NAME: NURDAYANA BINTI MOHD AIDI

MATRIC NO:B031610128

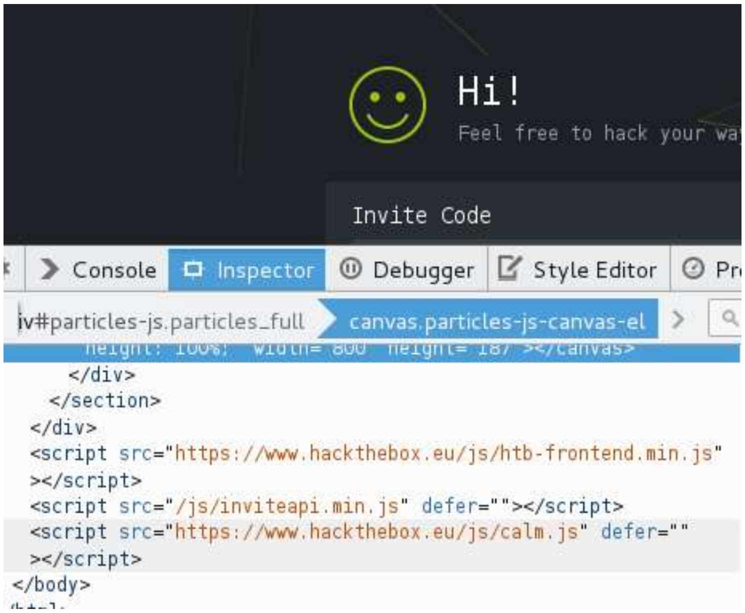
CLASS: 3BITZ

LECTURER: MOHD ZAKI BIN MAS’UD

**Hack the Box Challenges Report**

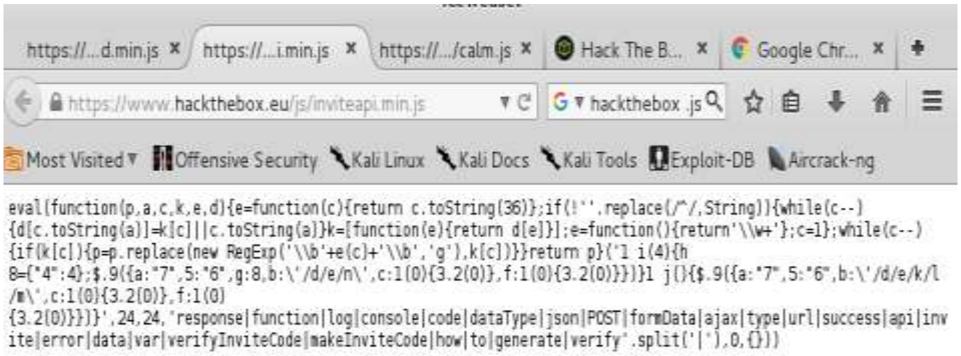
1.0 Invite code

1.In the invite code page press Fn+F12. Notice the link” js/inviteapi.min.js” in the script src code. It does not sound typical and might contain clues.

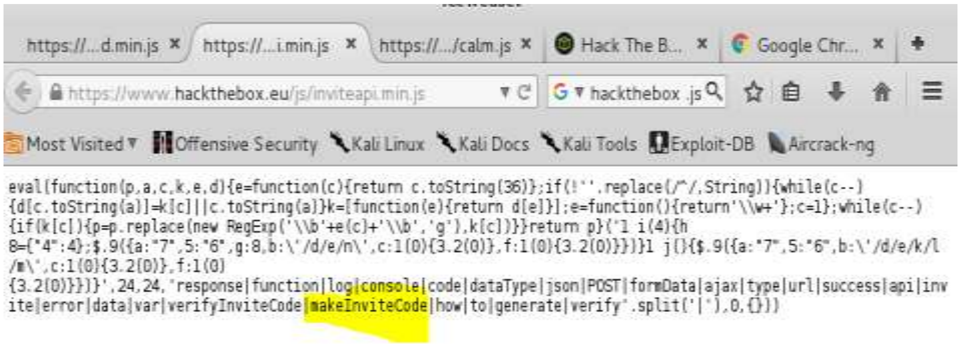

Hi!
Feel free to hack your way
Invite Code
Console
Inspector
Debugger
Style Editor
Pr
iv#particles-js.particles_full
canvas.particles-js-canvas-el

```
      height: 100%;" width="800" height="187"></canvas>
    </div>
  </section>
</div>
<script src="https://www.hackthebox.eu/js/htb-frontend.min.js"
></script>
<script src="/js/inviteapi.min.js" defer=""></script>
<script src="https://www.hackthebox.eu/js/calm.js" defer=""
></script>
</body>
```

2.Open the link in the browser. Notice the MakeInviteCode function.

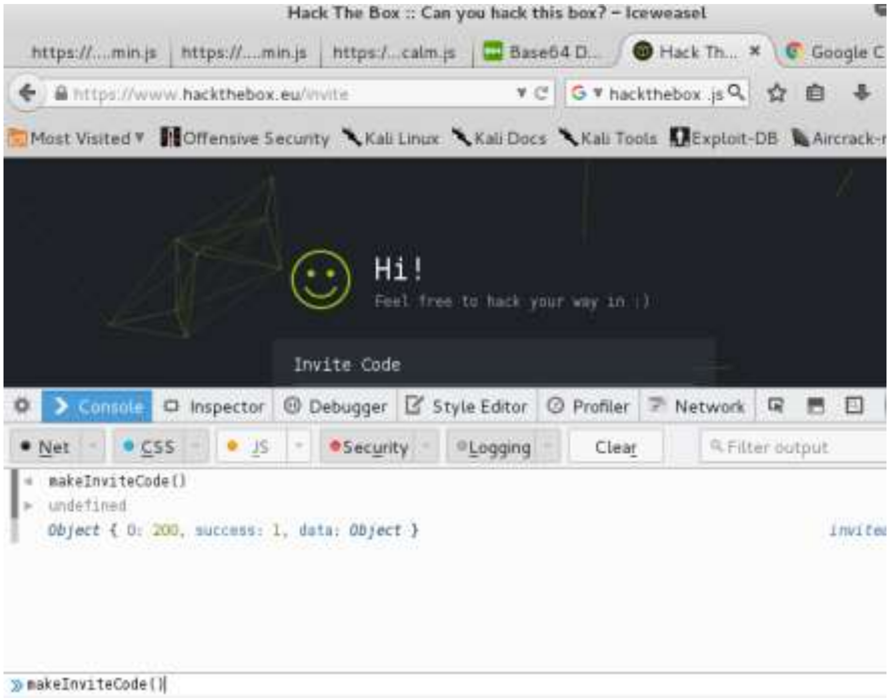


```
eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};if(!''.replace(/^/,String)){while(c--)
{d[c.toString(a)]=k[c]||c.toString(a)}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1};while(c--)
{if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('1 i(4){h
8={"4":4};$.9({a:"7",5:"6",g:8,b:\'/d/e/n\',c:1(0){3.2(0)},f:1(0){3.2(0)}})}1 j(){$.9({a:"7",5:"6",b:\'/d/e/k/l
/m\',c:1(0){3.2(0)},f:1(0)
{3.2(0)}})}',24,24,'response|function|log|console|code|dataType|json|POST|formData|ajax|type|url|success|api|inv
ite|error|data|var|verifyInviteCode|makeInviteCode|how|to|generate|verify'.split('|'),0,{}))
```



```
eval(function(p,a,c,k,e,d){e=function(c){return c.toString(36)};if(!''.replace(/^/,String)){while(c--)
{d[c.toString(a)]=k[c]||c.toString(a)}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1};while(c--)
{if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('1 i(4){h
8={"4":4};$.9({a:"7",5:"6",g:8,b:\'/d/e/n\',c:1(0){3.2(0)},f:1(0){3.2(0)}})}1 j(){$.9({a:"7",5:"6",b:\'/d/e/k/l
/m\',c:1(0){3.2(0)},f:1(0)
{3.2(0)}})}',24,24,'response|function|log|console|code|dataType|json|POST|formData|ajax|type|url|success|api|inv
ite|error|data|var|verifyInviteCode|makeInviteCode|how|to|generate|verify'.split('|'),0,{}))
```

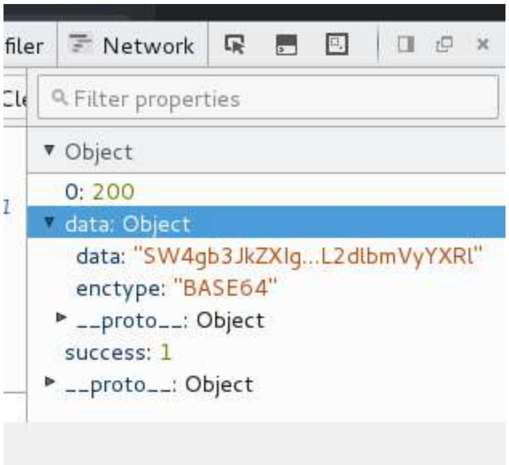
3.So we open console in browser by typing Fn+F12. Enter the makeinviteCode() in the console.



```
makeInviteCode()
undefined
Object { 0: 200, success: 1, data: Object }
```

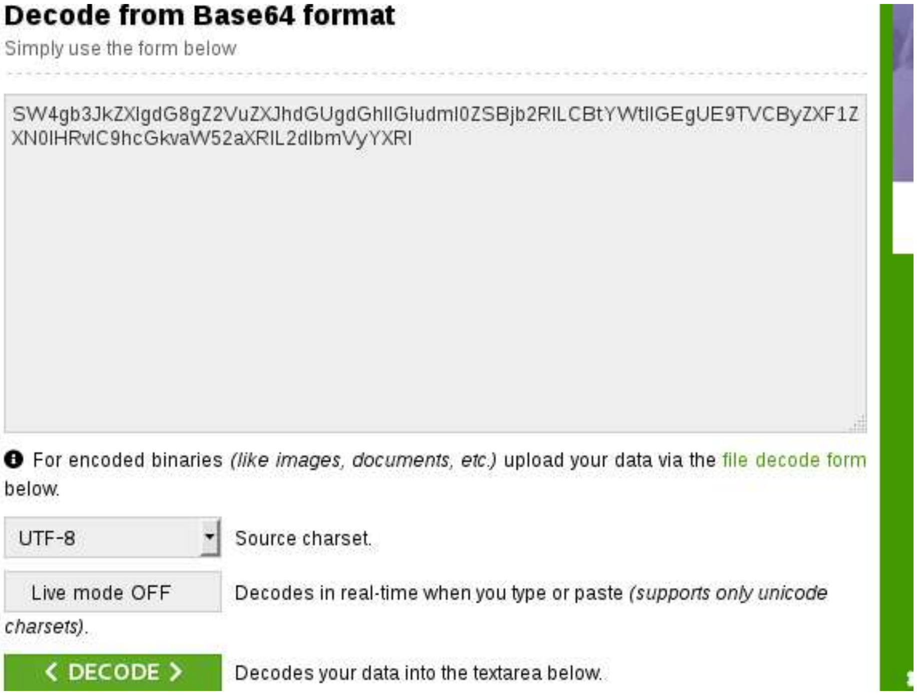
```
» makeInviteCode()
```

4.In the data segment you will find a base64 encoded strong which might contain clue.



```
▼ Object
    0: 200
  ▼ data: Object
      data: "SW4gb3JkZXIg...L2dlbmVyYXRl"
      enctype: "BASE64"
    ▶ __proto__: Object
    success: 1
  ▶ __proto__: Object
```

5.Decode it using online base decoder.


Decode from Base64 format

Simply use the form below

```
SW4gb3JkZXIgdG8gZ2VuZXJhdGUgdGhlIGludml0ZSBjb2RlLCBtYWtlIGEgUE9TVCByZXF1Z
XN0IHRvIC9hcGkvaW52aXRlL2dlbmVyYXRl
```

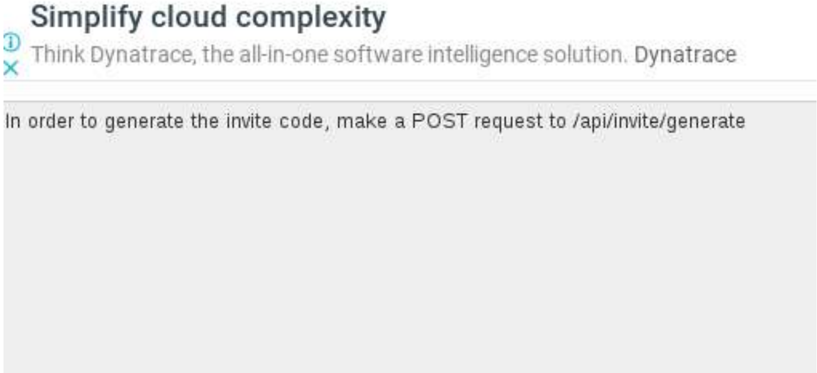
For encoded binaries *(like images, documents, etc.)* upload your data via the file decode form below.

UTF-8 Source charset.

Live mode OFF Decodes in real-time when you type or paste *(supports only unicode charsets)*.

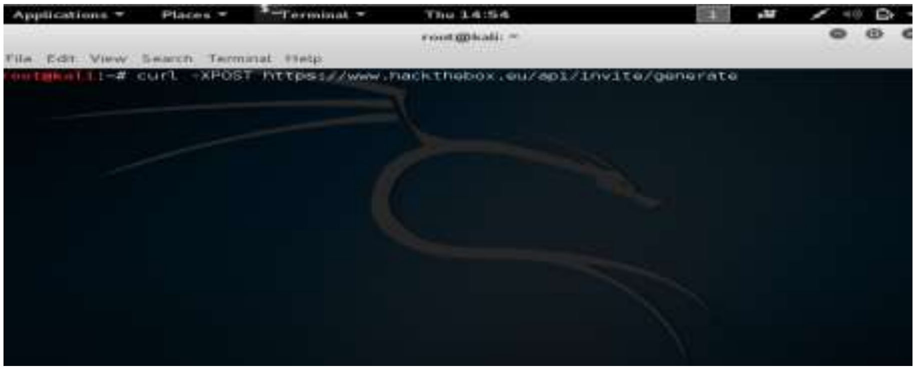
< DECODE > Decodes your data into the textarea below.

6.It tells us to make POST request to a url.



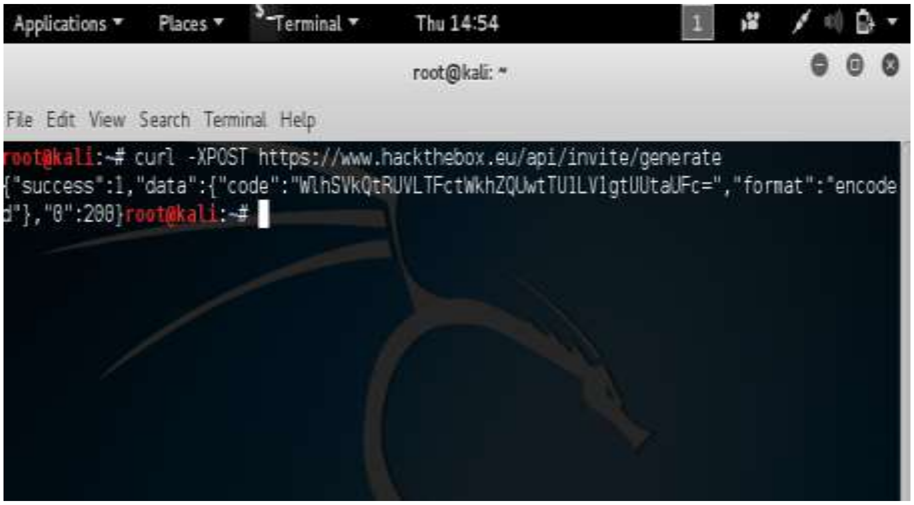
In order to generate the invite code, make a POST request to /api/invite/generate

7.Open Kali Linux terminal and use curl command to make POST request as below.



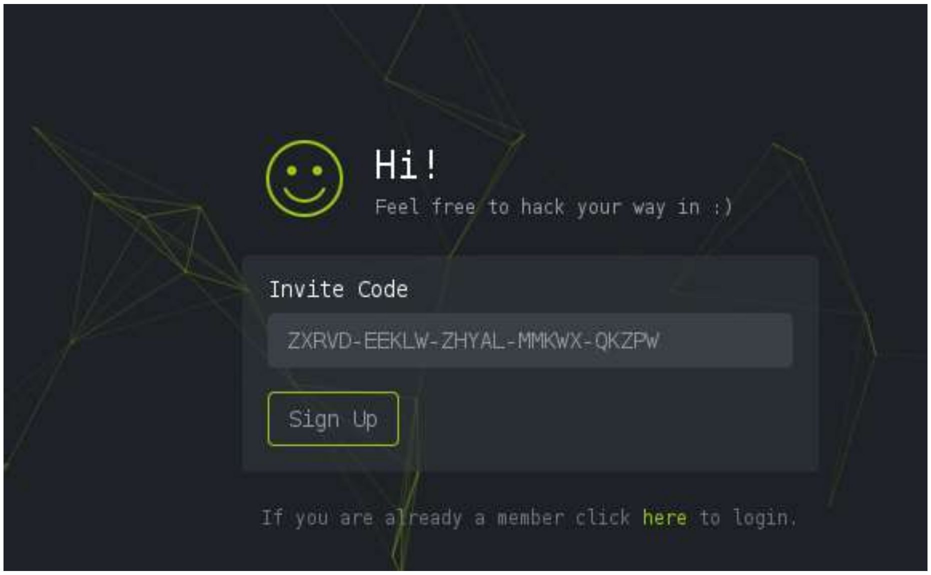
```
root@kali:~# curl -XPOST https://www.hackthebox.eu/api/invite/generate
```

8.It seems that the Invite Code is now generated. We must decode this first before proceeding.



```
root@kali:~# curl -XPOST https://www.hackthebox.eu/api/invite/generate
{"success":1,"data":{"code":"WlhSVkQtRUVLTFctWkhZQUwtTU1LV1gtUUtaUFc=","format":"encode
d"},"0":200}root@kali:~# 
```

9.After decoding it as in step 5 using online base decoder, we enter the Invite code in the page.



Hi!

Feel free to hack your way in :)

Invite Code

ZXRVD-EEKLW-ZHYAL-MMKWX-QKZPW

If you are already a member click here to login.

10. Now we can register and sign in to Hack the box.

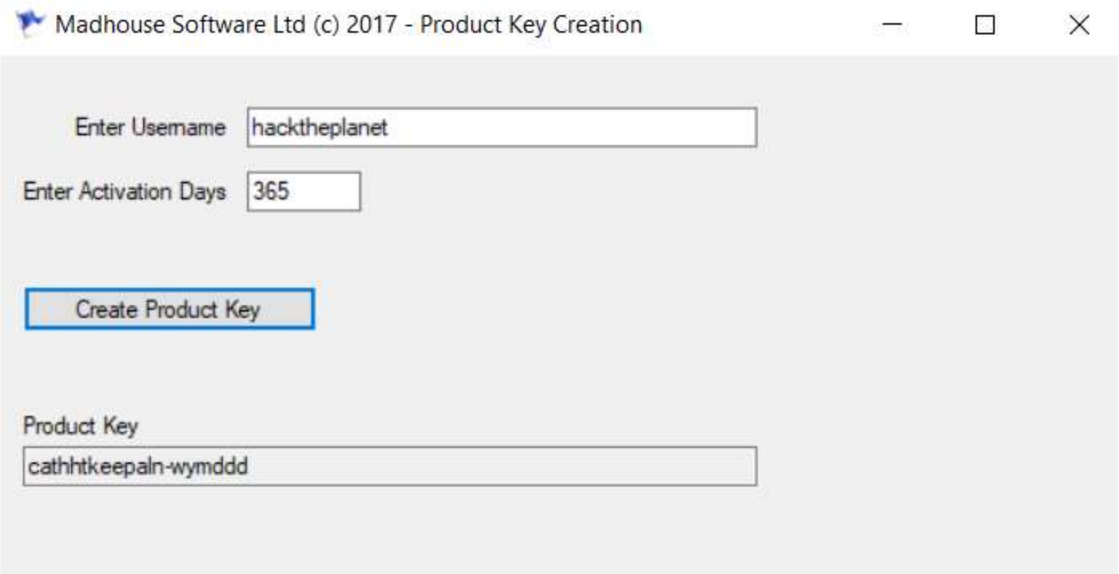
Hack The Box

PEN-TESTING LABS

Congratulations!

