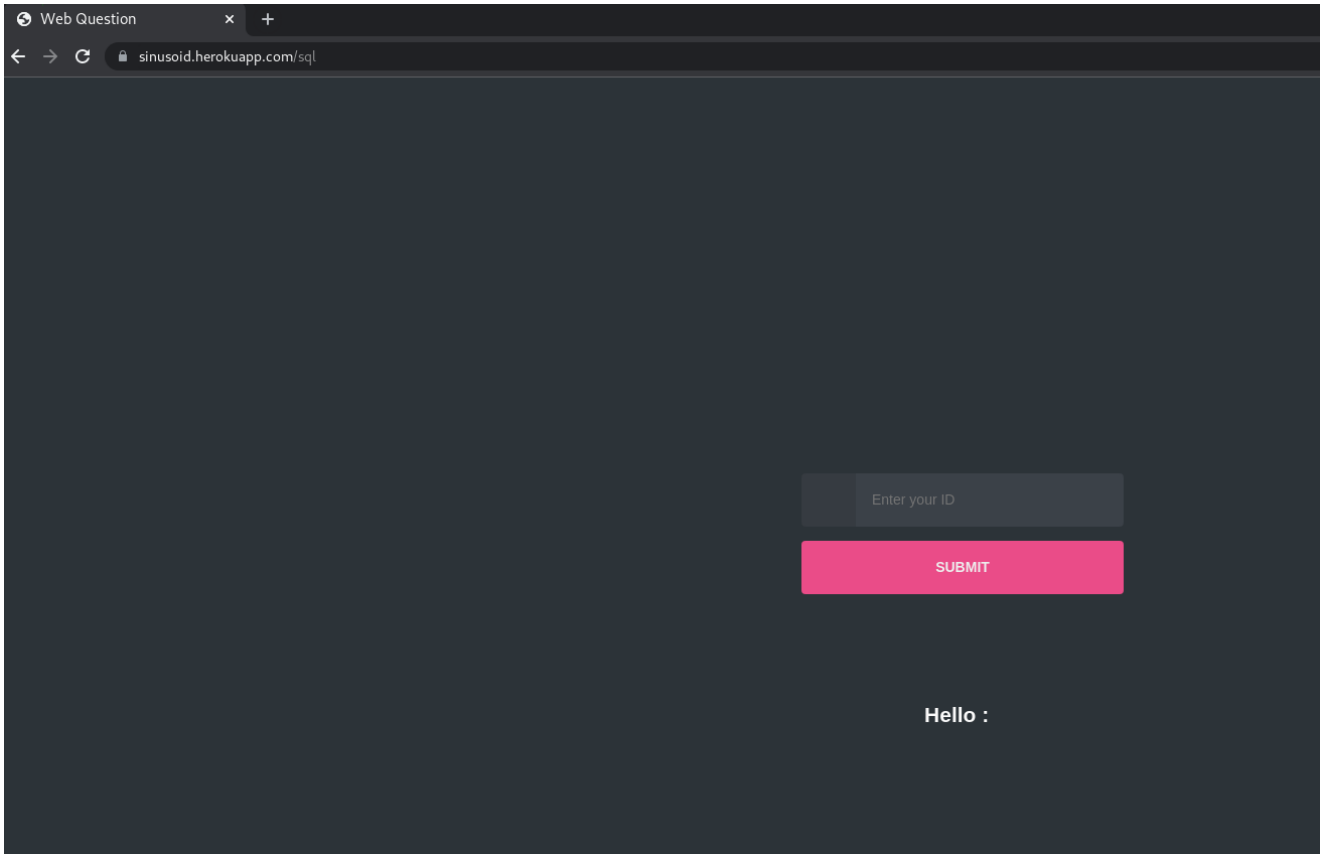
Q9

This one was also one of the very common question.
we just need to see the properties of this image

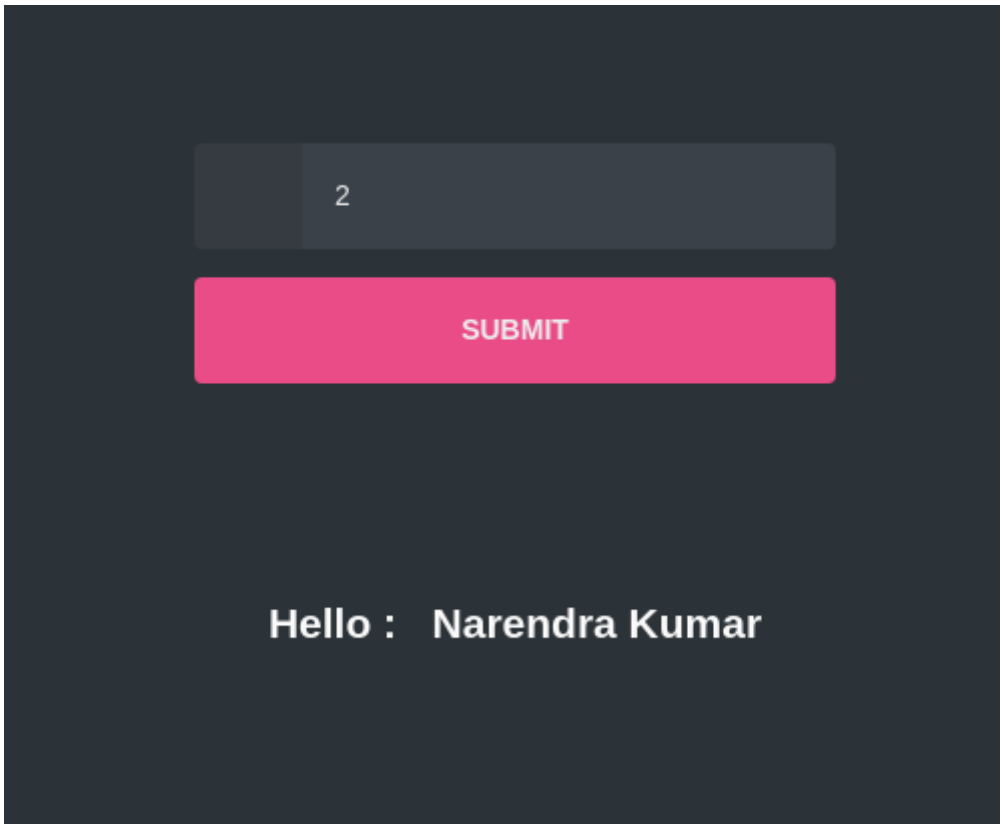


Q10

`https://sinusoid.herokuapp.com/sql`



This one was SQL Injection we can conclude that by looking at the URL



We can also confirm by just using this payload `1'--`

If this payload give you a 200 OK and returns the page that means it is accepting the SQL Injection.