2.0 The Art of Reversing

1)After extracting the zip file, open the .exe file and try entering the password until the product key matches with the one given in the question.Keep retrying until you get it.



Madhouse Software Ltd (c) 2017 - Product Key Creation

Enter Username: hacktheplanet

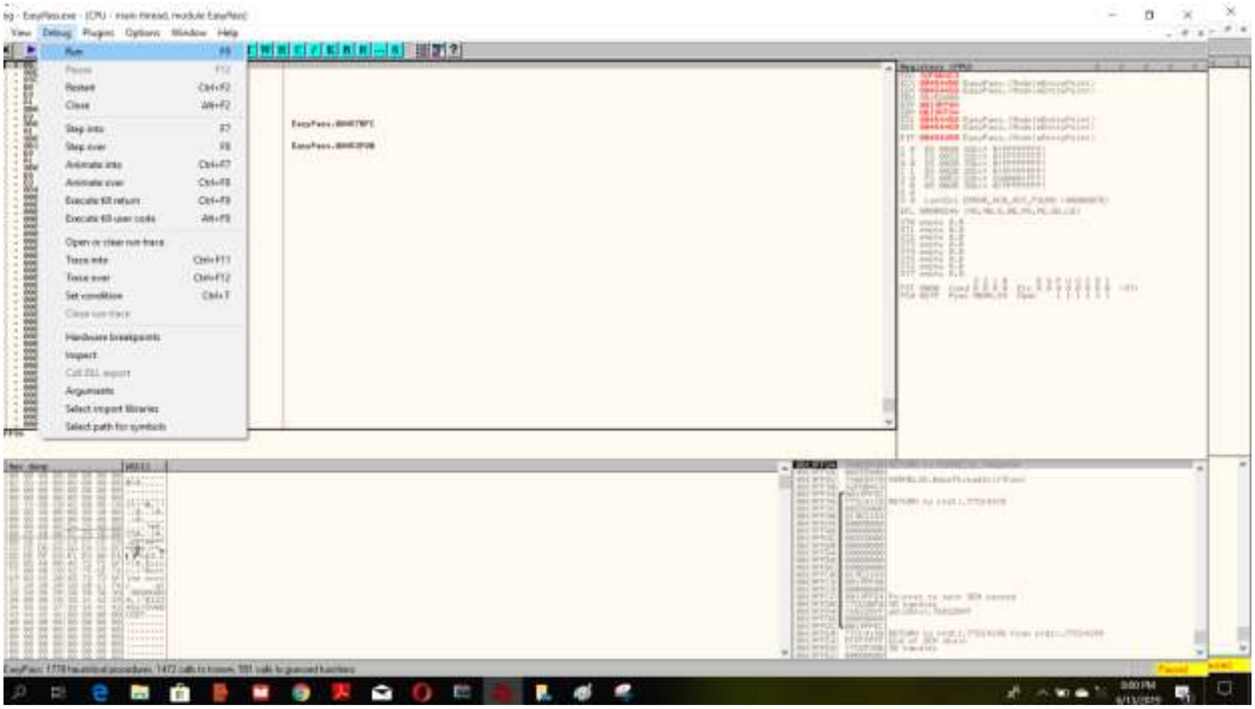
Enter Activation Days: 365

Product Key: cathhtkeepaln-wymddd

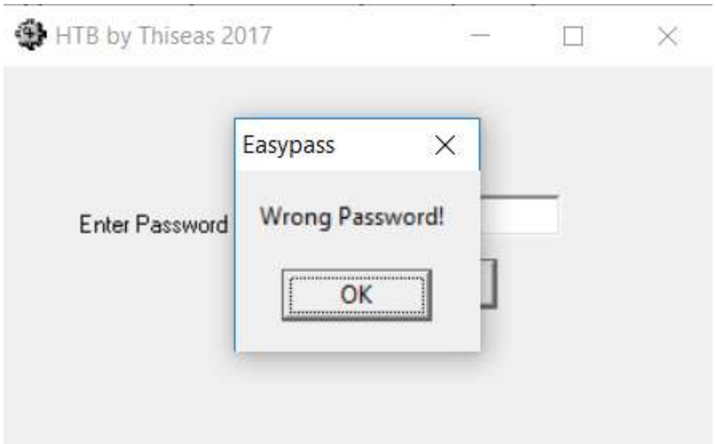
Flag: HTB{hacktheplanet365}..

3.0 Find the Easy Pass

1)Open the .exe file in Ollydbug. Navigate to debug>Run and run the program.

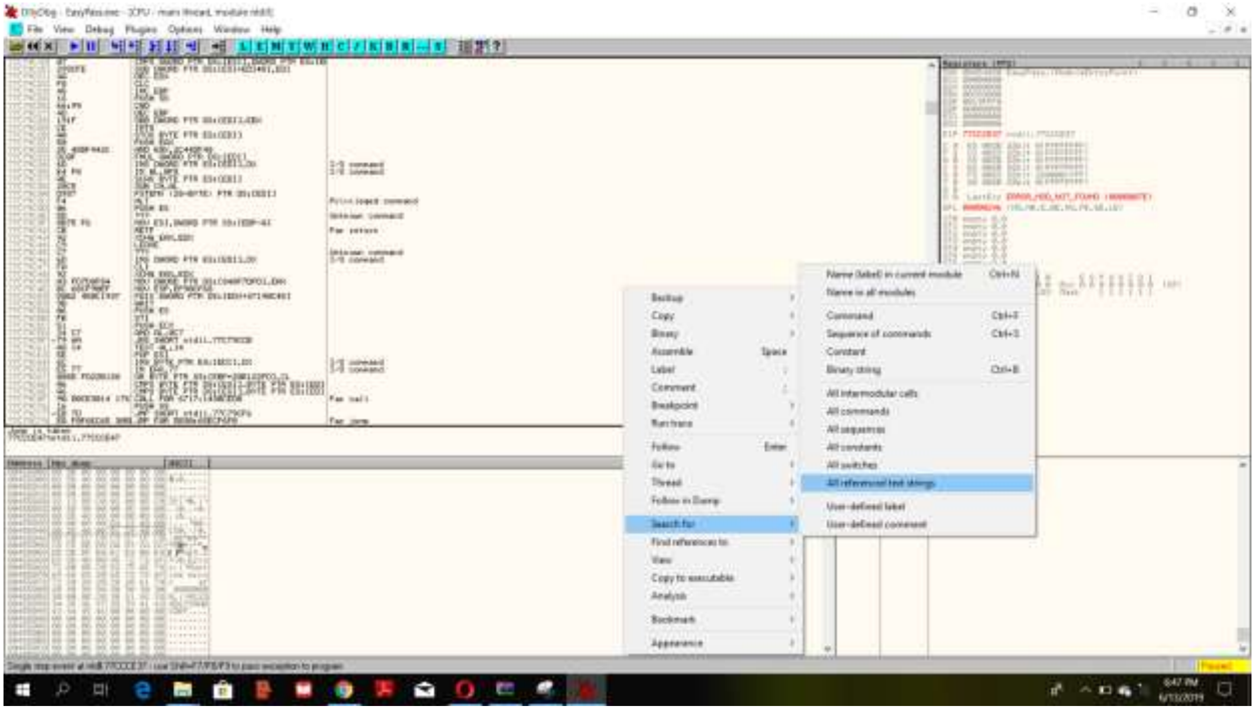


2)The program will run.Enter any password and the Wrong password error pop up.

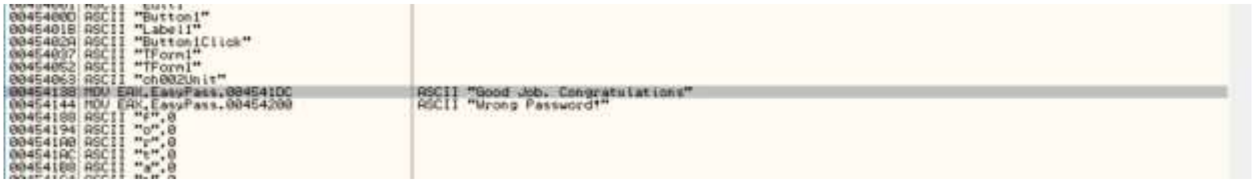


Wrong Password!

3)After that, right click on the comment section in the middle of the console as below and navigate to search for>all referenced text strings. We want to look for the location of the ‘Wrong password’ message in the register.

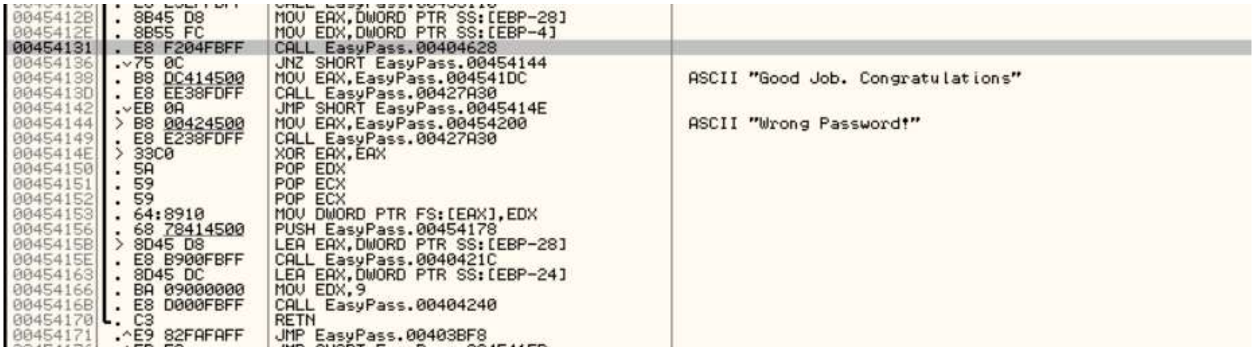


4)A window of referenced strings will pop up. Find the strings as below and double click on the very left of the line highlighted as below.



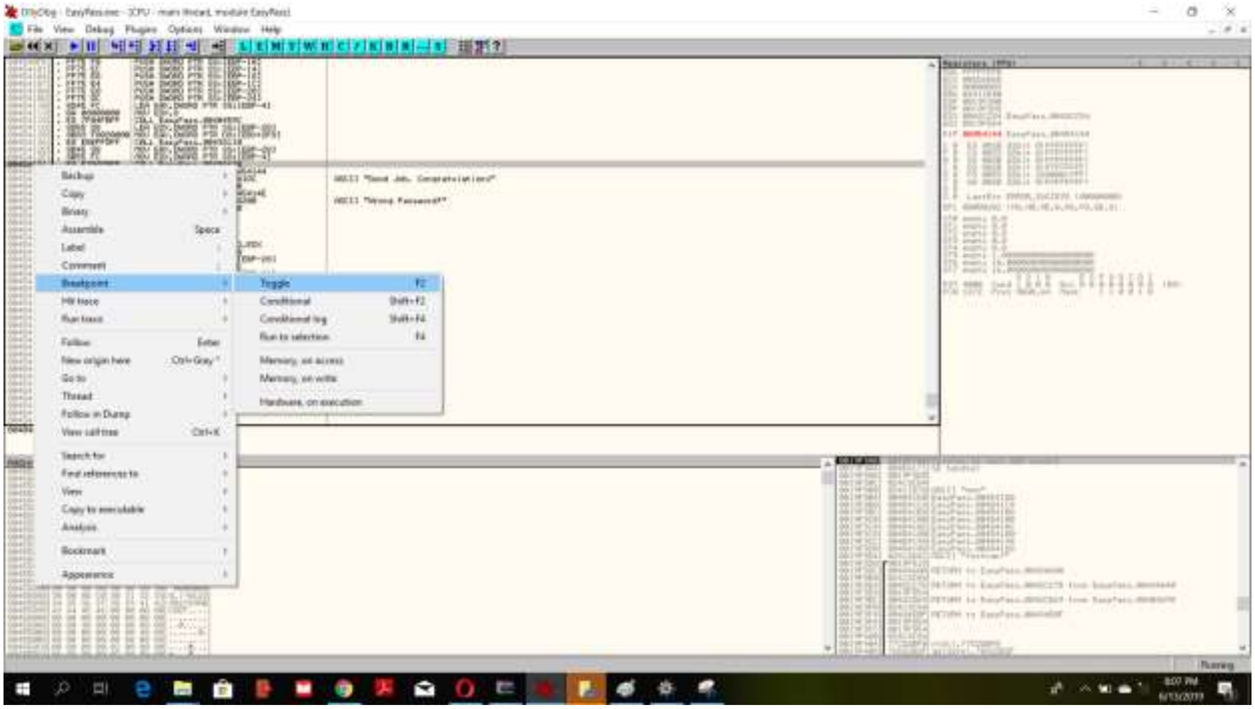
```
0045400D ASCII "Button1"
0045401B ASCII "Label1"
0045402A ASCII "Button1Click"
00454037 ASCII "TForm1"
00454052 ASCII "TForm1"
00454063 ASCII "ch002Unit"
00454138 MOV EAX,EasyPass.004541DC          ASCII "Good Job. Congratulations"
00454144 MOV EAX,EasyPass.00454200          ASCII "Wrong Password!"
00454188 ASCII "f",0
00454194 ASCII "o",0
004541A0 ASCII "r",0
004541AC ASCII "t",0
004541B8 ASCII "a",0
```

5)Another window will pop up showing the location and instruction matching with the strings. We need to choose which part of the instruction is there when the .exe file is comparing the string. The line highlighted below shows the call that has been made before JNZ instruction happened, which might be where the .exe compares the password entered by user with the password inside the program.

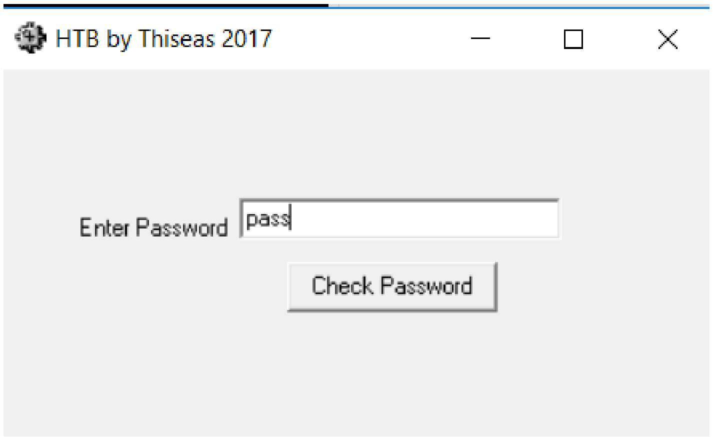


```
0045412B  . 8B45 D8       MOV EAX,DWORD PTR SS:[EBP-28]
0045412E  . 8B55 FC       MOV EDX,DWORD PTR SS:[EBP-4]
00454131  . E8 F204FBFF   CALL EasyPass.00404628
00454136  .v75 0C         JNZ SHORT EasyPass.00454144
00454138  . B8 DC414500   MOV EAX,EasyPass.004541DC         ASCII "Good Job. Congratulations"
0045413D  . E8 EE38FDFF   CALL EasyPass.00427A30
00454142  .vEB 0A         JMP SHORT EasyPass.0045414E
00454144  > B8 00424500   MOV EAX,EasyPass.00454200         ASCII "Wrong Password!"
00454149  . E8 E238FDFF   CALL EasyPass.00427A30
0045414E  > 33C0          XOR EAX,EAX
00454150  . 5A            POP EDX
00454151  . 59            POP ECX
00454152  . 59            POP ECX
00454153  . 64:8910       MOV DWORD PTR FS:[EAX],EDX
00454156  . 68 78414500   PUSH EasyPass.00454178
0045415B  > 8D45 D8       LEA EAX,DWORD PTR SS:[EBP-28]
0045415E  . E8 B900FBFF   CALL EasyPass.0040421C
00454163  . 8D45 DC       LEA EAX,DWORD PTR SS:[EBP-24]
00454166  . BA 09000000   MOV EDX,9
0045416B  . E8 D000FBFF   CALL EasyPass.00404240
00454170  L. C3           RETN
00454171  .^E9 82FAFAFF   JMP EasyPass.00403BF8
```

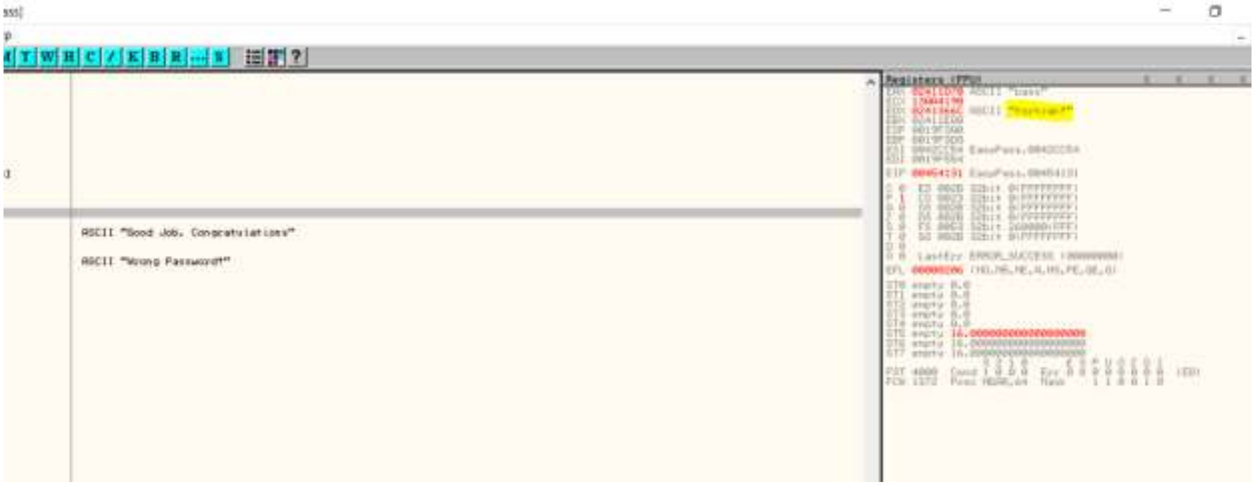
6)Now is the time we set breakpoint at the line highlighted in step 5. Right click on the register section in very left area of the lineand navigate to breakpoint>toggle.



7)Now, run the program and enter any password when the program opened. Click on check password.

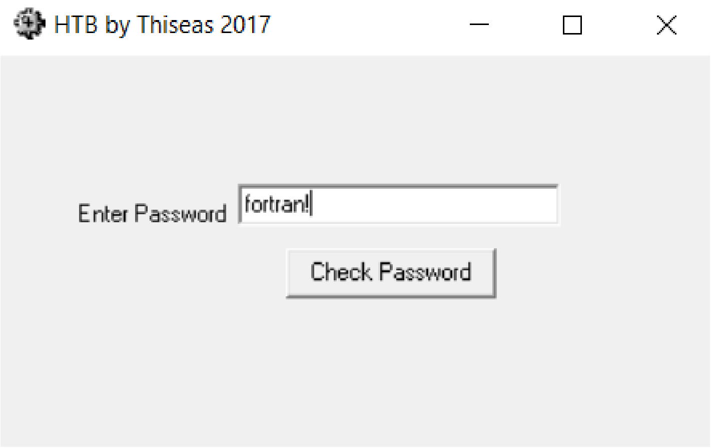


8)We see in the registers section on the left of the main section is changing. It seems that our entered password is loaded into EAX register and 'fortran!' is loaded in EDX. The ECX might be comparing both of the values. We deduce that 'fortran! ' might be the password.