The website is taking ID from the user and displaying the name for that particular ID.

Now if we look at the question description it says the

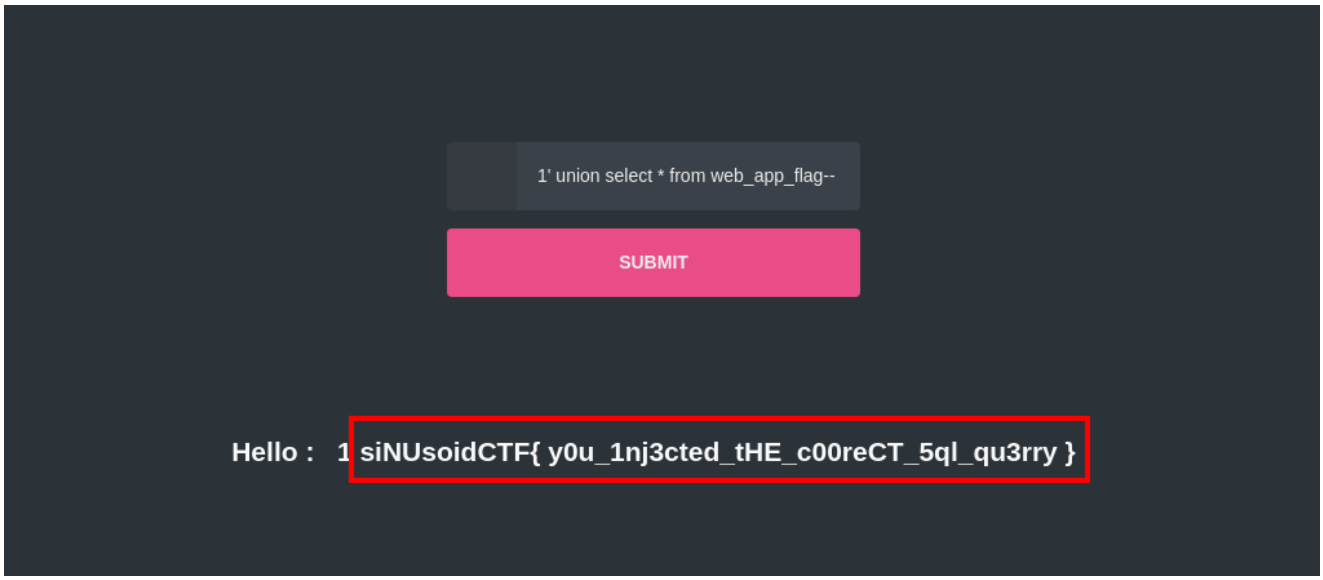
```
web_app_flag U https://sinusoid.herokuapp.com/sql
```

so from here it is clear that we need to inject the union payload.

The web_app_flag could be the database or the column name.

By basic enumeration we also came to know that there are two columns in the table so trying web_app_flag as a table name and executing this particular payload for the SQL Injection we can retrieve the flag

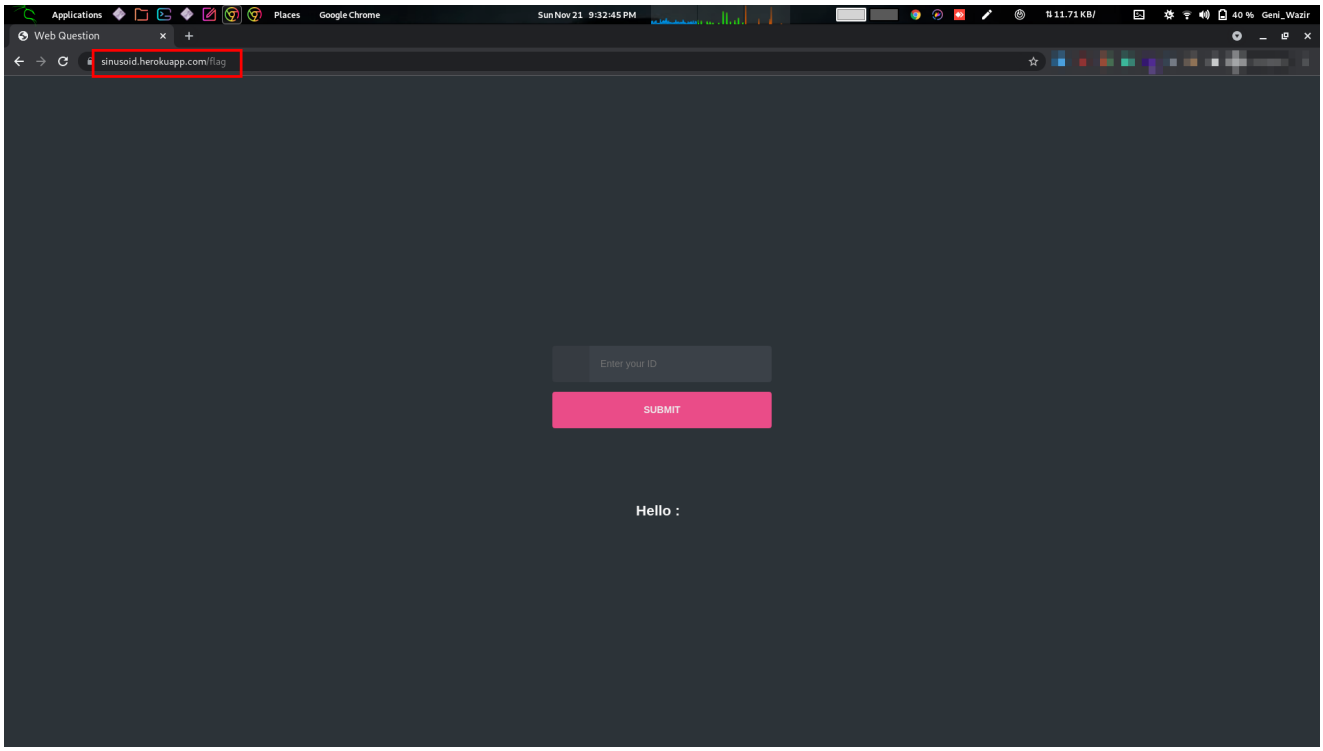
```
1' union select * from web_app_flag--
```