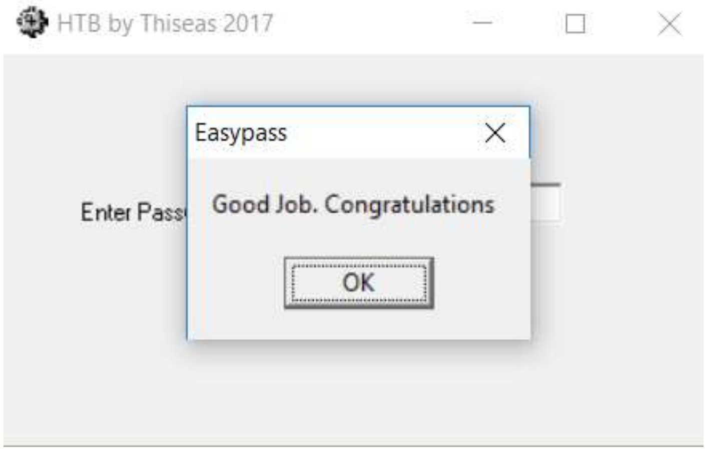

```
ASCII "Good Job. Congratulations"

ASCII "Wrong Password"
```

```
Registers (FPU)
EAX [illegible] ASCII "pass"
ECX [illegible]
EDX [illegible] ASCII "fortran!"
EBX [illegible]
ESP [illegible]
EBP [illegible]
ESI [illegible] EasyPass.[illegible]
EDI [illegible]
EIP [illegible] EasyPass.[illegible]
    LastErr ERROR_SUCCESS [illegible]
EFL [illegible]
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 16.[illegible]
ST6 empty 16.[illegible]
ST7 empty 16.[illegible]
```

9)Enter the 'fortran!' Password.



10)A congratulations message appeared. We have found the flag.



HTB by Thiseas 2017

Enter Pass

Easypass

Good Job. Congratulations

Flag:HTB{fortran!}

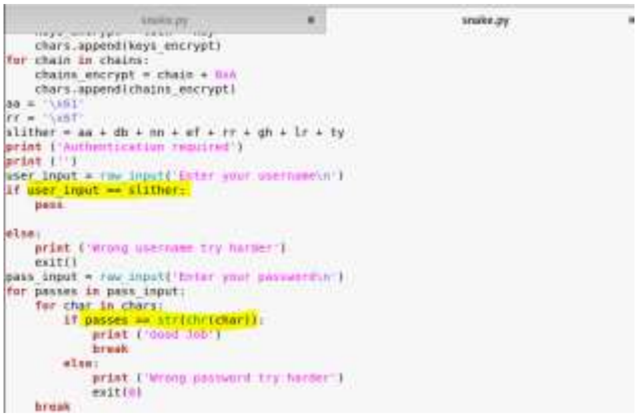
4.0 Snake

1)First, we view the snake.py file. Notice that

user_input=slither

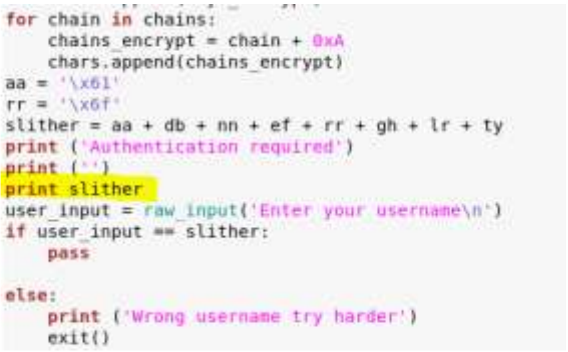
and

passes=str(chr(char)).



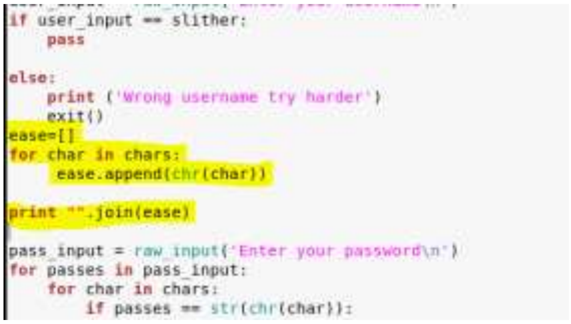
```
    chars.append(keys_encrypt)
for chain in chains:
    chains_encrypt = chain + 0xA
    chars.append(chains_encrypt)
aa = '\x61'
rr = '\x67'
slither = aa + db + nn + ef + rr + gh + lr + ty
print ('Authentication required')
print ('')
user_input = raw_input('Enter your username\n')
if user_input == slither:
    pass

else:
    print ('Wrong username try harder')
    exit()
pass_input = raw_input('Enter your password\n')
for passes in pass_input:
    for char in chars:
        if passes == str(chr(char)):
            print ('Good Job')
            break
        else:
            print ('Wrong password try harder')
            exit(0)
    break
```

2)We can manipulate this by telling the program to print ‘slither’ for us to get the username by adding the line highlighted.



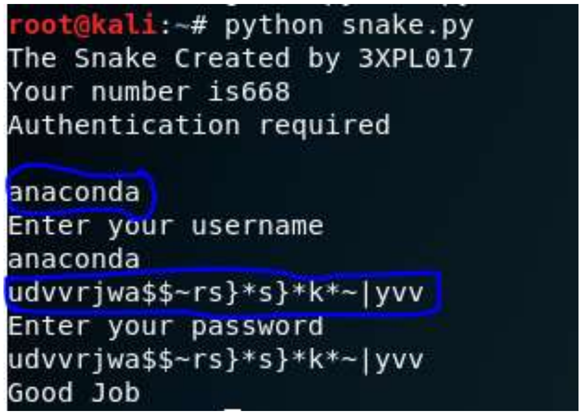
```
for chain in chains:
    chains_encrypt = chain + 0xA
    chars.append(chains_encrypt)
aa = '\x61'
rr = '\x6f'
slither = aa + db + nn + ef + rr + gh + lr + ty
print ('Authentication required')
print ('')
print slither
user_input = raw_input('Enter your username\n')
if user_input == slither:
    pass

else:
    print ('Wrong username try harder')
    exit()
```

3)Then, we add code to create new array and append each of the character array ‘chars’. We use chr() function to add character elements to our new array rather than ASCII.



```
if user_input == slither:
    pass

else:
    print ('Wrong username try harder')
    exit()
ease=[]
for char in chars:
     ease.append(chr(char))

print "".join(ease)

pass_input = raw_input('Enter your password\n')
for passes in pass_input:
    for char in chars:
        if passes == str(chr(char)):
```

4)So,we run the program and it will print both username and password for us.



```
root@kali:~# python snake.py
The Snake Created by 3XPL017
Your number is668
Authentication required

anaconda
Enter your username
anaconda
udvvrjwa$$~rs}*s}*k*~|yvv
Enter your password
udvvrjwa$$~rs}*s}*k*~|yvv
Good Job
```

The flag should be in the format HTB{username:password}.

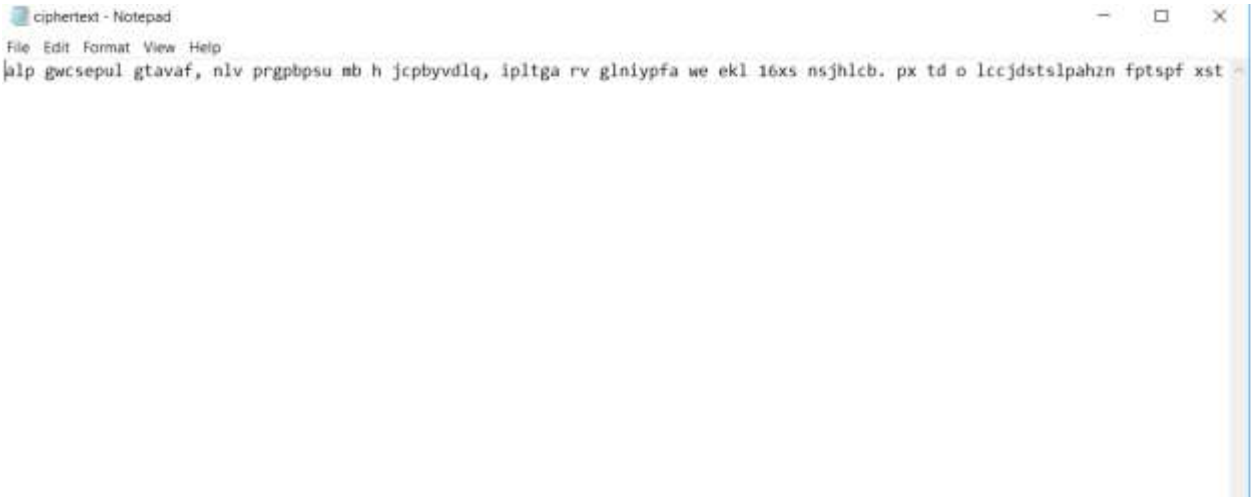
We only have to pick the earliest 10 characters from the password printed.

Flag:

HTB{anaconda:udvvrjwa$$}

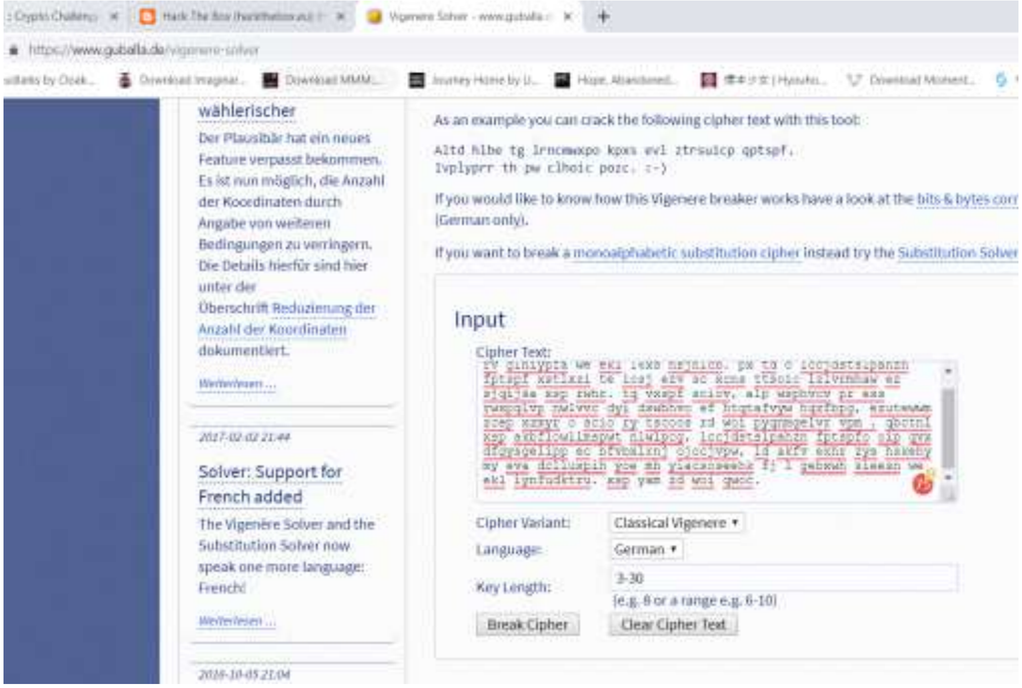
5.0 Classic, yet complicated

1)After extracting the .zip file we found this text file. Inside is a cipher text.



```
alp gwcsepul gtavaf, nlv prgpbpsu mb h jcpbyvdlq, ipltga rv glniypfa we ekl 16xs nsjhlcb. px td o lccjdstslpahzn fptspf xst
```

2)We use the vigenere cipher decoder in the url https://www.guballa.de/vigenere-solver. It can break vigenere ciphers without knowing the key.


Vigenere Solver - www.guballa.de
https://www.guballa.de/vigenere-solver

wählerischer

Der Plausibär hat ein neues Feature verpasst bekommen. Es ist nun möglich, die Anzahl der Koordinaten durch Angabe von weiteren Bedingungen zu verringern. Die Details hierfür sind hier unter der Überschrift Reduzierung der Anzahl der Koordinaten dokumentiert.

Weiterlesen ...

2017-02-02 21:44

Solver: Support for French added

The Vigenère Solver and the Substitution Solver now speak one more language: French!

Weiterlesen ...

2016-10-05 21:04

As an example you can crack the following cipher text with this tool:

```
Altd hlbe tg lrncmwxpo kpxs evl ztrsuicp qptspf.
Ivplyprr th pw clhoic pozc. :-)
```

If you would like to know how this Vigenere breaker works have a look at the bits & bytes corr (German only).

If you want to break a monoalphabetic substitution cipher instead try the Substitution Solver

Input

Cipher Text:

```
rv glniypfa we ekl 16xs nsjhlcb. px td o lccjdstslpahzn
fptspf xstlxzi te lcsj ezv sc xcms ttsoic lzlvinhaw ez
zjqijse xsp rwhc. tg vxspf snisv, alp wsphvcv pr ess
rwspqlvp rwlvvc dyi dswbhvo ef htqtafvyw hqrfbpg, ezutewwm
zcep xzsyr o scio ry tscoos rd wol pygmgelvr ypm , qbotnl
xsp akbflcwilsspwt nlwlpcy, lccjdstslpahzn fptspfo slp qyx
dfqyagelipp ec bfvbalxnj ojccjvpw, ld akfv exhr zys hakeby
xy eve dclluxpih ypw mh yiacaseehz fj l gebxwh zlesan we
ekl iynfudktru. xsp ywm zd wol qmoc.
```

Cipher Variant: Classical Vigenere ▾

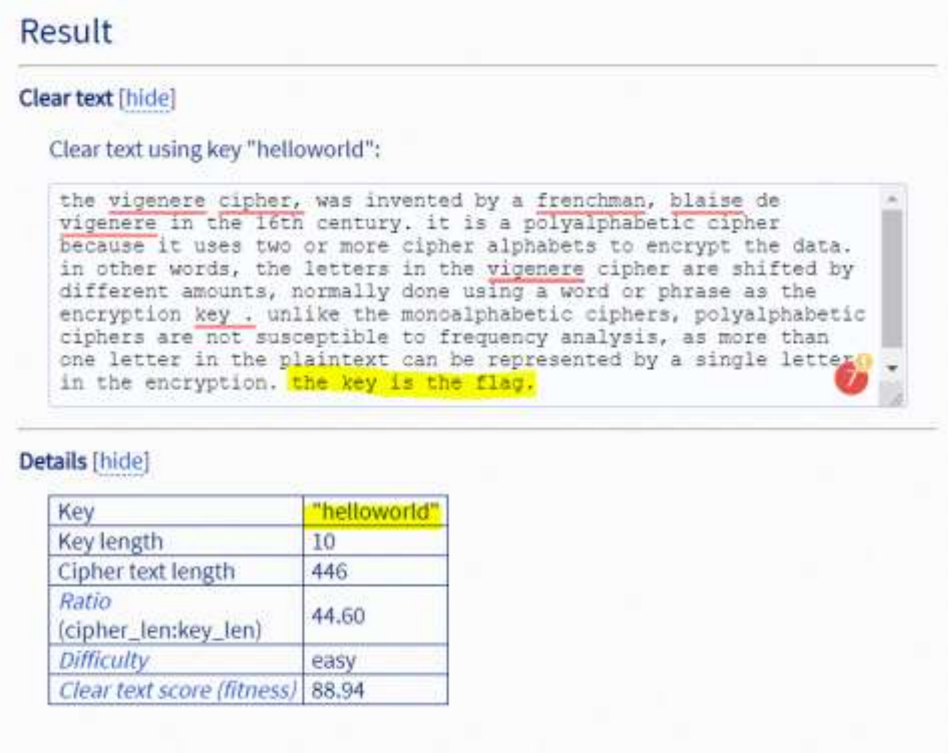
Language: German ▾

Key Length: 3-30

(e.g. 8 or a range e.g. 6-10)

Break Cipher    Clear Cipher Text

3)We found this in the result section. It says that the key is the flag. Hence , “helloworld” is the flag.



Result

Clear text [hide]

Clear text using key "helloworld":

```
the vigenere cipher, was invented by a frenchman, blaise de
vigenere in the 16th century. it is a polyalphabetic cipher
because it uses two or more cipher alphabets to encrypt the data.
in other words, the letters in the vigenere cipher are shifted by
different amounts, normally done using a word or phrase as the
encryption key . unlike the monoalphabetic ciphers, polyalphabetic
ciphers are not susceptible to frequency analysis, as more than
one letter in the plaintext can be represented by a single letter
in the encryption. the key is the flag.
```

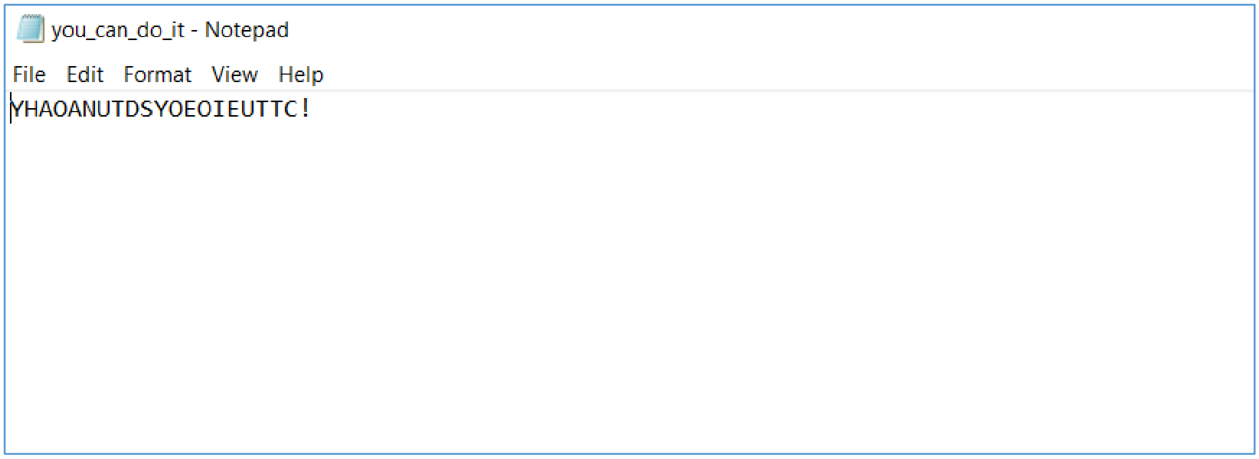
Details [hide]

| | |
|---|---|
| Key | "helloworld" |
| Key length | 10 |
| Cipher text length | 446 |
| *Ratio*<br>(cipher_len:key_len) | 44.60 |
| *Difficulty* | easy |
| *Clear text score (fitness)* | 88.94 |

Flag: HTB{helloworld}

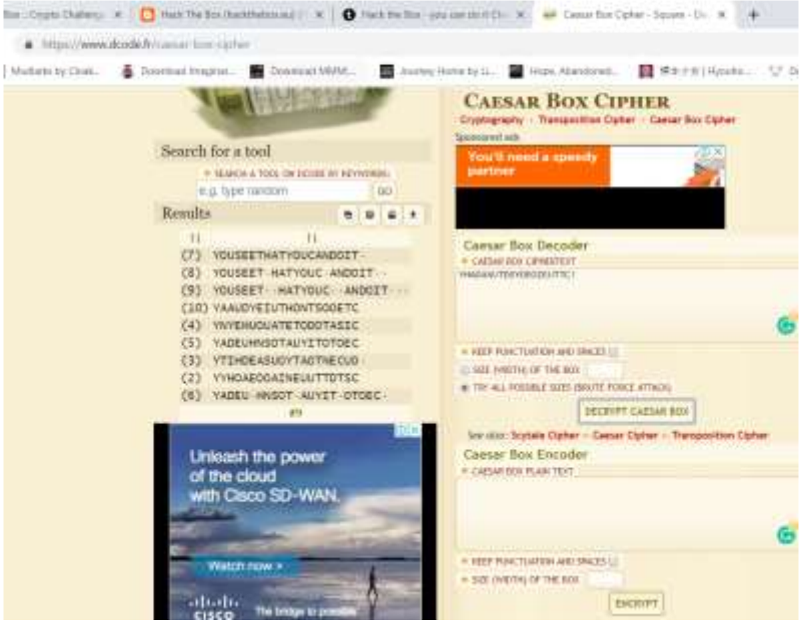
6.0 You can do it

1) We saw the encrypted text in a txt file after unzipping the content.



```
YHAOANUTDSYOEOIEUTTC!
```

2) This word is a form of Caesar Cipher. We use tool in https://www.dcode.fr/caesar-box-cipher to decode the text. One of the results show 'YOUSEETHATYOUCANDOIT'. In the original text there is the '!' symbol. We must include that when entering the flag in the Hack the box website. After entering it, we find that this is our flag.

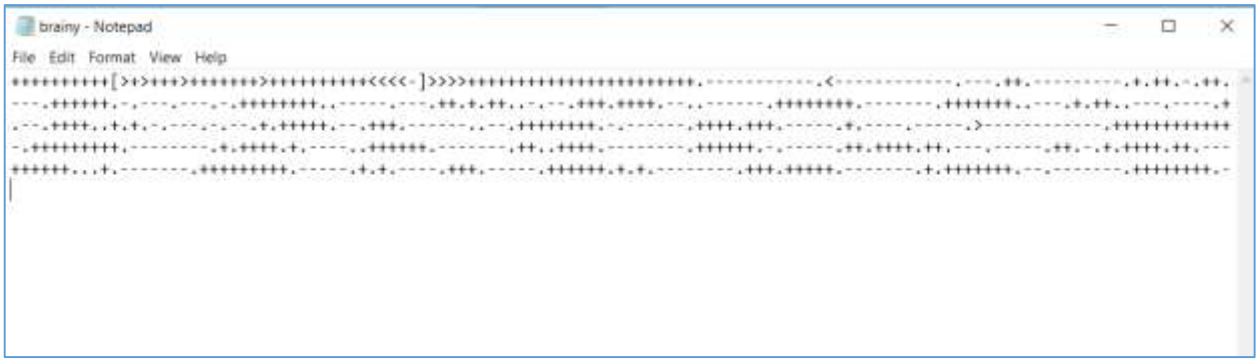

Search for a tool
e.g. type random
GO
Results
CAESAR BOX CIPHER
Cryptography › Transposition Cipher › Caesar Box Cipher
Caesar Box Decoder
CAESAR BOX CIPHERTEXT
KEEP PUNCTUATION AND SPACES
SIZE (WIDTH) OF THE BOX
TRY ALL POSSIBLE SIZES (BRUTE FORCE ATTACK)
DECRYPT CAESAR BOX
See also: Scytale Cipher – Caesar Cipher – Transposition Cipher
Caesar Box Encoder
CAESAR BOX PLAIN TEXT
KEEP PUNCTUATION AND SPACES
SIZE (WIDTH) OF THE BOX
ENCRYPT

```
(7)  YOUSEETHATYOUCANDOIT
(8)  YOUSEET HATYOUC ANDOIT
(9)  YOUSEET  HATYOUC  ANDOIT
(10) YAAUDYEIUTHONTSODETC
(4)  YNYEHUOUATETODOTASIC
(5)  YADEUHNSOTAUYITOTOEC
(3)  YTIHDEASUOYTAOTNECUO
(2)  YYHOAEOOAINEUUTTOTSC
(6)  YADEU HNSOT AUYIT OTOEC
```

Flag: HTB{YOUSEETHATYOUCANDOIT!}

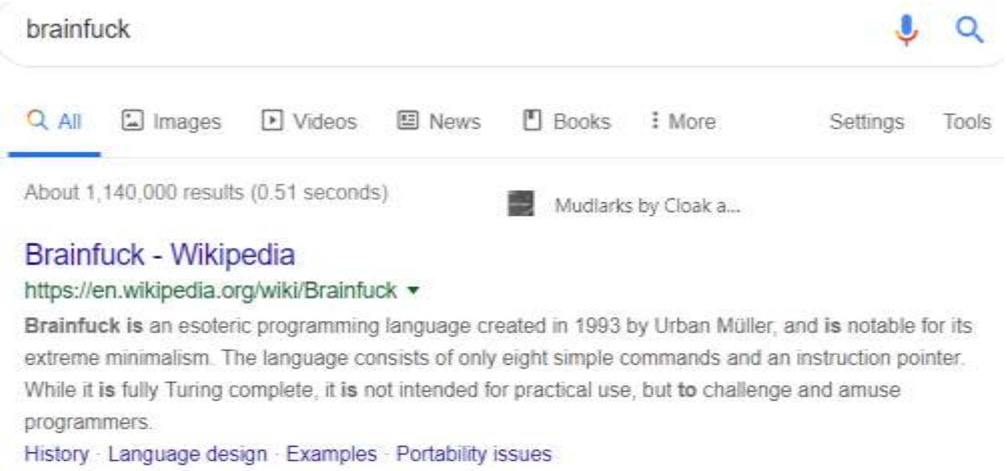
7.0 Brainy's cipher

1)We find an encrypted text when opening the .txt file from the downloaded .zip file.



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

2)After a bit of googling , we come across the Brainfuck language.



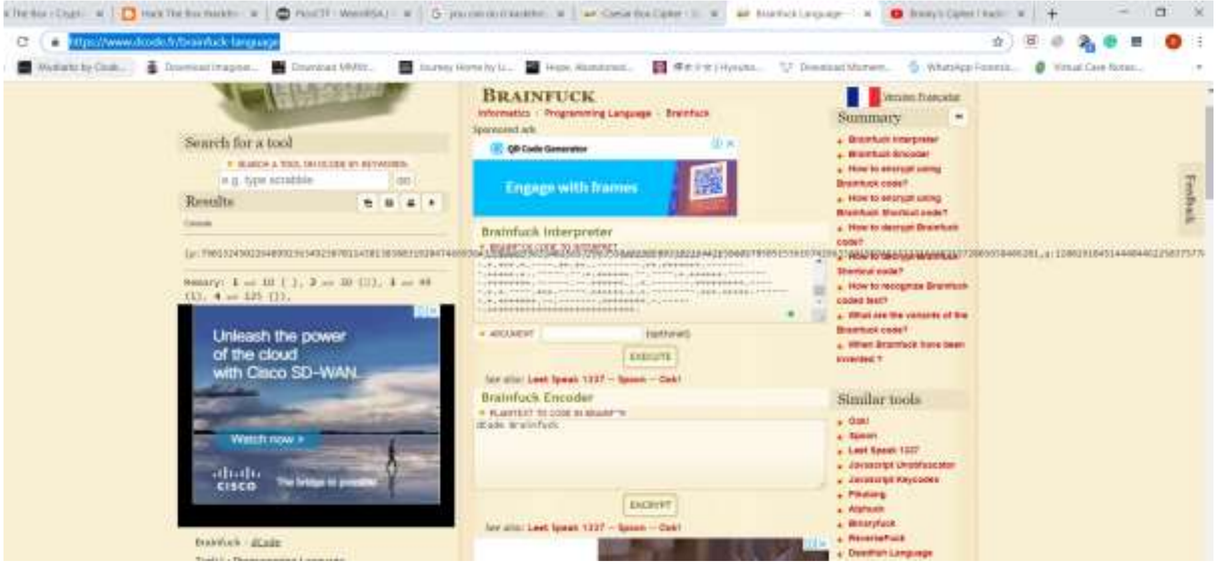
brainfuck

About 1,140,000 results (0.51 seconds)

Brainfuck - Wikipedia
https://en.wikipedia.org/wiki/Brainfuck

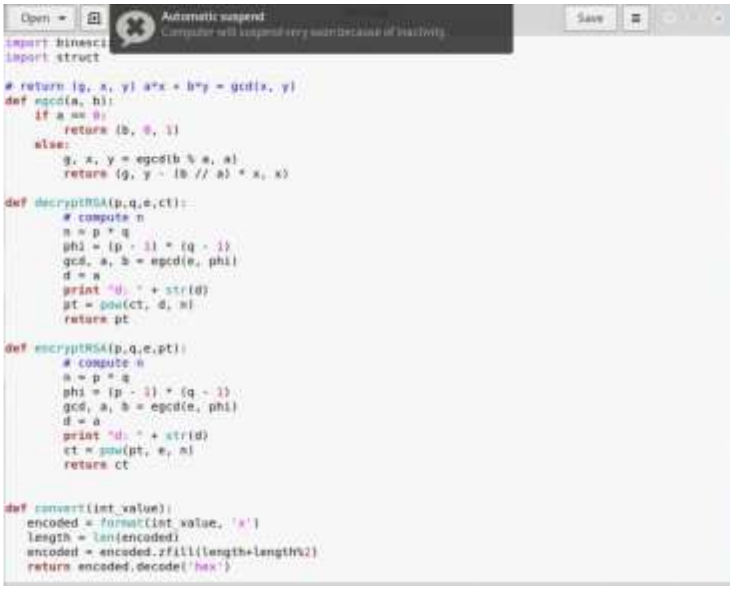
**Brainfuck is** an esoteric programming language created in 1993 by Urban Müller, and **is** notable for its extreme minimalism. The language consists of only eight simple commands and an instruction pointer. While it **is** fully Turing complete, it **is** not intended for practical use, but **to** challenge and amuse programmers.

3)We use tool in https://www.dcode.fr/brainfuck-language to decode it. We find that there is a string similar to weirdRSA after decrypting the Brainfuck text.



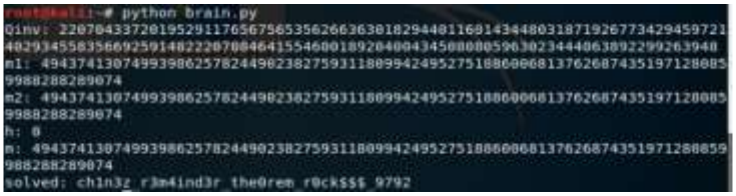
https://www.dcode.fr/brainfuck-language

4)This is asymmetric cryptography . We can create code in python to decrypt the code.



```
import binascii
import struct

# return (g, x, y) a*x + b*y = gcd(x, y)
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)

def decryptRSA(p,q,e,ct):
        # compute n
        n = p * q
        phi = (p - 1) * (q - 1)
        gcd, a, b = egcd(e, phi)
        d = a
        print "d: " + str(d)
        pt = pow(ct, d, n)
        return pt

def encryptRSA(p,q,e,pt):
        # compute n
        n = p * q
        phi = (p - 1) * (q - 1)
        gcd, a, b = egcd(e, phi)
        d = a
        print "d: " + str(d)
        ct = pow(pt, e, n)
        return ct


def convert(int_value):
   encoded = format(int_value, 'x')
   length = len(encoded)
   encoded = encoded.zfill(length+length%2)
   return encoded.decode('hex')
```

5)After decoding using python we found the flag.

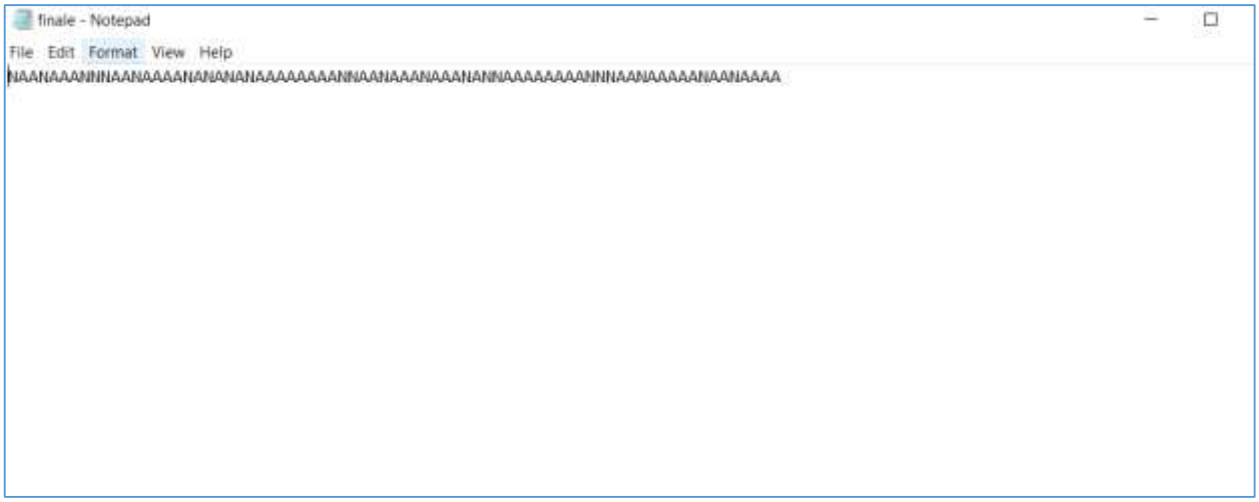


```
root@kali:~# python brain.py
Qinv: 22070433720195291176567565356266363018294401169143448031871926773429459721
40293455835669259148222070084641554600189204004345080805963023444063892299263940
m1: 49437413074993986257824490238275931180994249527518860068137626874351971280085
9988288289074
m2: 49437413074993986257824490238275931180994249527518860068137626874351971280085
9988288289074
h: 0
m: 494374130749939862578244902382759311809942495275188600681376268743519712880859
988288289074
solved: ch1n3z_r3m4ind3r_the0rem_r0ck$$$_9792
```

Flag: HTB{ch1n3z_r3n4indir_the0rem_rock$$$_9792}

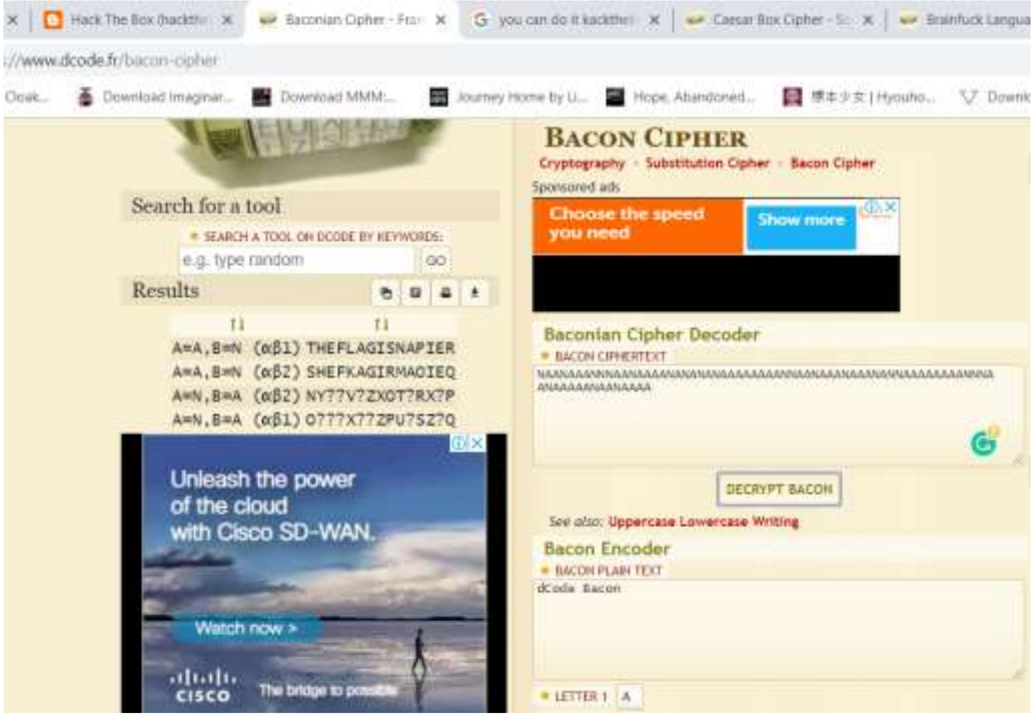
8.0 Deceitful batman

1)We find this text after opening the .zip file.



```
NAANAAANNNAANAAAANANANANAAAAAAAAANNAANAAANAAANANNAAAAAAAAANNNAANAAAAAAANAANAAAA
```

2)The encrypted text is familiar to Bacon Cipher. Hence , we use tool in [https://www.dcode.fr/bacon-cipher](https://www.dcode.fr/bacon-cipher) to decrypt the text. We find 'THEFLAGISNAPIER' in the results. Hence, the flag is 'NAPIER'.



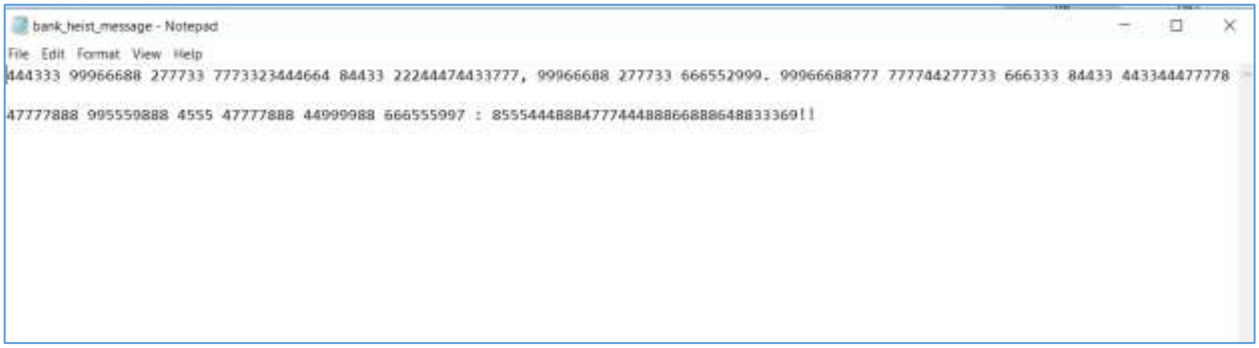
```
A=A,B=N (αβ1) THEFLAGISNAPIER
A=A,B=N (αβ2) SHEFKAGIRMAOIEQ
A=N,B=A (αβ2) NY??V?ZXOT?RX?P
A=N,B=A (αβ1) O???X??ZPU?5Z?Q
```

[illegible]

Flag: HTB{NAPIER}

## 9.0 Bank heist

1)After opening the unzipped file you will get this in a text file:

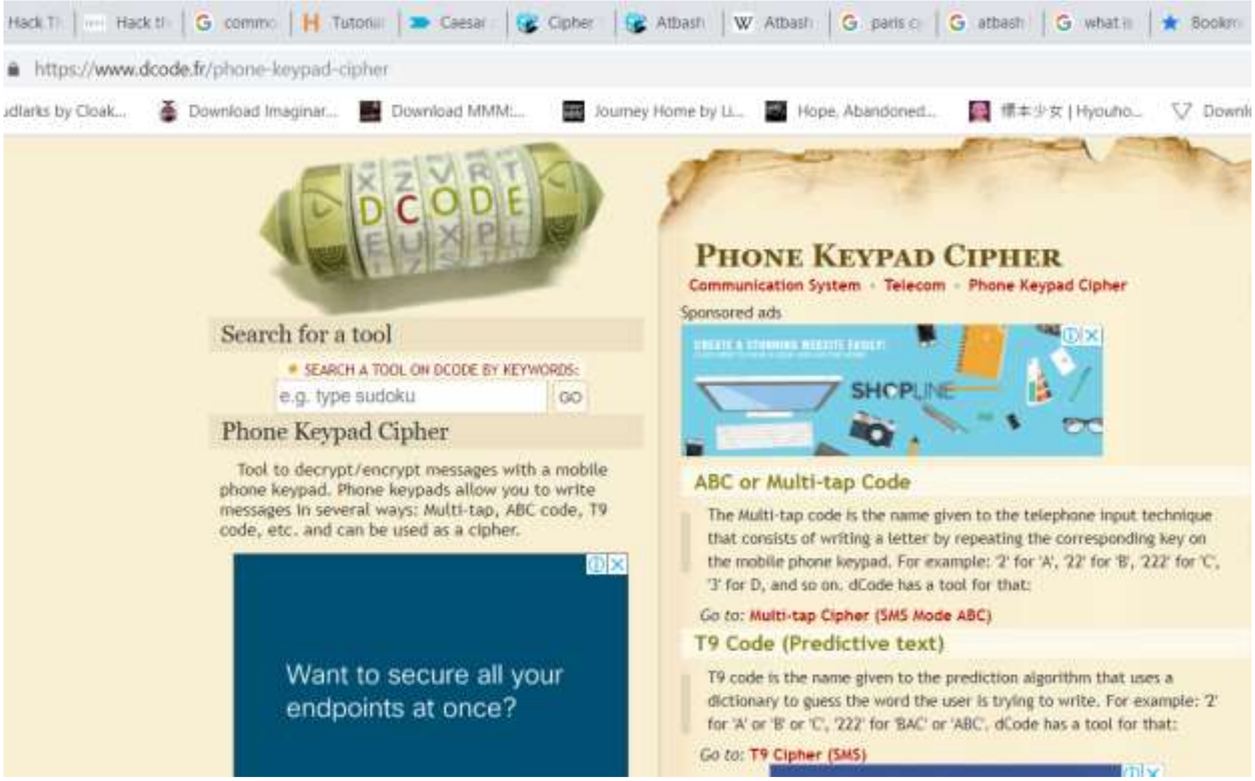


```
444333 99966688 277733 7773323444664 84433 22244474433777, 99966688 277733 666552999. 99966688777 777744277733 666333 84433 443344477778

47777888 995559888 4555 47777888 44999988 666555997 : 8555444888477744488866888648833369!!
```

2)In the question it hinted that the criminal used an old phone.