Q11

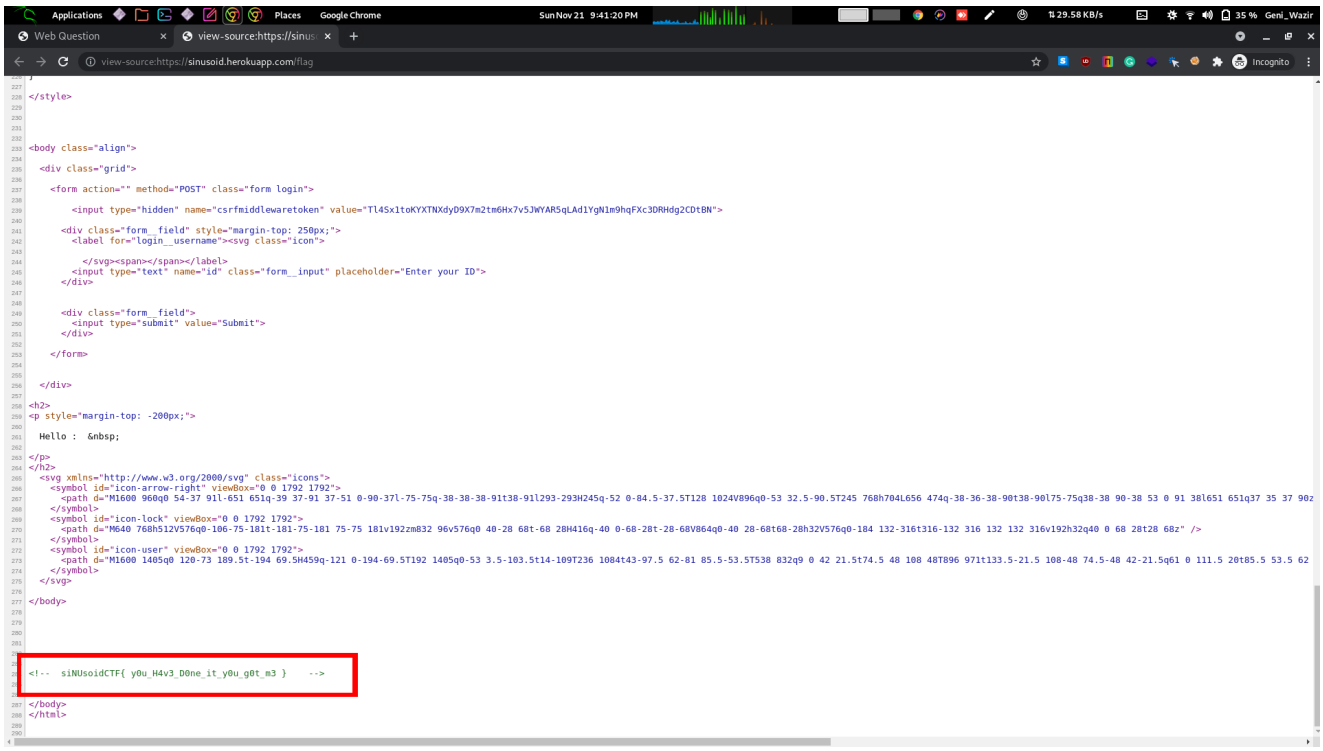
Doing a simple crawl in the BURP we will get that there is a page flag having URL `https://sinusoid.herokuapp.com/flag`

If we visit the page we get a similar web page one we got in SQL Injection.



But submitting anything in the input box do not do anything.

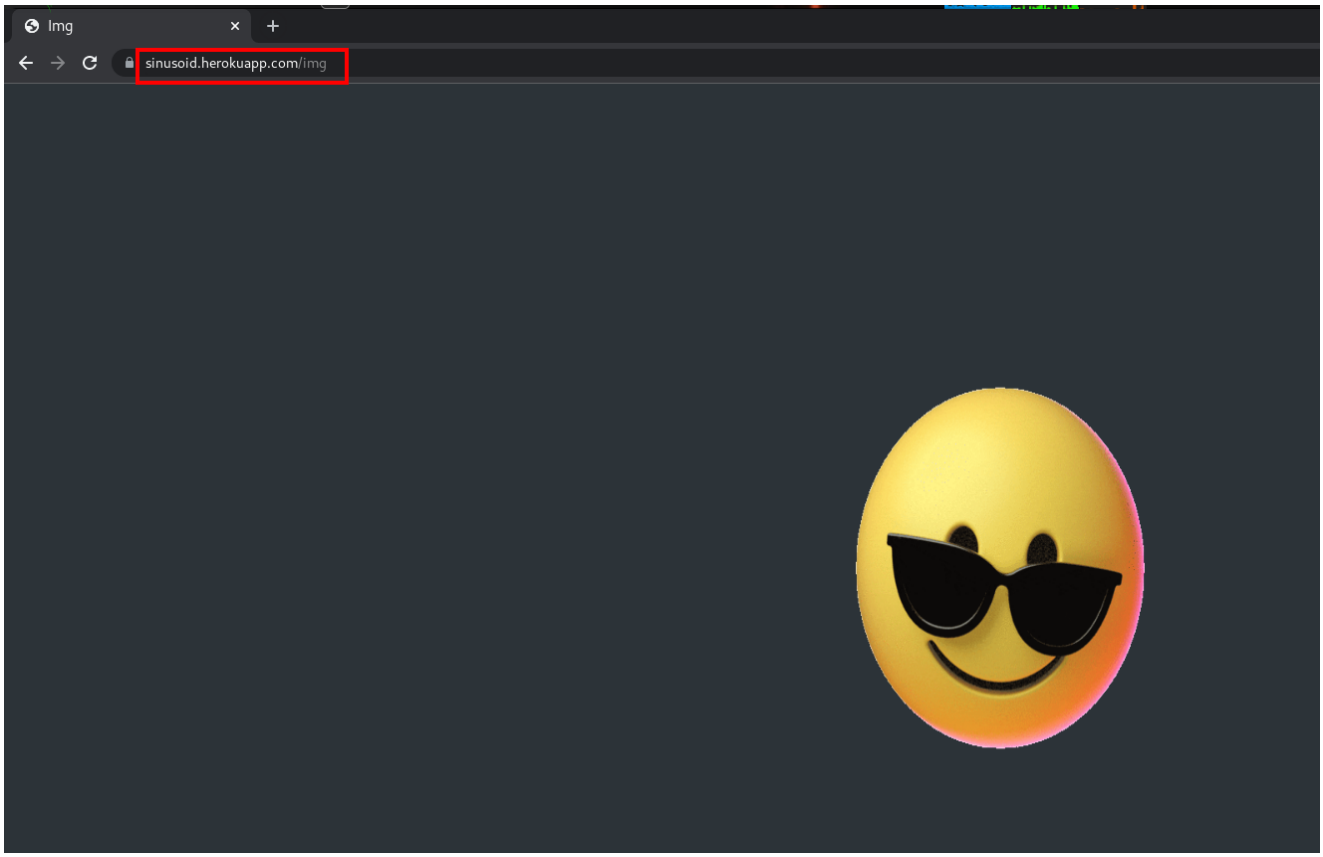
So the first thing that we should do is check the source-code and there you gooooooo!!!