“Under further analysis of the persons flip phone you see a message that seems suspicious. Can  you figure out what the message to put this guy in jail? “

Hence, we take this as a clue and google some ciphers related to old phones. We find out that there are two ciphers related to phones, ABC and T9 cipher.



https://www.dcode.fr/phone-keypad-cipher

Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type sudoku GO

Phone Keypad Cipher

Tool to decrypt/encrypt messages with a mobile phone keypad. Phone keypads allow you to write messages in several ways: Multi-tap, ABC code, T9 code, etc. and can be used as a cipher.

PHONE KEYPAD CIPHER

Communication System › Telecom › Phone Keypad Cipher

Sponsored ads

ABC or Multi-tap Code

The Multi-tap code is the name given to the telephone input technique that consists of writing a letter by repeating the corresponding key on the mobile phone keypad. For example: '2' for 'A', '22' for 'B', '222' for 'C', '3' for D, and so on. dCode has a tool for that:

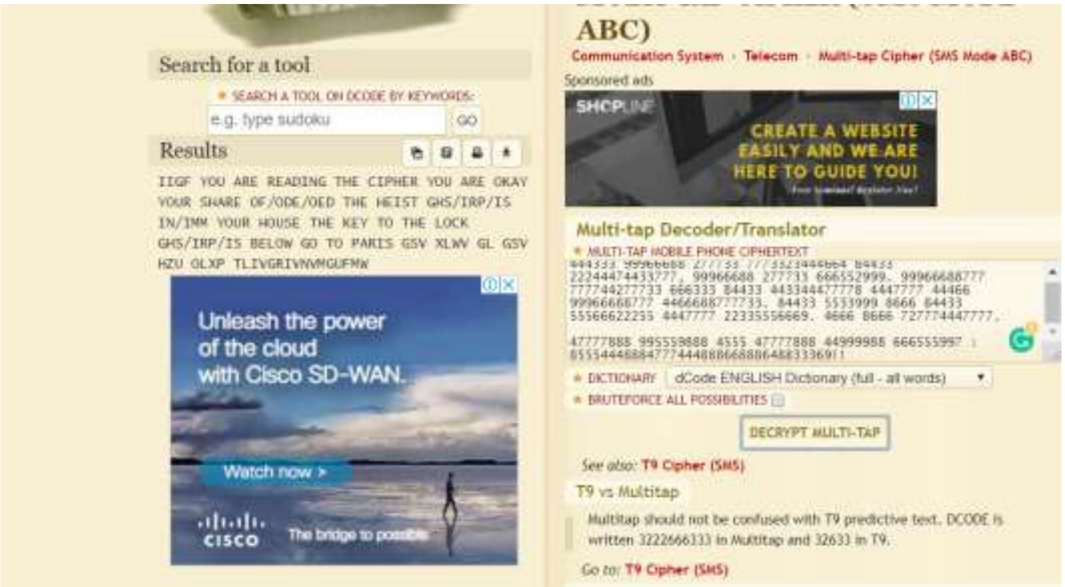
Go to: Multi-tap Cipher (SMS Mode ABC)

T9 Code (Predictive text)

T9 code is the name given to the prediction algorithm that uses a dictionary to guess the word the user is trying to write. For example: '2' for 'A' or 'B' or 'C', '222' for 'BAC' or 'ABC'. dCode has a tool for that:

Go to: T9 Cipher (SMS)

3)We decode the encrypted text in ABC decoder at https://www.dcode.fr/multitap-abc-cipher.



Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type sudoku GO

Results

```
IIGF YOU ARE READING THE CIPHER YOU ARE OKAY
YOUR SHARE OF/ODE/OED THE HEIST GHS/IRP/IS
IN/IMM YOUR HOUSE THE KEY TO THE LOCK
GHS/IRP/IS BELOW GO TO PARIS GSV XLWV GL GSV
HZU OLXP TLIVGRIVNVMGUFMW
```

ABC)

Communication System › Telecom › Multi-tap Cipher (SMS Mode ABC)

Sponsored ads

Multi-tap Decoder/Translator

MULTI-TAP MOBILE PHONE CIPHERTEXT

```
444333 99966688 27733 7773323444664 84433
2224447443377, 99966688 277733 666552999. 99966688777
777744277733 666333 84433 4433444777 8 4447777 44466
9996666777 4466688777733. 84433 5553999 8666 84433
55566622255 4447777 22333555666 9. 4666 8666 7277744447777.

47777888 995559888 4555 47777888 44999988 66655599 7 :
855544488847777444888666888 64883336 9!!
```

DICTIONARY dCode ENGLISH Dictionary (full - all words) ▼

BRUTEFORCE ALL POSSIBILITIES □

DECRYPT MULTI-TAP

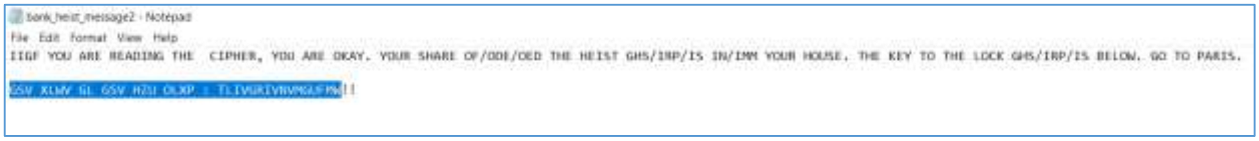
See also: T9 Cipher (SMS)

T9 vs Multitap

Multitap should not be confused with T9 predictive text. DCODE is written 3222666333 in Multitap and 32633 in T9.

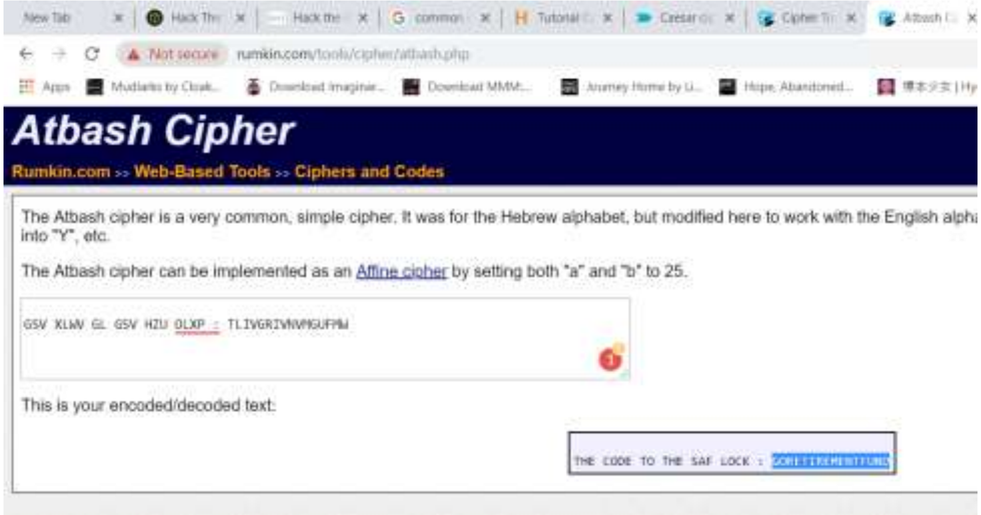
Go to: T9 Cipher (SMS)

4)The decoded message reads as below, but the line at the bottom does not seem to be understandable.



```
IIGF YOU ARE READING THE  CIPHER, YOU ARE OKAY. YOUR SHARE OF/ODE/OED THE HEIST GHS/IRP/IS IN/IMM YOUR HOUSE. THE KEY TO THE LOCK GHS/IRP/IS BELOW. GO TO PARIS.

GSV XLWV GL GSV HZU OLXP : TLIVGRIVNVMGUFMW!!
```

5)We try to decode the line with some common cipher decoders and found that the cipher is Atbash, a cipher based on Hebrew language. After decoding the message we get the flag.



Atbash Cipher

Rumkin.com >> Web-Based Tools >> Ciphers and Codes

The Atbash cipher is a very common, simple cipher. It was for the Hebrew alphabet, but modified here to work with the English alpha into "Y", etc.

The Atbash cipher can be implemented as an Affine cipher by setting both "a" and "b" to 25.

```
GSV XLWV GL GSV HZU OLXP : TLIVGRIVNVMGUFMW
```

This is your encoded/decoded text:

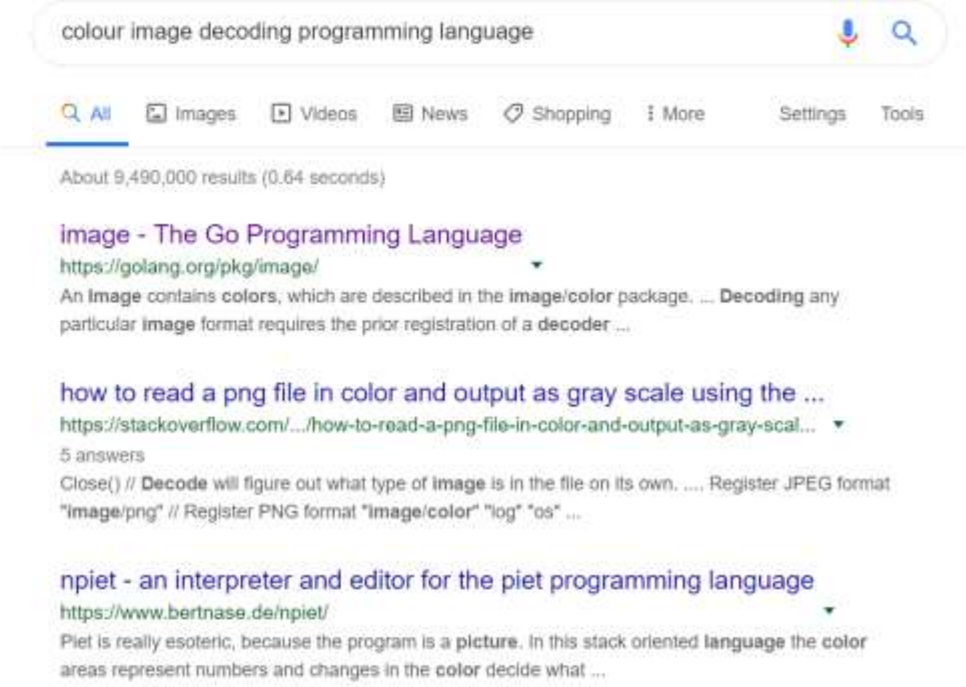
```
THE CODE TO THE SAF LOCK : GORETIREMENTFUND
```

Flag: HTB{GORETIREMENTFUND!!}

10.0 Art

1)After failing to find anything in the art.png image using various tools such as base64 png decoder , binwalk , stegsolve, etc, we try to find answer by googling about the language used to create the image.

We search 'colour image decoding programming language' and come up with a few links.



colour image decoding programming language

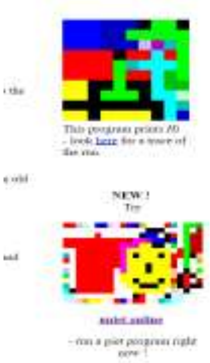
About 9,490,000 results (0.64 seconds)

image - The Go Programming Language
https://golang.org/pkg/image/
An Image contains colors, which are described in the image/color package. ... Decoding any particular image format requires the prior registration of a decoder ...

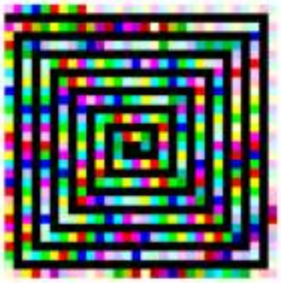
how to read a png file in color and output as gray scale using the ...
https://stackoverflow.com/.../how-to-read-a-png-file-in-color-and-output-as-gray-scal...
5 answers
Close() // Decode will figure out what type of image is in the file on its own. .... Register JPEG format "image/png" // Register PNG format "image/color" "log" "os" ...

npiet - an interpreter and editor for the piet programming language
https://www.bertnase.de/npiet/
Piet is really esoteric, because the program is a picture. In this stack oriented language the color areas represent numbers and changes in the color decide what ...

2)We find that on one of the links listed,in https://www.bertnase.de/npiet/ , there seems to be images similar to art.png.

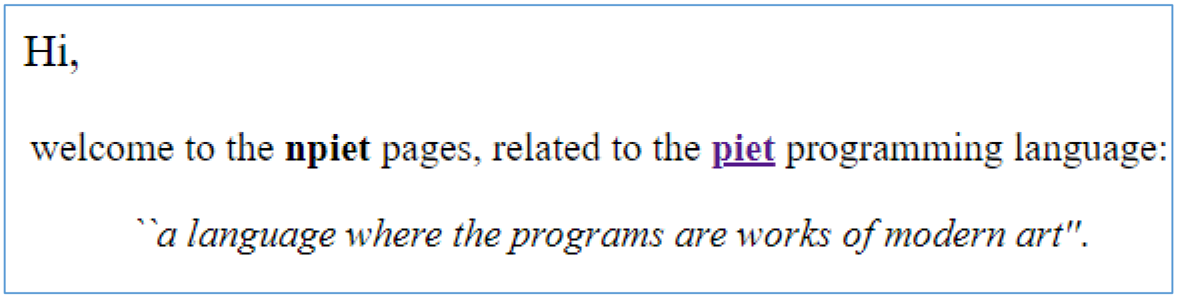


Images in bernatse.de/npiet



art.png

3)It seems that the language used is Piet language, a programming language in which programs look like abstract paintings. It is an esocentric language, which is is a programming language designed to test the boundaries of computer programming language design, as a proof of concept, as software art, as a hacking interface to another language (particularly functional programming or procedural programming languages), or as a joke.

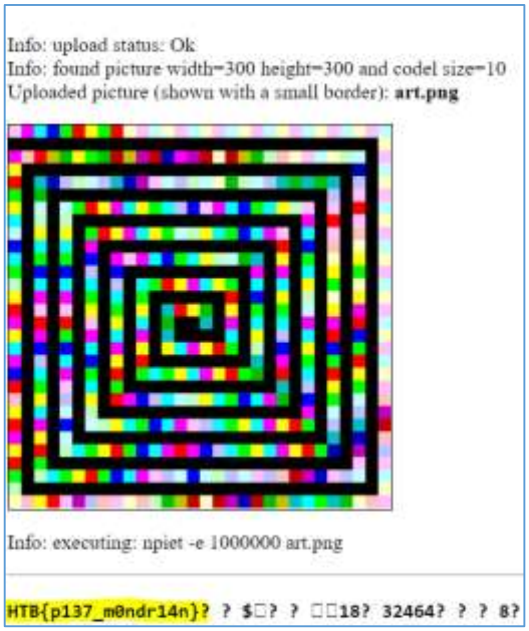


Hi,

welcome to the **npiet** pages, related to the piet programming language:

*``a language where the programs are works of modern art''.*

4)We upload the image in online tool to interpret piet language in https://www.bertnase.de/npiet/npiet-execute.php and find the flag.



Info: upload status: Ok
Info: found picture width=300 height=300 and codel size=10
Uploaded picture (shown with a small border): **art.png**

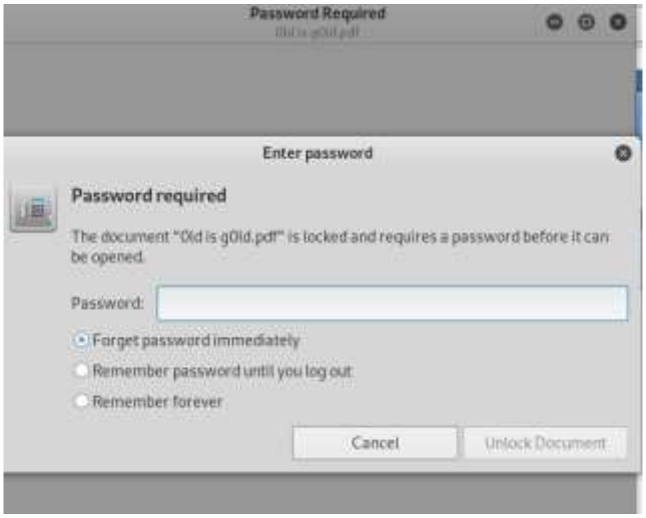
Info: executing: npiet -e 1000000 art.png

```
HTB{p137_m0ndr14n}? ? $□? ? □□18? 32464? ? ? 8?
```

Flag: HTB{p137_m0ndr14n}

## 11.0 Old is gold

1)we found out that the pdf file ‘Old is g0ld.pdf’ is password protected.



Password Required
Old is g0ld.pdf

Enter password

**Password required**

The document "Old is g0ld.pdf" is locked and requires a password before it can be opened.

Password:

● Forget password immediately
○ Remember password until you log out
○ Remember forever

2)Hence, after searching for pdf password cracking tool, we found pdfcrack in kali linux.

Use these commands in Kali Linux.

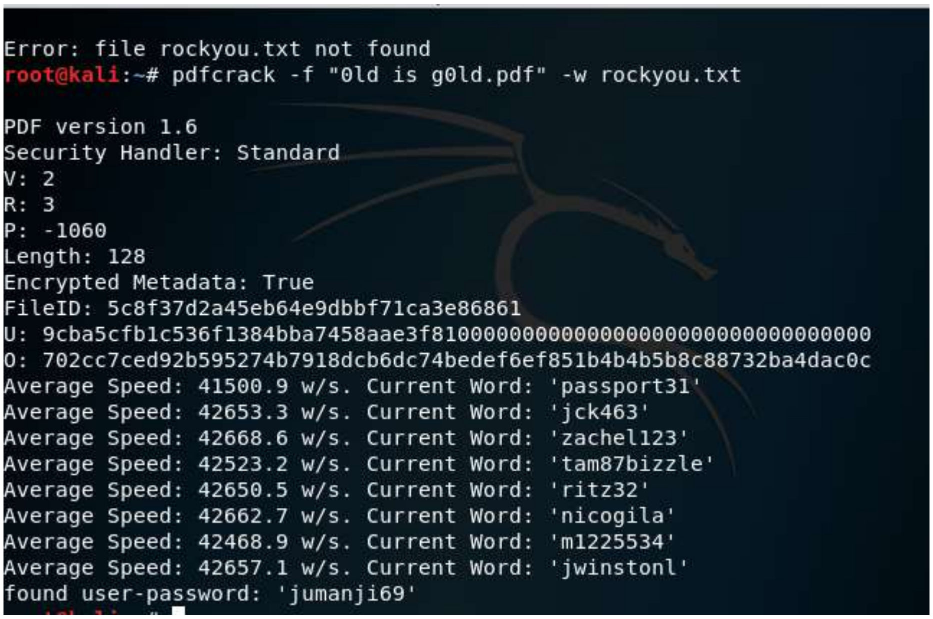
**apt-get update**

Then,

**apt-get install pdfcrack.**

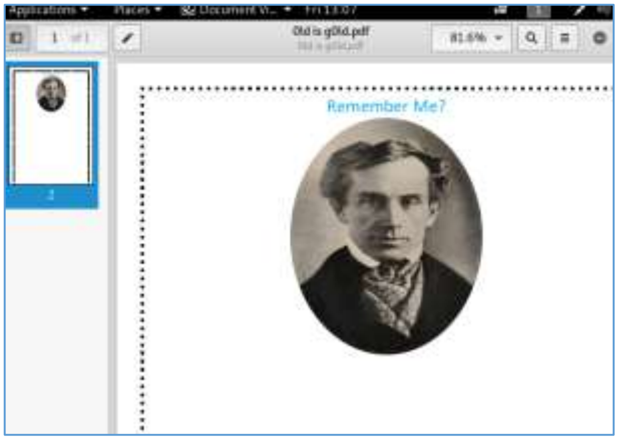
3)After that, we try using this command to brute force the pdf password with rockyou.txt in Kali.

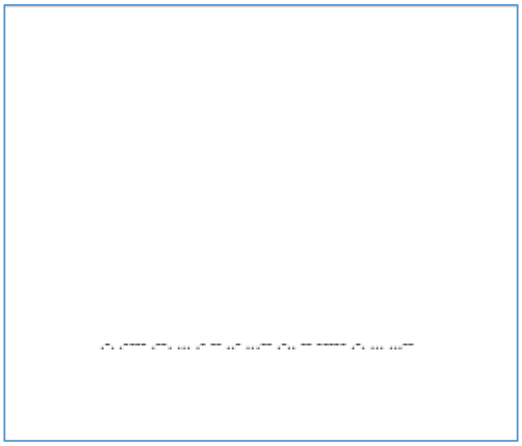
**pdfcrack –f “Old is g0ld.pdf” –w rockyou.txt**



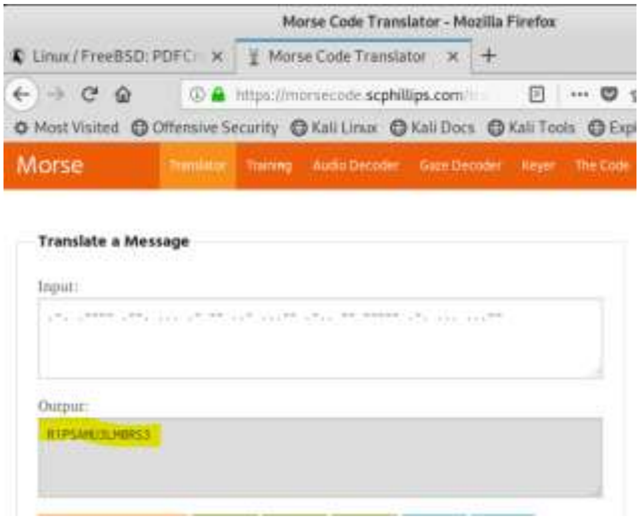
```
Error: file rockyou.txt not found
root@kali:~# pdfcrack -f "Old is g0ld.pdf" -w rockyou.txt

PDF version 1.6
Security Handler: Standard
V: 2
R: 3
P: -1060
Length: 128
Encrypted Metadata: True
FileID: 5c8f37d2a45eb64e9dbbf71ca3e86861
U: 9cba5cfb1c536f1384bba7458aae3f8100000000000000000000000000000000
O: 702cc7ced92b595274b7918dcb6dc74bedef6ef851b4b4b5b8c88732ba4dac0c
Average Speed: 41500.9 w/s. Current Word: 'passport31'
Average Speed: 42653.3 w/s. Current Word: 'jck463'
Average Speed: 42668.6 w/s. Current Word: 'zachel123'
Average Speed: 42523.2 w/s. Current Word: 'tam87bizzle'
Average Speed: 42650.5 w/s. Current Word: 'ritz32'
Average Speed: 42662.7 w/s. Current Word: 'nicogila'
Average Speed: 42468.9 w/s. Current Word: 'm1225534'
Average Speed: 42657.1 w/s. Current Word: 'jwinstonl'
found user-password: 'jumanji69'
```

4)We find a picture of a man and after zooming carefully , we find  some dashes and dots, which points to possible Morse code.





5)We translate it using Morse decoder online and found the flag.



Morse Code Translator - Mozilla Firefox

https://morsecode.scphillips.com/

Translate a Message

Input:

```
.-. .---- .--. ... .- -- ..- ...-- .-.. -- ----- .-. ... ...--
```

Output:

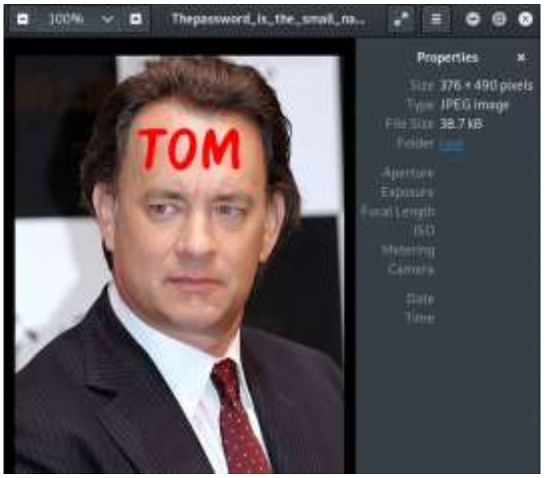
```
R1PSAMU3LM0RS3
```

Flag:HTB{RIPSAMU3LMORS3}

Stego challenges

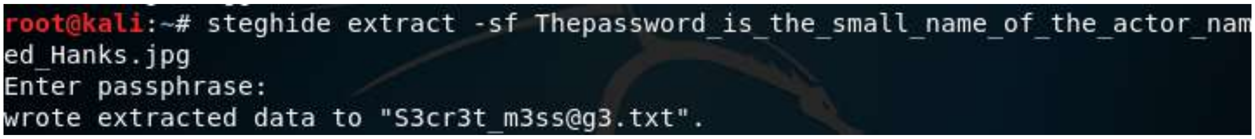
12.0 DaVinci

1)We have 3 files,Plans.jpg,monalisa.jpg and Thepassword_is_the_small_name_of_the_actor_named_Hanks.jpg.In the latter picture, Thepassword_is_the_small_name_of_the_actor_named_Hanks, we can deduce the password is 'TOM' from its name.



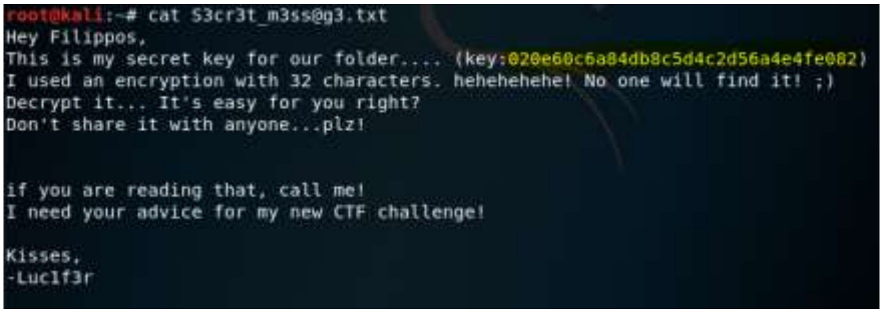
Size 376 × 490 pixels
Type JPEG image
File Size 38.7 kB
Folder
Aperture
Exposure
Focal Length
ISO
Metering
Camera
Date
Time

2)So, we run steghide to retrieve the hidden file using the password TOM.



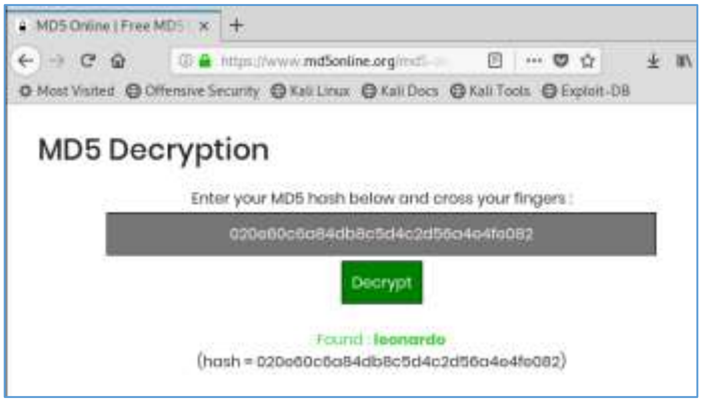
```
root@kali:~# steghide extract -sf Thepassword_is_the_small_name_of_the_actor_nam
ed_Hanks.jpg
Enter passphrase:
wrote extracted data to "S3cr3t_m3ss@g3.txt".
```

3)We obtained an MD5 hash.



```
root@kali:~# cat S3cr3t_m3ss@g3.txt
Hey Filippos,
This is my secret key for our folder.... (key:020e60c6a84db8c5d4c2d56a4e4fe082)
I used an encryption with 32 characters. hehehehehe! No one will find it! ;)
Decrypt it... It's easy for you right?
Don't share it with anyone...plz!


if you are reading that, call me!
I need your advice for my new CTF challenge!

Kisses,
-Luc1f3r
```

4)We decode and found the password.



MD5 Decryption

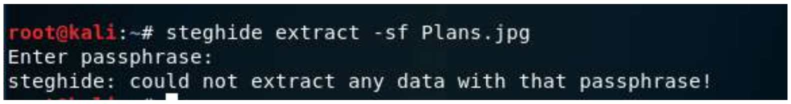
Enter your MD5 hash below and cross your fingers :

020e60c6a84db8c5d4c2d56a4e4fe082

Found : leonardo

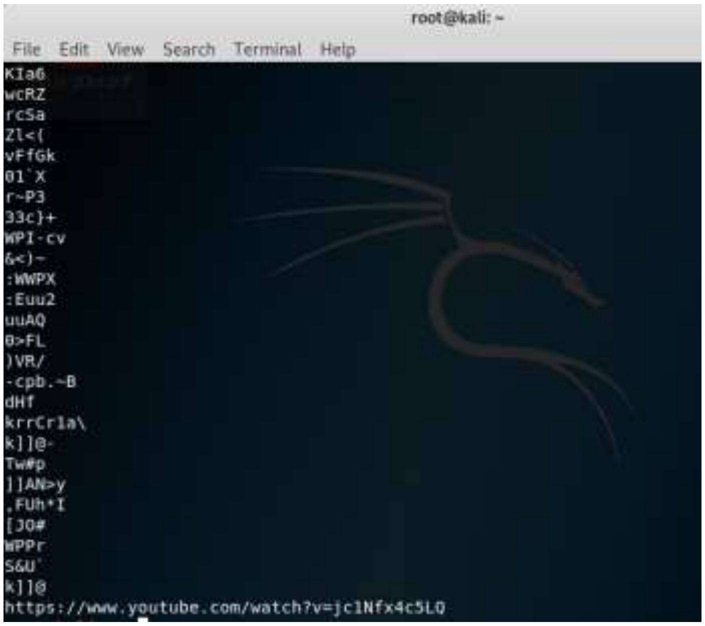
(hash = 020e60c6a84db8c5d4c2d56a4e4fe082)

5)The password 'leonardo' failed and could not extract the Plans.jpg file.



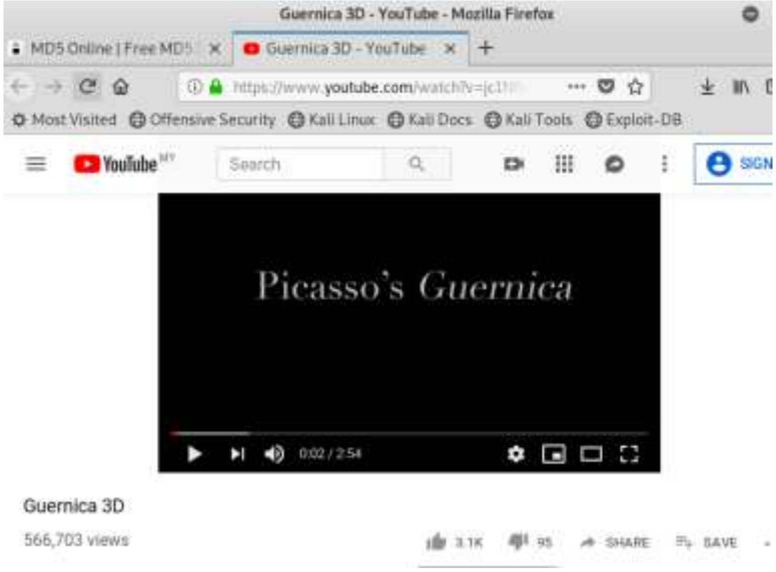
```
root@kali:~# steghide extract -sf Plans.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

6)We take a look at the strings using command **strings Plans.jpg** and found a youtube link.



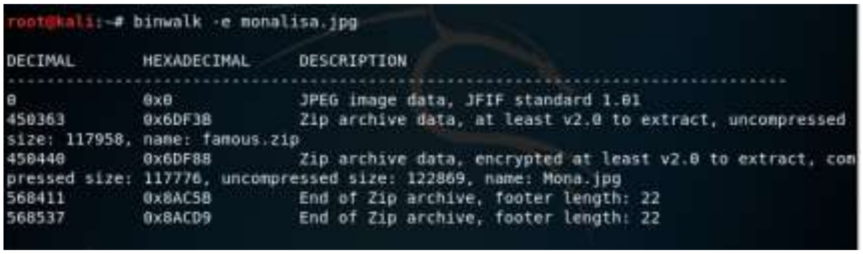
```
KIa6
wcRZ
rcSa
Zl<(
vFfGk
01`X
r~P3
33c}+
WPI-cv
&<)~
:WWPX
:Euu2
uuAQ
0>FL
)VR/
-cpb.~B
dHf
krrCr1a\
k]]@-
Tw#p
]]AN>y
,FUh*I
[JO#
WPPr
S&U`
k]]@
https://www.youtube.com/watch?v=jc1Nfx4c5LQ
```

7)We find that this leads to a youtube video titled ‘Guernica 3D’ .



8)We analyze the monalisa.jpg and extract the files inside it using

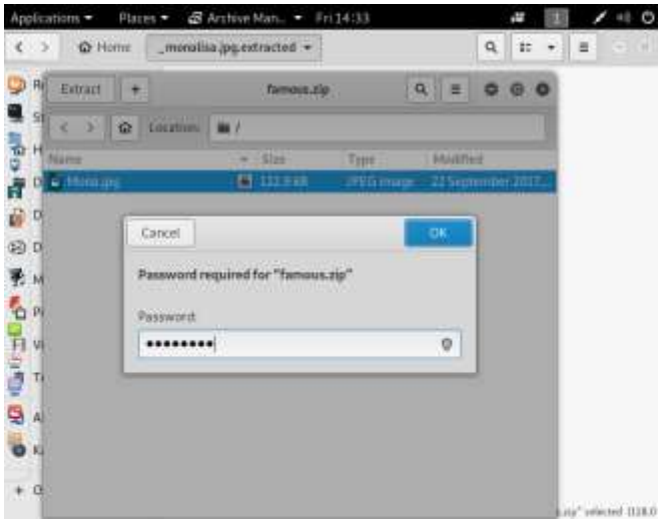
**binwalk –e monalisa.jpg**



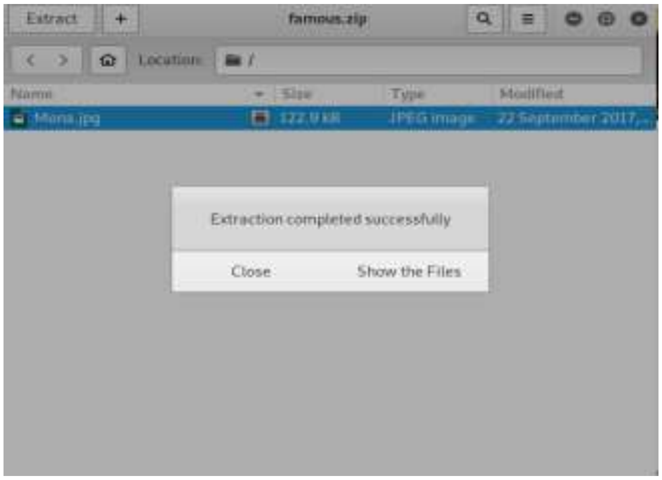
```
root@kali:~# binwalk -e monalisa.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
450363        0x6DF3B         Zip archive data, at least v2.0 to extract, uncompressed
size: 117958, name: famous.zip
450440        0x6DF88         Zip archive data, encrypted at least v2.0 to extract, com
pressed size: 117776, uncompressed size: 122869, name: Mona.jpg
568411        0x8AC5B         End of Zip archive, footer length: 22
568537        0x8ACD9         End of Zip archive, footer length: 22
```

9)There is file named **_monalisa.jpg.extracted** which contains a file named famous.zip.

We need to extract it. We try the password 'leonardo' and we succeeded in extracting the **Mona.jpg** file inside famous.zip.

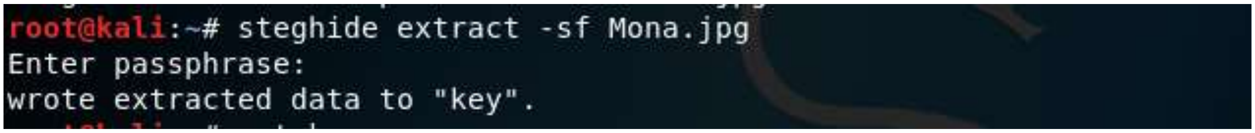


Password required for "famous.zip"



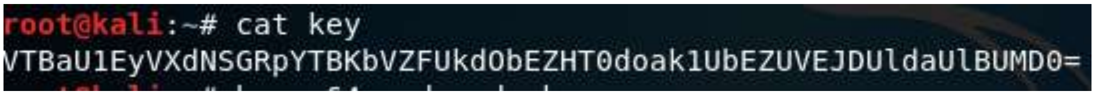
Extraction completed successfully

10)Then, we try to binwalk the extracted image 'Mona.jpg'. After trying passwords, we found that 'Guernica', the title of the Youtube video we obtained earlier, was the password.



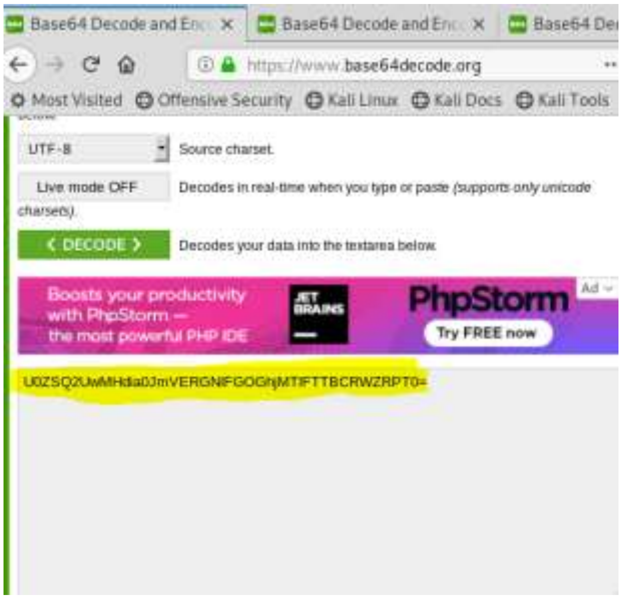
```
root@kali:~# steghide extract -sf Mona.jpg
Enter passphrase:
wrote extracted data to "key".
```

11)We obtained a base64 encoded text.