Q12

In the last question we did the directory brute-force in that we will also find an img page.

Visiting to that page we get an emoji



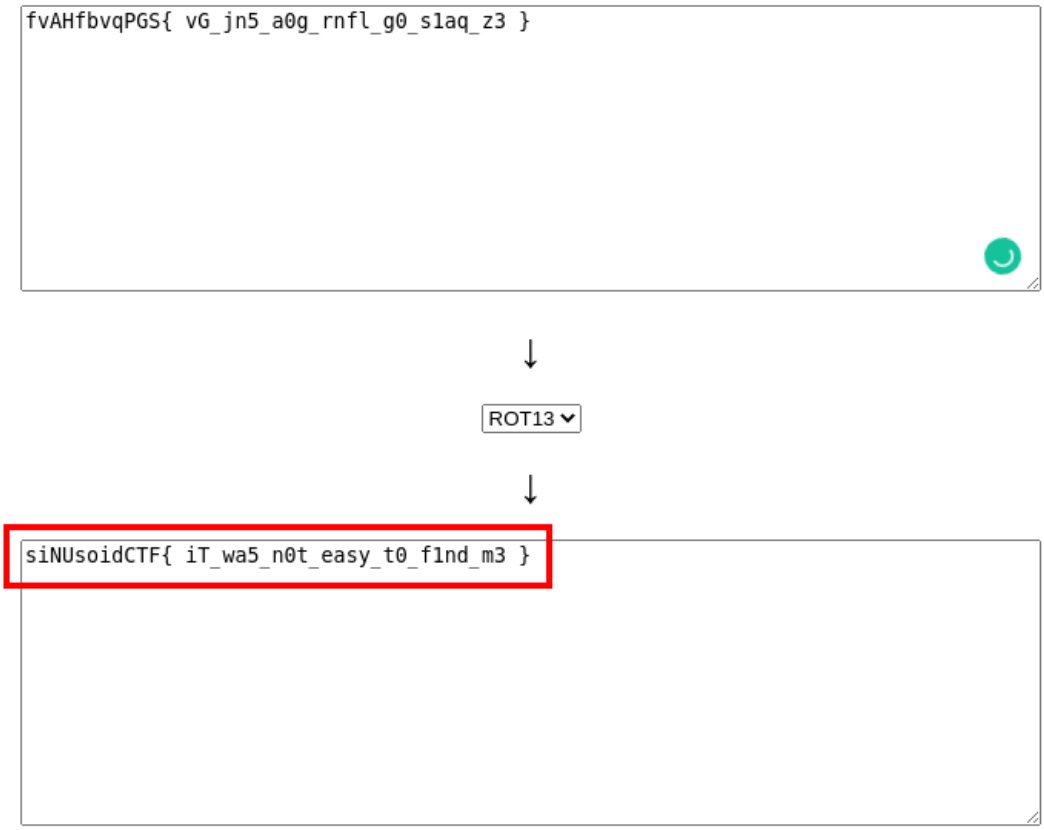
just hovering to the source-code of that page and wondering to see such a long name for a file with some url encoded characters.

```
<body class="align">
  <div class="grid">
    <img src="/static/ZnZBSGZidnFQR1N7IHZHX2puNV9hMGdfcm5mbF9nMF9zMWFxX3ozIH0%3D%3D.gif" height="500" width="400">
  </div>
```

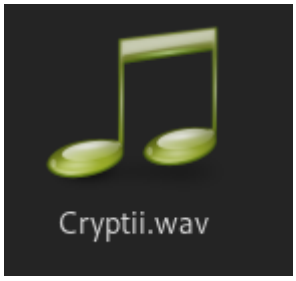
Decoding the URL encoded file name we got a base64 encoded massage and decoding it further we get a new string.

If we analyse that carefully we can see that each character is just incremented by 13 position that *a* has been replaced by*n* and so on.

This is a ROT13 encoding so decoding that we get the flag

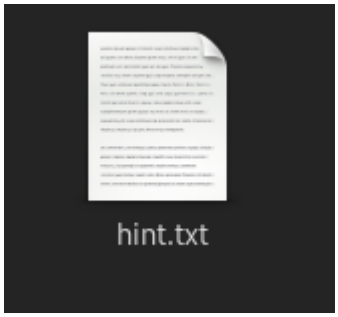


Q13



This is an audio file so there are only few possibilities that there could be a hidden file into it that is stenography. As in the hint DEEP SOUND was mentioned and it is quite famous tool also used in MR.ROBOT web series.