```
root@kali:~# cat key
VTBaU1EyVXdNSGRpYTBKbVZFUkdObEZHT0doak1UbEZUVEJDUldaUlBUMD0=
```

12)After decoding the text, we find another base 64 text. Maybe Guernica 3D, the ‘3D’ in the Youtube video title means 3 times decoding.



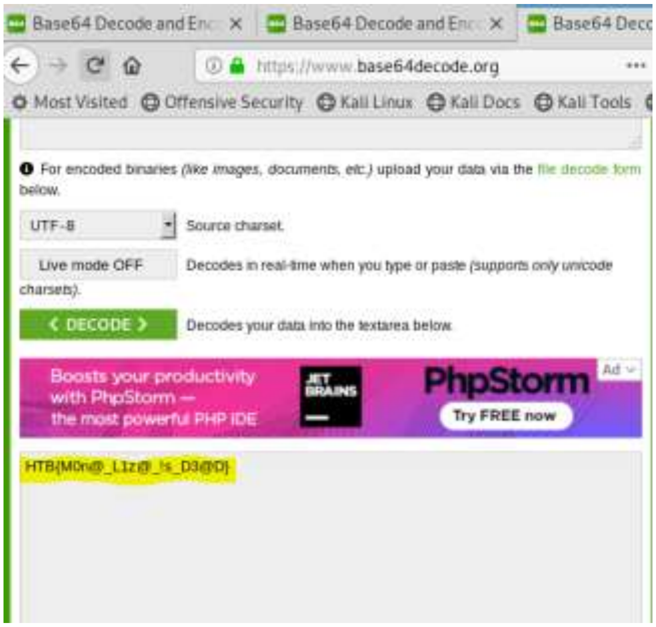
UTF-8 Source charset.

Live mode OFF Decodes in real-time when you type or paste (supports only unicode charsets).

DECODE Decodes your data into the textarea below.

```
U0ZSQ2UwMHdia0JmVERGNlFGOGhjMTlFTTBCRWZRPT0=
```

13)We repeat step 12 again 2 times with the decoded texts obtained and finally found the flag.



For encoded binaries (like images, documents, etc.) upload your data via the file decode form below.

UTF-8 Source charset.

Live mode OFF Decodes in real-time when you type or paste (supports only unicode charsets).

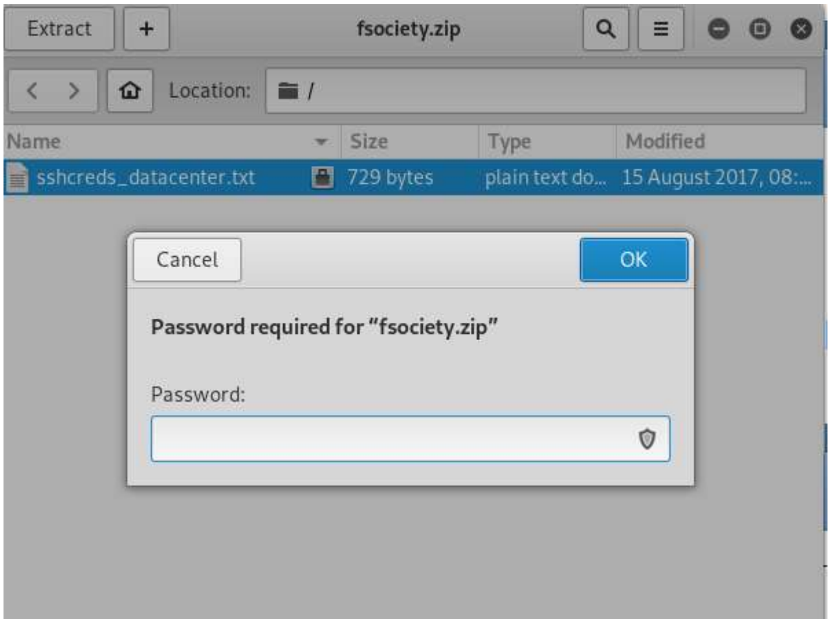
DECODE Decodes your data into the textarea below.

```
HTB{M0n@_L1z@_!s_D3@D}
```

Flag: HTB{Mon@_L1z@_!s_D3@D}

## 13.0 Fs0ciety

1)We have a fs0ciety.zip file. When we extract it , it has fsociety.zip inside. We need a password to extract this file.

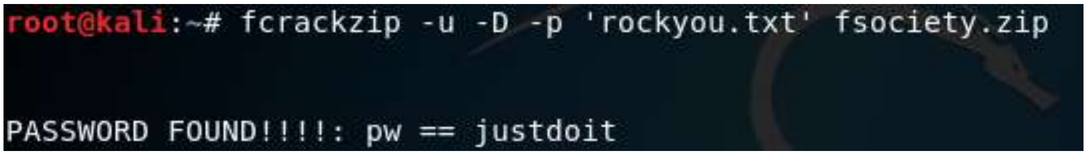


| Name | Size | Type | Modified |
|---|---|---|---|
| sshcreds_datacenter.txt | 729 bytes | plain text do... | 15 August 2017, 08:... |

Password required for "fsociety.zip"

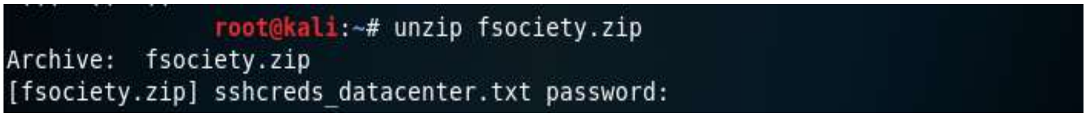
Password:

2)We use Fcrackzip in Kali Linux to brute force the password using rockyou.txt.



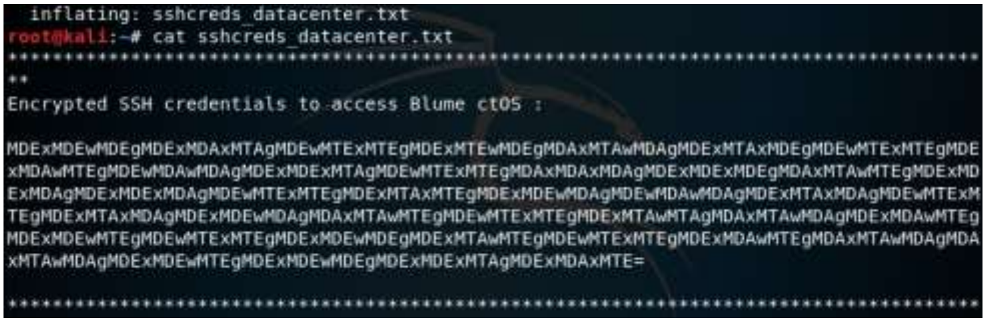
```
root@kali:~# fcrackzip -u -D -p 'rockyou.txt' fsociety.zip

PASSWORD FOUND!!!!: pw == justdoit
```

3)We unzip the file and enter the password obtained in step 2.



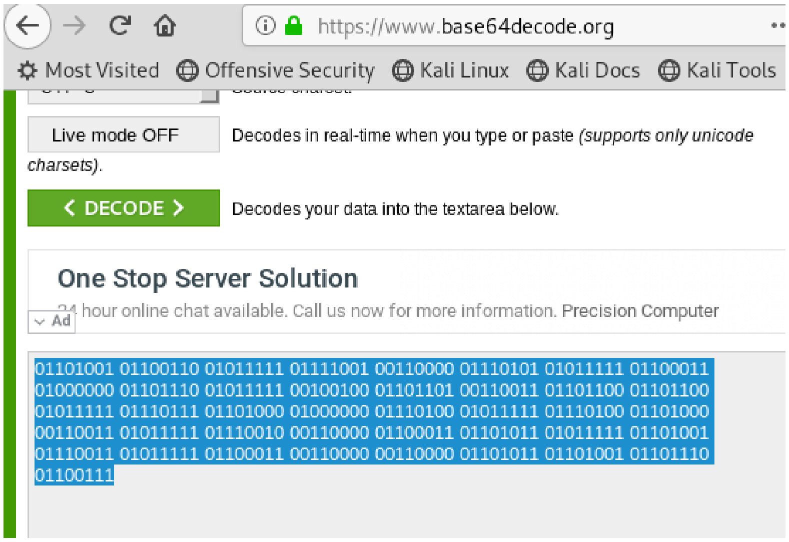
```
                root@kali:~# unzip fsociety.zip
Archive:  fsociety.zip
[fsociety.zip] sshcreds_datacenter.txt password:
```

4)Now, we see that inside is a .txt file containing base64 hash.



```
  inflating: sshcreds_datacenter.txt
root@kali:~# cat sshcreds_datacenter.txt
********************************************************************************
**
Encrypted SSH credentials to access Blume ctOS :

MDExMDEwMDEgMDExMDAxMTAgMDEwMTExMTEgMDExMTEwMDEgMDAxMTAwMDAgMDExMTAxMDEgMDEwMTExMTEgMDE
xMDAwMTEgMDEwMDAwMDAgMDExMDExMTAgMDEwMTExMTEgMDAxMDAxMDAgMDExMDExMDEgMDAxMTAwMTEgMDExMD
ExMDAgMDExMDExMDAgMDEwMTExMTEgMDExMTAxMTEgMDExMDEwMDAgMDEwMDAwMDAgMDExMTAxMDAgMDEwMTExM
TEgMDExMTAxMDAgMDExMDEwMDAgMDAxMTAwMTEgMDEwMTExMTEgMDExMTAwMTAgMDAxMTAwMDAgMDExMDAwMTEg
MDExMDEwMTEgMDEwMTExMTEgMDExMDEwMDEgMDExMTAwMTEgMDEwMTExMTEgMDExMDAwMTEgMDAxMTAwMDAgMDA
xMTAwMDAgMDExMDEwMTEgMDExMDEwMDEgMDExMDExMTAgMDExMDAxMTE=

********************************************************************************
```

5)We use base64 decoder and obtained what seems like binary.

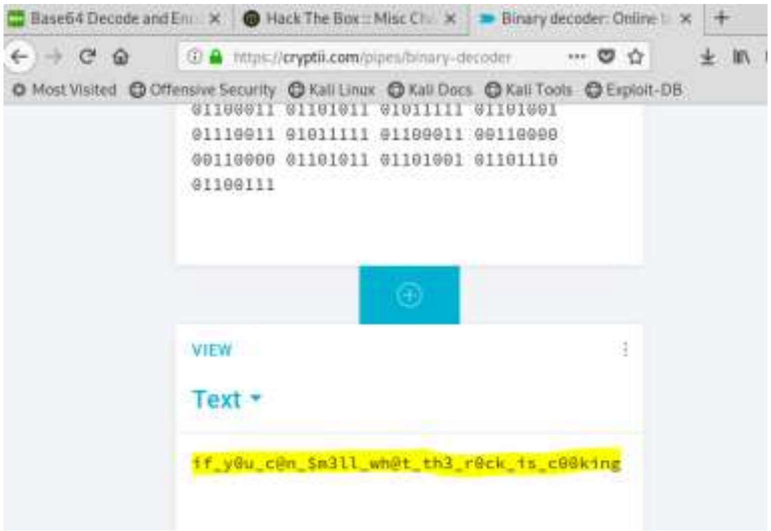


Decodes in real-time when you type or paste *(supports only unicode charsets).*

Decodes your data into the textarea below.

```
01101001 01100110 01011111 01111001 00110000 01110101 01011111 01100011
01000000 01101110 01011111 00100100 01101101 00110011 01101100 01101100
01011111 01110111 01101000 01000000 01110100 01011111 01110100 01101000
00110011 01011111 01110010 00110000 01100011 01101011 01011111 01101001
01110011 01011111 01100011 00110000 00110000 01101011 01101001 01101110
01100111
```

6)After using binary decoder, we finally found the flag.



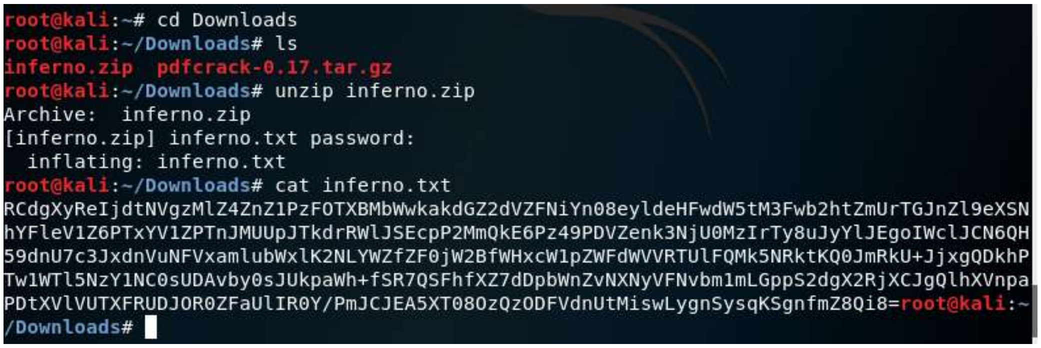
```
01100011 01101011 01011111 01101001
01110011 01011111 01100011 00110000
00110000 01101011 01101001 01101110
01100111
```

```
if_y0u_c@n_$m3ll_wh@t_th3_r0ck_is_c00king
```

Flag: HTB{if_y0u_$m3ll_wh@t_th3_r0ck_is_c00king}

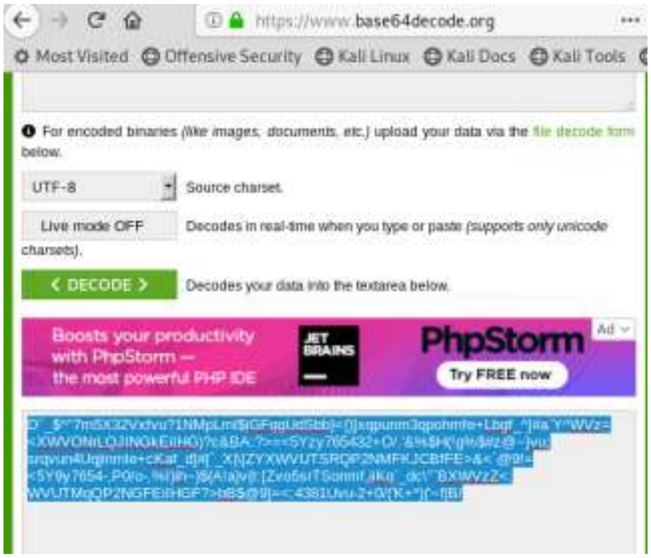
14.0 Inferno

1)After we unzip the file we get a base64 hash in a .txt file.



```
root@kali:~# cd Downloads
root@kali:~/Downloads# ls
inferno.zip  pdfcrack-0.17.tar.gz
root@kali:~/Downloads# unzip inferno.zip
Archive:  inferno.zip
[inferno.zip] inferno.txt password:
  inflating: inferno.txt
root@kali:~/Downloads# cat inferno.txt
RCdgXyReIjdtNVgzMlZ4ZnZ1PzFOTXBMbWwkakdGZ2dVZFNiYn08eyldeHFwdW5tM3Fwb2htZmUrTGJnZl9eXSN
hYFleV1Z6PTxYV1ZPTnJMUUpJTkdrRWlJSEcpP2MmQkE6Pz49PDVZenk3NjU0MzIrTy8uJyYlJEgoIWclJCN6QH
59dnU7c3JxdnVuNFVxamlubWxlK2NLYWZfZF0jW2BfWHxcW1pZWFdWVVRTUlFQMk5NRktKQ0JmRkU+JjxgQDkhP
Tw1WTl5NzY1NC0sUDAvby0sJUkpaWh+fSR7QSFhfXZ7dDpbWnZvNXNyVFNvbm1mLGppS2dgX2RjXCJgQlhXVnpa
PDtXVlVUTXFRUDJOR0ZFaUlIR0Y/PmJCJEA5XT08OzQzODFVdnUtMiswLygnSysqKSgnfmZ8Qi8=root@kali:~
/Downloads# 
```

2)So, we use base64 decoder and find some non-understandable text.



For encoded binaries (like images, documents, etc.) upload your data via the file decode form below.

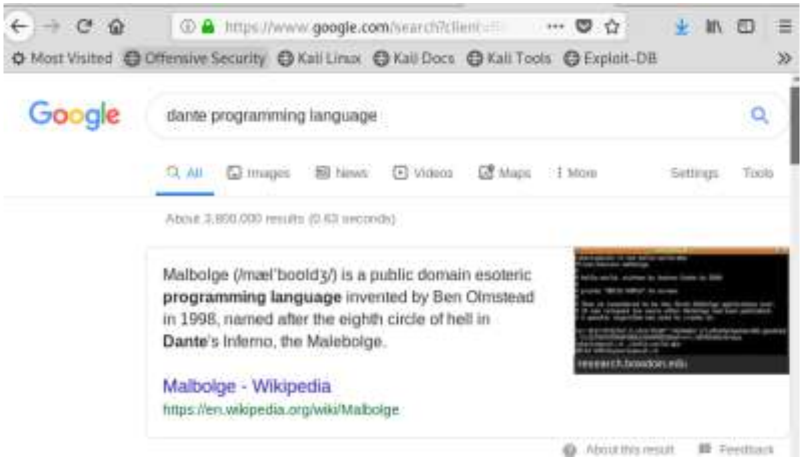
UTF-8 Source charset.

Live mode OFF Decodes in real-time when you type or paste (supports only unicode charsets).

Decodes your data into the textarea below.