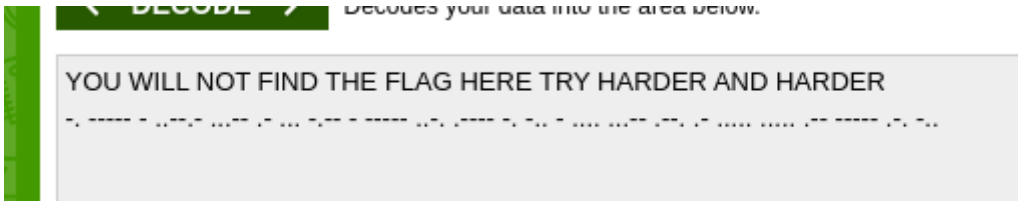
So trying the steghide with no password we get a hint file



The file contains this

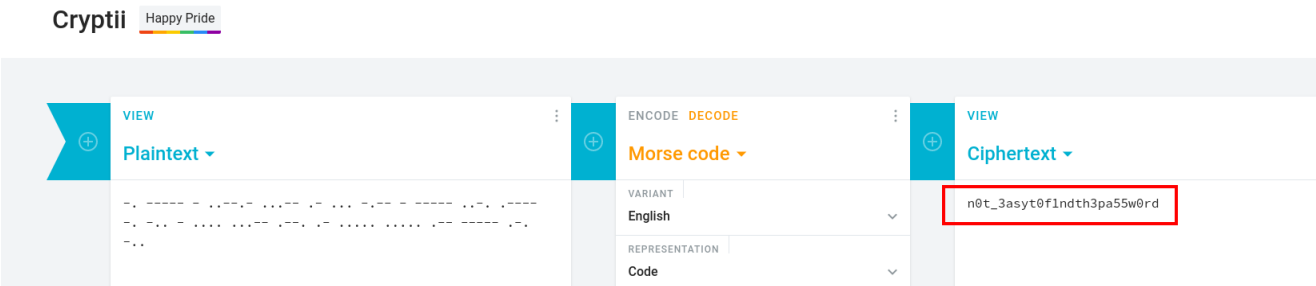
```
WU9VIFdJTEwgTk9UIEZJTkQgVEhFIEZMQUcgSEVSRSBUUlkgSEFSREVSIEFORCBIQVJERVIgCi0u
IC0tLS0tIC0gLi4tLS4tIC4uLi0tIC4tIC4uLiAtLi0tIC0gLS0tLS0gLi4tLiAuLS0tLSAtLiAt
Li4gLSAuLi4uIC4uLi0tIC4tLS4gLi0gLi4uLi4gLi4uLi4gLi0tIC0tLS0tIC4tLiAtLi4==
```

which is a base64 encoded massage decrypting that provide us this

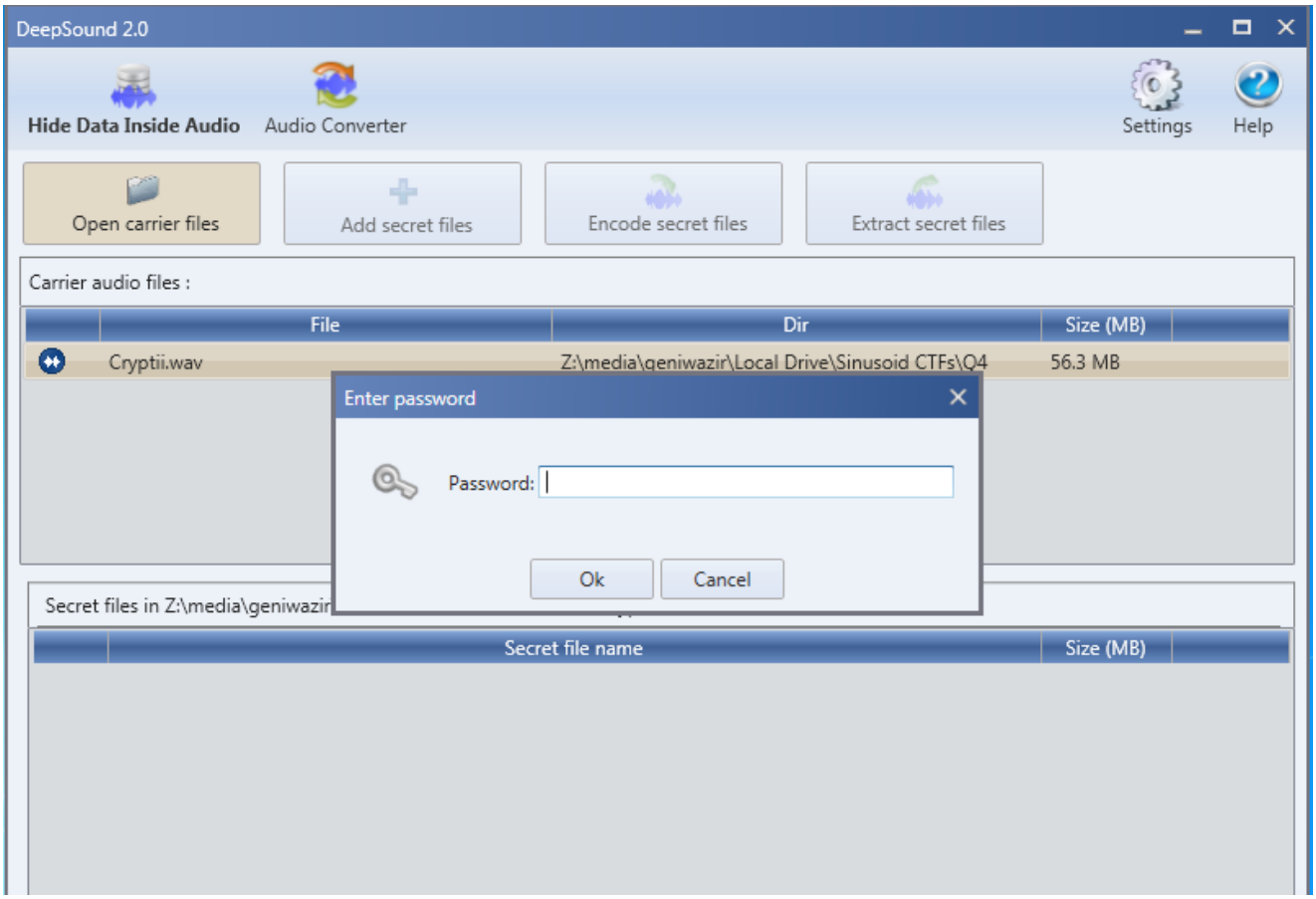


We can see that there is something below the massage `YOU WILL NOT FIND THE FLAG HERE TRY HARDER AND HARDER`
which is `-. ——— - .. --•– ... -- •– ... -.-- - ——— .. -. .——— -. -.. - ....`
`... -- .--. •– ..... ..... .-- ——— •–. -..`

And that is clearly a morse code and name of the audio file depicts where we need to crack the code that is `[Cryptii](https://cryptii.com/)` as some sites converts the massage into uppercase.

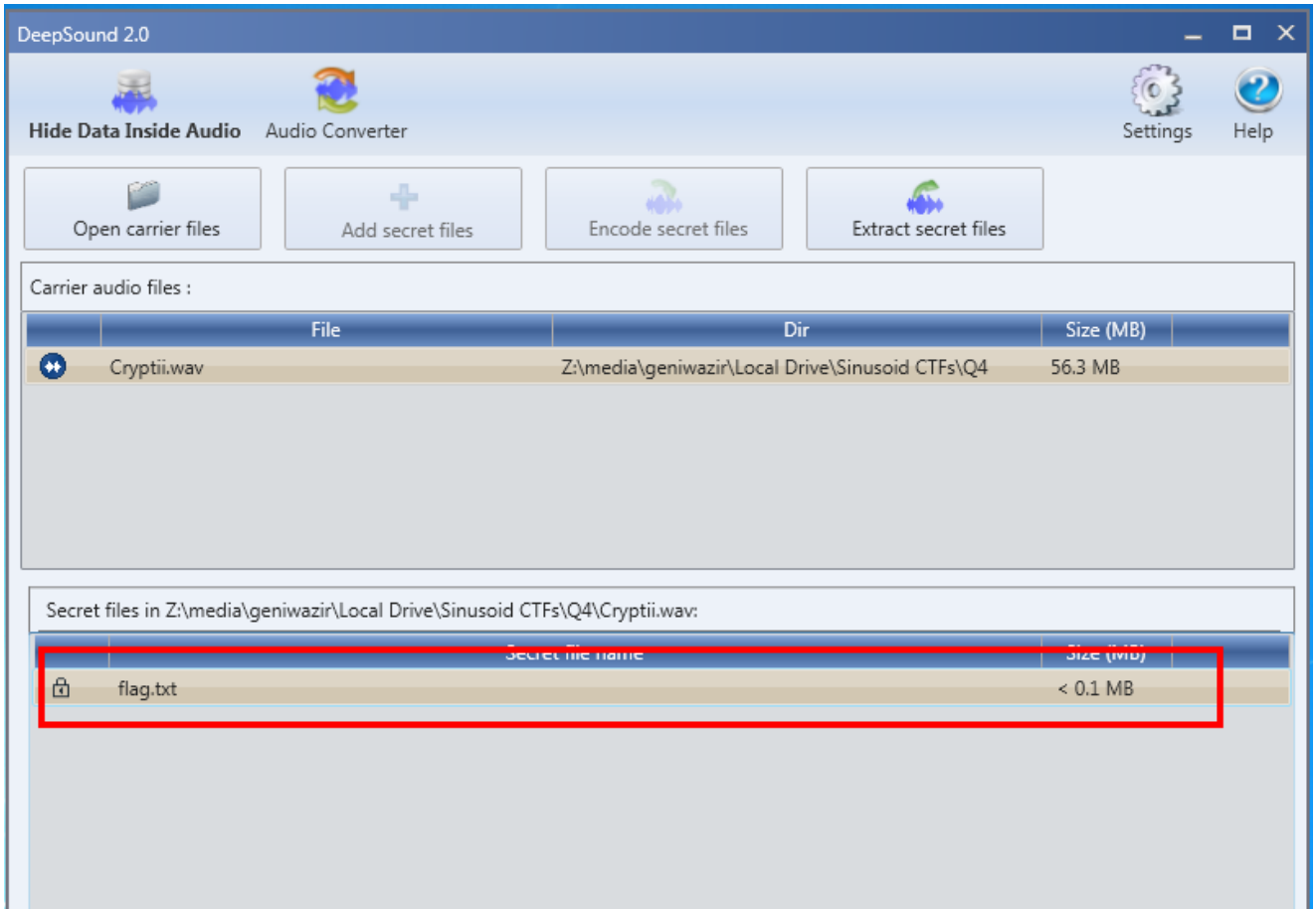


So basically this is the password that will be required in next step.
As DEEP SOUND was mentioned in the hint we trying to use that tool and importing the audio file.



It is asking a password so we will try to use the text that we obtained in the last step after decrypting the morse code that is `n0t_3asyt0f1ndth3pa55w0rd`

Here we can see that there is a text file with name as flag extracting that file.




```
siNUsoidCTF{ wh4t_An_Id3a_f0r_st3gn0gr4phy }
```

Q14

The hint for this file is very simple and famous for buffer overflow
The file name `2.0` depicts the buffer memory size that is 20.

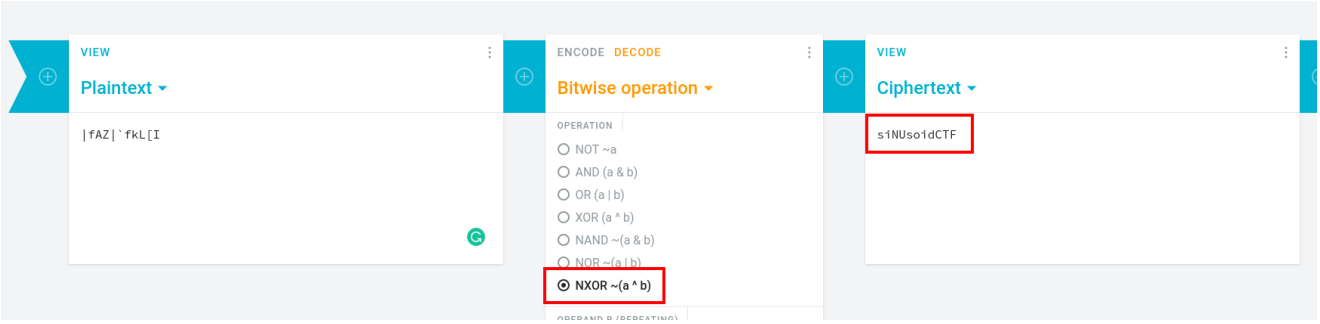
```
┌──(geniwazir㉿kali)-[ [redacted] ]
└─$ ./2.0

 Enter the password :
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

 Wrong Password

A B == A` B`
|fAZ|`fkL[I

A = 65
HQYXtAnEi>?N

==/2
NUZW2MDSPFPQ====

255
67 114 52 99 107 51 100 95 109 101 95

If you know, you know
U+21 U+21 U+20 U+7D
zsh: segmentation fault  ./2.0
```

Here are 5 statements.

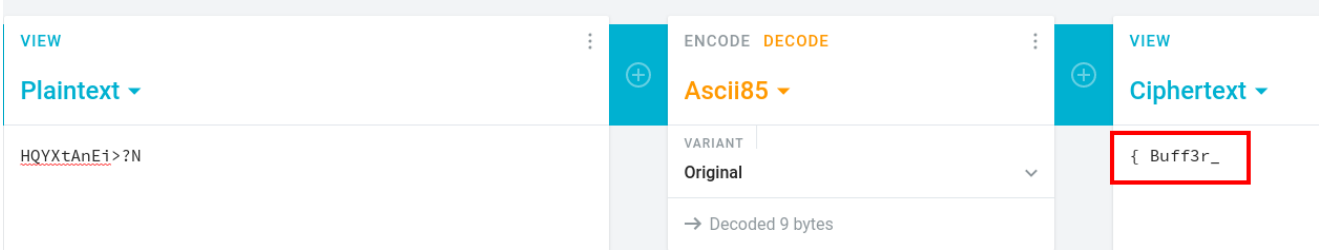
The first one is

```
A B == A` B`
```

from this we can say that the massage bellow it is xor and nxor encrypted so trying that. And we found that its a nxor

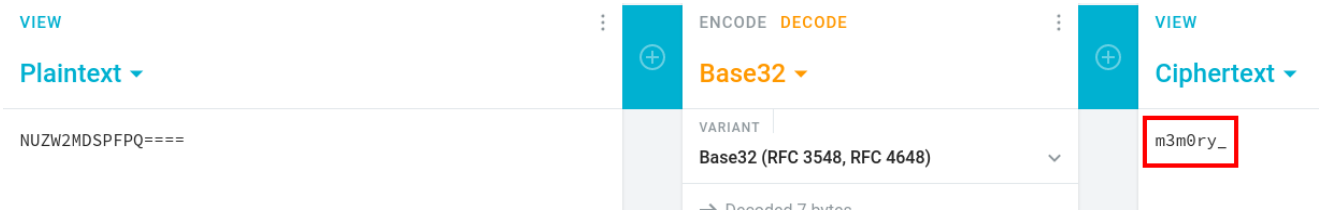


The second section is `A = 65` which is clearly point us to the ASCII so

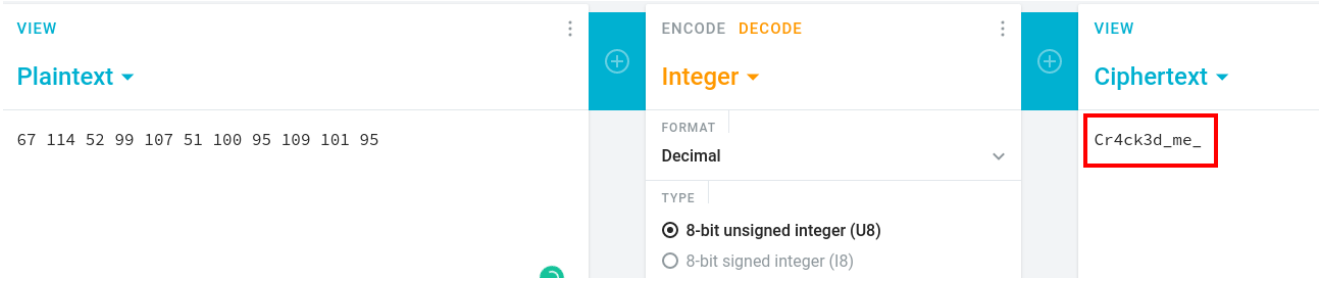


The third one is `==/2` and from the encoded massage itself we can point out that its a base32 encoded because of 4 `=`

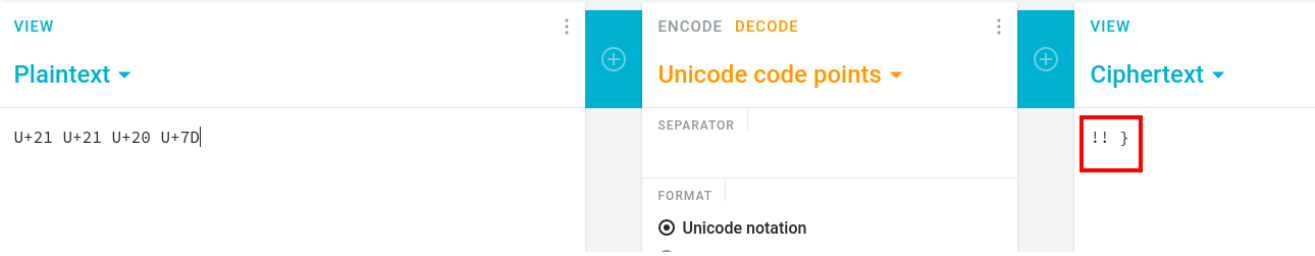
From the hint == is for basse64 and dividing it by 2 gives us 32 so base32.



The second last section is `255` which is equivalent to 8 bit binary number that is `1 1 1 1 1 1 1 1`



And the last section can be Unicode encoded massage as the encoded massage format shows `U+21 U+21 U+20 U+7D`



Combining all the 5 parts that we decode just now makes the flag.

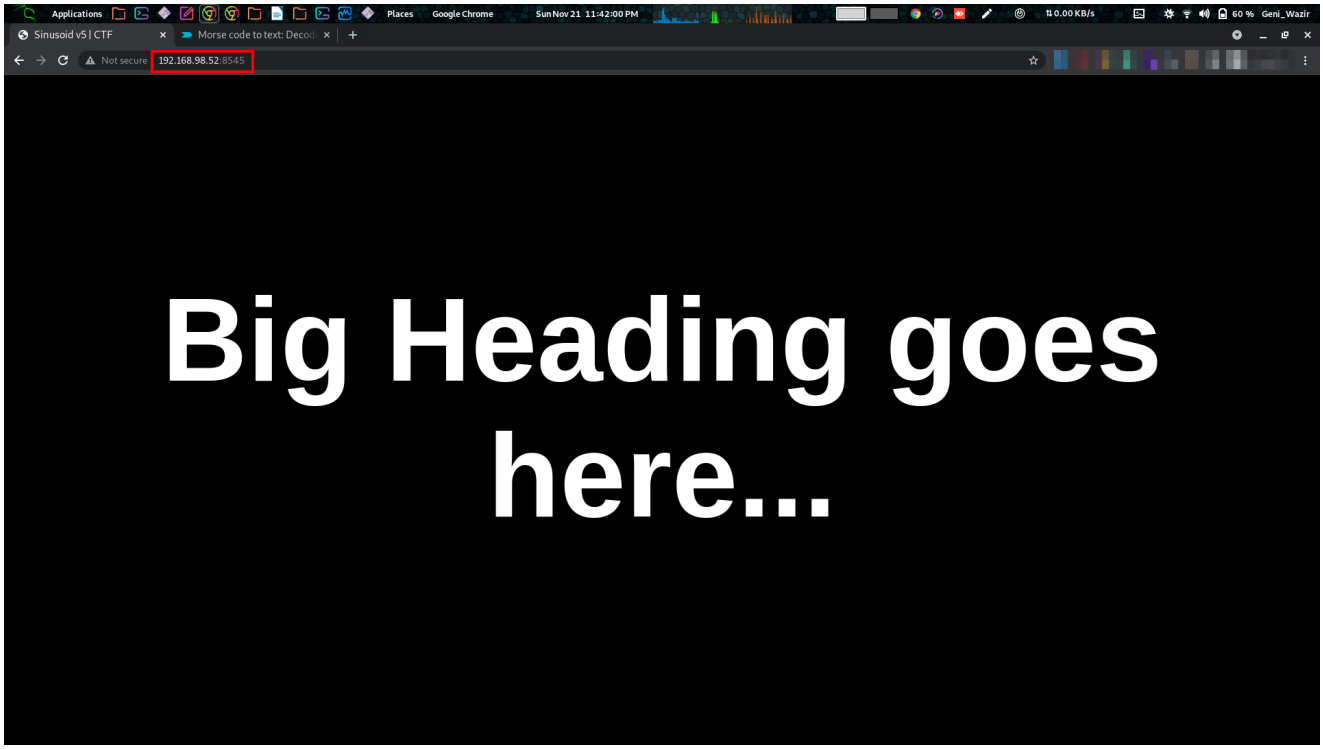
Q15

For this ubuntu server the username and the password was provided to us.

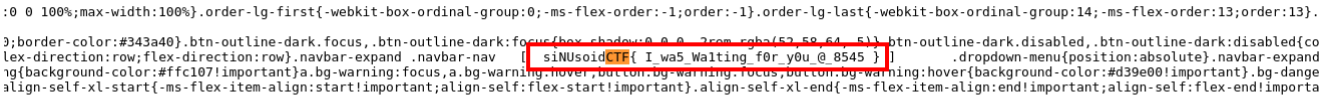
So just we need to import it into our virtual box and set the network to bridge and scan the ubuntu server using `nmap` with a vierson and over all ports as its a network question.

```
┌──(geniwazir㉿kali)-[~]
└─$ nmap -T4 -sV 192.168.98.52 -p-
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-21 23:39 IST
Nmap scan report for 192.168.98.52
Host is up (0.00032s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8545/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.15 seconds

┌──(geniwazir㉿kali)-[~]
└─$
```

We can see that Apache server is up but not on a usual port insted it is running over port 8545 visiting to that port from our browser we get this



Checking its source code and searching for the flag and BOOOOOM!!!!!



Q16

If we look at the home directory and list there look what we get

```
sinusoidctf@sinusoidv5:/home$ ls -la
total 16
drwxr-xr-x  4 root        root        4096 Aug 13 07:20 .
drwxr-xr-x 20 root        root        4096 Aug 13 06:47 ..
drwxr-xr-x  4 sinusoidctf sinusoidctf 4096 Nov  6 05:26 sinusoidctf
drwxr-xr-x  3        1000        1000 4096 Nov  6 03:09 test
sinusoidctf@sinusoidv5:/home$
```

A test directory is there and if list the test directory we get a flag file

```
sinusoidctf@sinusoidv5:/home$ cd test/
sinusoidctf@sinusoidv5:/home/test$ ls
flag  t  w
sinusoidctf@sinusoidv5:/home/test$ cat flag
siNUsoidCTF{ it_wa5_v3ry_E4sy_t0_FinD_m3!!!! }
sinusoidctf@sinusoidv5:/home/test$
```

Q17

In the hint of this question there is a riddle if we solve that we get the answer as shadow.
So, there is a shadow file in the /etc directory which stores the hashed passwords of the users. It is same as SAM file for windows.

Printing the file content we get a small encoded massage at the end of the file

```
systemd-coredump:!!:18852::::::
lxd:!:18852::::::
sinusoidctf:$6$5lDewpp/M0HXWWt6$EMyZI1Atd
q/j1s0pMxL8phTrWe/:18852:0:99999:7:::

#c2l0VXNvaWRDVEZ7IElfVzRzX2gxZ==
sinusoidctf@sinusoidv5:/home/test$ _
```

The shadow file contains the hashed password and also the passwd file in the /etc directory contains the hashed password.

So, if we cat that file there also we get a encoded massage

```
landscape:x:109:115::/var/lib/landscape:
pollinate:x:110:1::/var/cache/pollinate:
usbmux:x:111:46:usbmux daemon,,,:/var/li
sshd:x:112:65534::/run/sshd:/usr/sbin/no
systemd-coredump:x:999:999:systemd Core
lxd:x:998:100::/var/snap/lxd/common/lxd:
sinusoidctf:x:1001:1001:,,,:/home/sinuso

#GQzbl93aVRoX3RoM19wQTU1dzByZcB9==
sinusoidctf@sinusoidv5:/home/test$
```

The `==` tells us that they are base64 encoded. But they are not individually encoded into base64.
They the the 2 parts of the encoded massage first part was in shadow file and the second in the passwd file combining both of them and then decoding them gives us the required flag.

```
sinusoidctf@sinusoidv5:~$ echo "c2lOVXNvaWRDVEZ7IElfVzRzX2gxZGQzbl93aVRoX3RoM19wQTU1dzByZcB9==" | base64 -d
siNUsoidCTF{ I_W4s_h1dd3n_wiTh_th3_pA55w0re}base64: invalid input
sinusoidctf@sinusoidv5:~$
```

END