```
D'`_$"'7m5X32Vxdvu?1NMpLml$jGFggUdSbb}=[]xqpunm3qpohmfe+Lbgf_^]#a`Y^WVz=
<XWVONrLQJINGkEIIHG)?c&BA:?>=<5Yzy765432+O/.'&%$H(`g%$#z@~}vu;
srqvun4Uqjnmle+cKaf_d]#[`_XN|ZYXWVUTSRQP2NMFKJCBfFE>&<`@9!=
<5Y9y7654-,P0/o-,%I)ih~}$(A!a}v@:[Zvo5srTSonmf,jiKg`_dc\"`BXWVzZ<
WVUTMqQP2NGFEiIHGF?>bB$@9]=<;4381Uvu-2+0/(K+*)('~fB/
```

3)The name of the challenge, ‘Inferno’ might hold a clue. What if it has anything to do with Dante’s inferno? We google ‘Dante programming language’ and find information about Malbolge, an esoteric programming language.



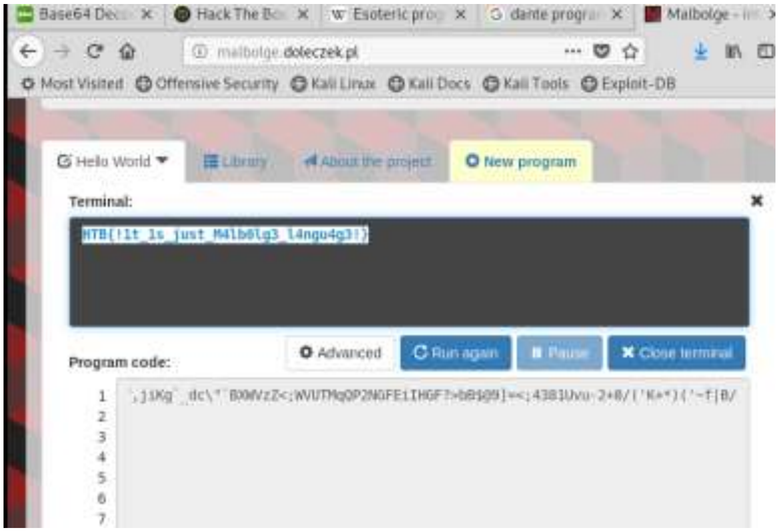
dante programming language

About 3,800,000 results (0.63 seconds)

Malbolge (/mælˈboʊldʒ/) is a public domain esoteric **programming language** invented by Ben Olmstead in 1998, named after the eighth circle of hell in **Dante**'s Inferno, the Malebolge.

Malbolge - Wikipedia
https://en.wikipedia.org/wiki/Malbolge

4)We use a Malbolge interpreter online and finally found the flag.



Terminal:

```
HTB{!1t_1s_just_M4lb0lg3_l4ngu4g3!}
```

Program code:

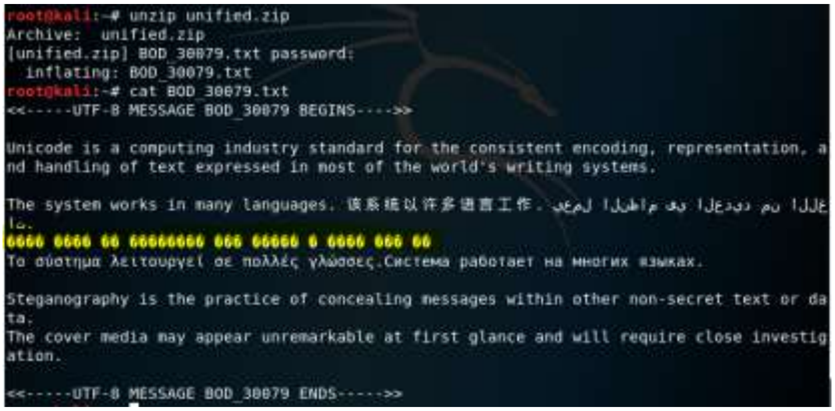
```
,jiKg`_dc\"`BXWVzZ<;WVUTMqQP2NGFEiIHGF?>bB$@9]=<;4381Uvu-2+0/('K+*)('~f|B/
```

Flag: HTB{!1t_1s_just_M4lb0lg3_l4ngu4g3!}

Stego challenge

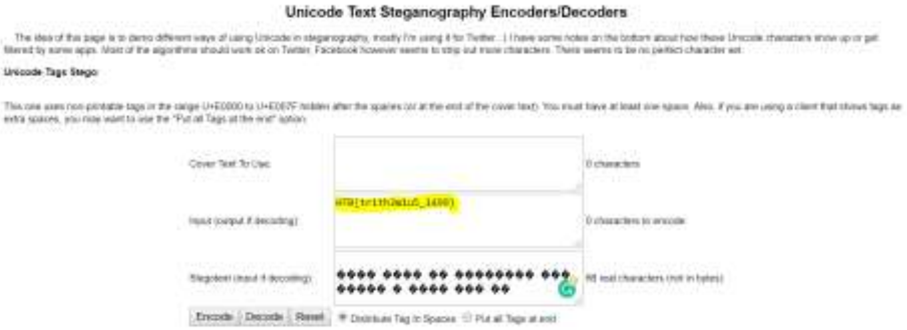
14.0  Unified

1)After unzipping the file and looking into the .txt file, we find that there are some unfamiliar characters.



```
root@kali:~# unzip unified.zip
Archive:  unified.zip
[unified.zip] BOD_30079.txt password:
  inflating: BOD_30079.txt
root@kali:~# cat BOD_30079.txt
<<-----UTF-8 MESSAGE BOD_30079 BEGINS---->>

Unicode is a computing industry standard for the consistent encoding, representation, a
nd handling of text expressed in most of the world's writing systems.

The system works in many languages. 该系统以许多语言工作. لمعي اماظنلا يف ديدعلا نم اتاغل
ل.
6666 6666 66 66666666 666 66666 6 6666 666 66
To σύστημα λειτουργεί σε πολλές γλώσσες.Система работает на многих языках.

Steganography is the practice of concealing messages within other non-secret text or da
ta.
The cover media may appear unremarkable at first glance and will require close investig
ation.

<<-----UTF-8 MESSAGE BOD_30079 ENDS----->>
```

2)The name of the challenge 'Unified', serves as  a clue. We found a Unicode text decoder. And try decoding the unfamiliar text. The flag is found.



Unicode Text Steganography Encoders/Decoders

The idea of this page is to demo different ways of using Unicode in steganography, mostly I'm using it for Twitter. :) I have some notes on the bottom about how these Unicode characters show up or get filtered by some apps. Most of the algorithms should work ok on Twitter, Facebook however seems to strip out more characters. There seems to be no perfect character set.

Unicode Tags Stego

This one uses non-printable tags in the range U+E0000 to U+E007F hidden after the spaces (or at the end of the cover text). You must have at least one space. Also, if you are using a client that shows tags as extra spaces, you may want to use the "Put all Tags at the end" option.

Cover Text To Use: 0 characters

Input (output if decoding): HTB{tr1th3m1u5_1499} 0 characters to encode

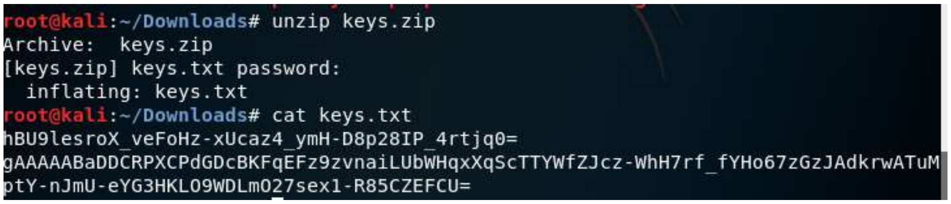
Stegotext (input if decoding): ◆◆◆◆ ◆◆◆◆ ◆◆ ◆◆◆◆◆◆◆◆ ◆◆◆ ◆◆◆◆◆ ◆ ◆◆◆◆ ◆◆◆ ◆◆ 98 real characters (not in bytes)

Flag: HTB{tr1th3m1u5_1499}

Crypto challenge

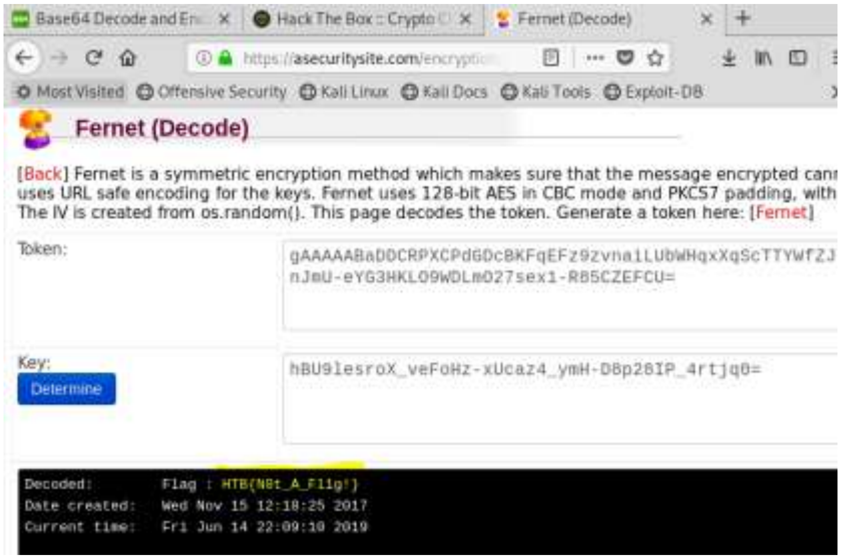
15.0 Keys

1.After unzipping the file, we found an encypted text in a .txt file.



```
root@kali:~/Downloads# unzip keys.zip
Archive:  keys.zip
[keys.zip] keys.txt password:
  inflating: keys.txt
root@kali:~/Downloads# cat keys.txt
hBU9lesroX_veFoHz-xUcaz4_ymH-D8p28IP_4rtjq0=
gAAAAABaDDCRPXCPdGDcBKFqEFz9zvnaiLUbWHqxXqScTTYWfZJcz-WhH7rf_fYHo67zGzJAdkrwATuM
ptY-nJmU-eYG3HKLO9WDLm027sex1-R85CZEFCU=
```

2)The text is a Fernet algorithm. Hence, we use a Ferner decoder and got the flag.



Fernet (Decode)

[Back] Fernet is a symmetric encryption method which makes sure that the message encrypted cann
uses URL safe encoding for the keys. Fernet uses 128-bit AES in CBC mode and PKCS7 padding, with
The IV is created from os.random(). This page decodes the token. Generate a token here: [Fernet]

Token: gAAAAABaDDCRPXCPdGDcBKFqEFz9zvnaiLUbWHqxXqScTTYWfZJ
nJmU-eYG3HKLO9WDLm027sex1-R85CZEFCU=

Key: hBU9lesroX_veFoHz-xUcaz4_ymH-D8p28IP_4rtjq0=

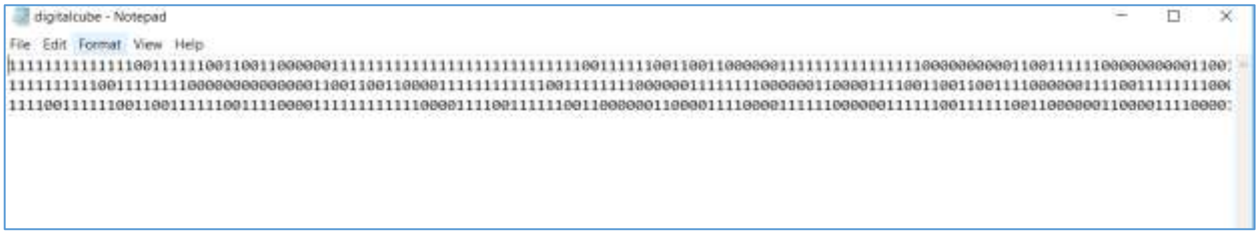
```
Decoded:        Flag : HTB{N0t_A_F l1g!}
Date created:   Wed Nov 15 12:18:25 2017
Current time:   Fri Jun 14 22:09:10 2019
```

Flag:HTB{N0t_A_Fl1g!}

Stego challenge

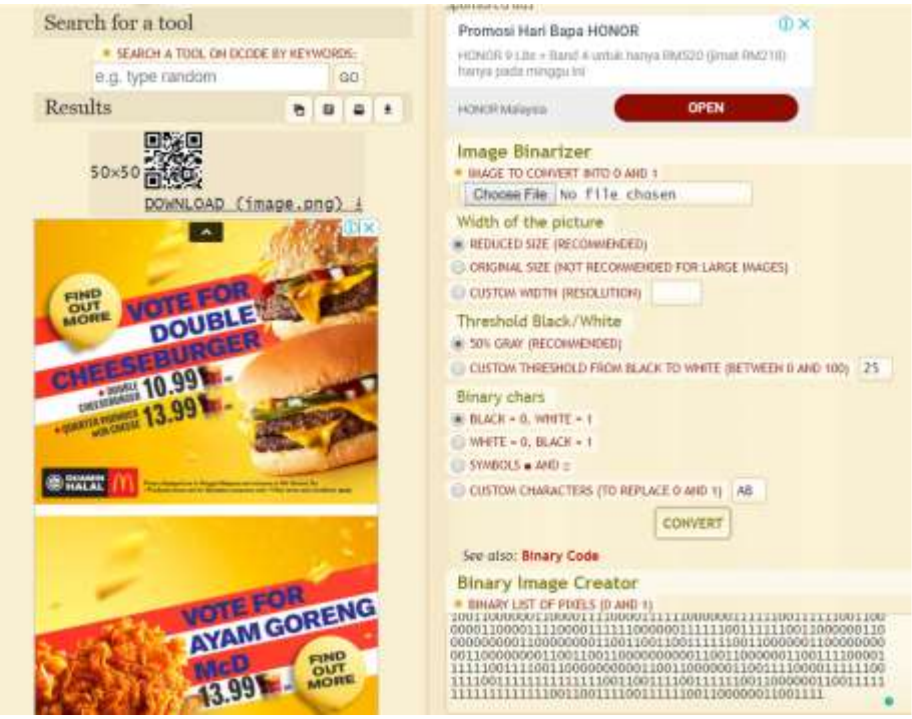
16.0 Digital Cube

1.After unzipping the file, we find a .txt file filled with binary



```
1111111111111001111110011001100000011111111111111111111111111111001111110011001100000011111111111111111000000000001100111111100000000000011001
1111111111001111111100000000000000011001100110000111111111111001111111100000011111111000000110000111100110011001111000000111100111111111100
1111001111110011001111110011110000011111111111100001110011111100110000001100001111000011111100000011111100111111001100000001100001111000001
```

2.We make the binary into image using binary image encoder in https://www.dcode.fr/binary-image.



Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type random GO

Results

50×50

DOWNLOAD (image.png)

Image Binarizer

IMAGE TO CONVERT INTO 0 AND 1

Choose File No file chosen

Width of the picture

- ● REDUCED SIZE (RECOMMENDED)
- ○ ORIGINAL SIZE (NOT RECOMMENDED FOR LARGE IMAGES)
- ○ CUSTOM WIDTH (RESOLUTION)

Threshold Black/White

- ● 50% GRAY (RECOMMENDED)
- ○ CUSTOM THRESHOLD FROM BLACK TO WHITE (BETWEEN 0 AND 100) 25

Binary chars

- ● BLACK = 0, WHITE = 1
- ○ WHITE = 0, BLACK = 1
- ○ SYMBOLS ■ AND □
- ○ CUSTOM CHARACTERS (TO REPLACE 0 AND 1) AB

CONVERT

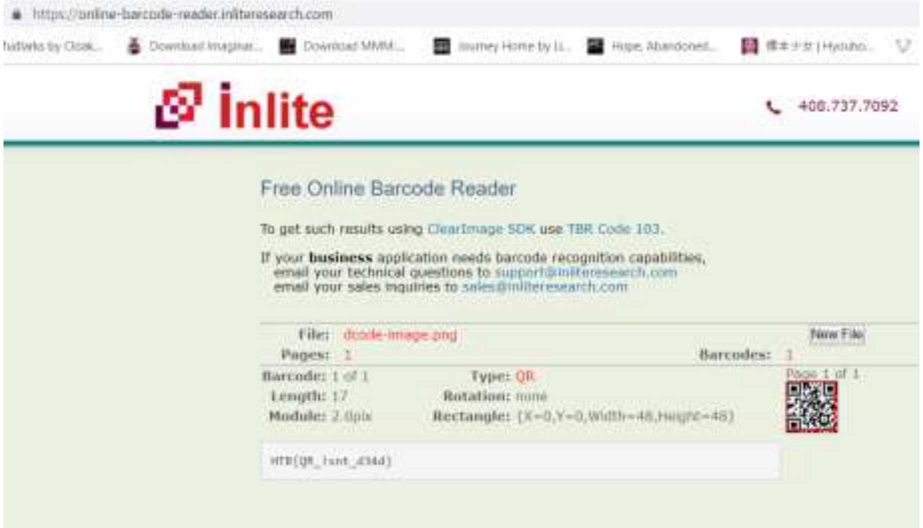
See also: Binary Code

Binary Image Creator

BINARY LIST OF PIXELS (0 AND 1)

```
000011000011110000111111000000111111001111110011000000110
0000000001100000000110011001100111111001100000011000000000
0011000000001100110011000000000001100110000000110011110000 1
111110011110011000000000001100110000000110011110000011111100
111100111111111111110011001111001111110011000000110011111
1111111111111100110011110011111100110000000110011111
```

3.Then, we upload the image into a QR code reader and we found the flag.



https://online-barcode-reader.inliteresearch.com

Free Online Barcode Reader

To get such results using ClearImage SDK use TBR Code 103.

If your **business** application needs barcode recognition capabilities,
email your technical questions to support@inliteresearch.com
email your sales inquiries to sales@inliteresearch.com

File: dcode-image.png
Pages: 1
Barcodes: 1
New File

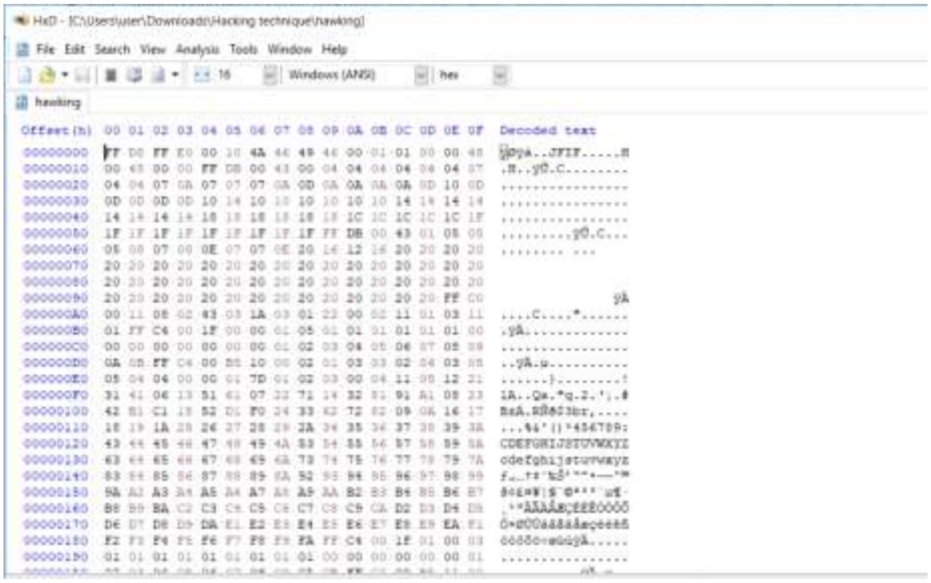
Barcode: 1 of 1
Length: 17
Module: 2.0pix
Type: QR
Rotation: none
Rectangle: {X=0,Y=0,Width=48,Height=48}
Page 1 of 1

```
HTB{QR_!snt_d34d}
```

Flag: HTB{QR_!snt_d34d}

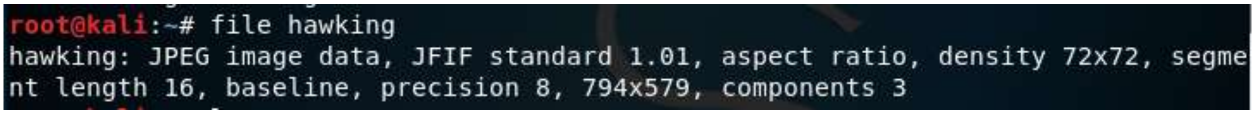
## 17.0 Blackhole

1.After unzipping the blackhole.zip, we find that it contains archive.zip. Inside  is a hawking file. When opened with HxD, it shows that the file might have relations with JFIF.


HxD - [C:\Users\user\Downloads\Hacking technique\hawking]
File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
hawking

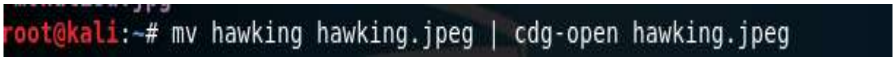
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 48  ÿØÿà..JFIF.....H
00000010  00 48 00 00 FF DB 00 43 00 04 04 04 04 04 04 07  .H..ÿÛ.C........
00000020  04 04 07 0A 07 07 07 0A 0D 0A 0A 0A 0A 0D 10 0D  ................
00000030  0D 0D 0D 0D 10 14 10 10 10 10 10 10 14 14 14 14  ................
00000040  14 14 14 14 18 18 18 18 18 18 1C 1C 1C 1C 1C 1F  ................
00000050  1F 1F 1F 1F 1F 1F 1F 1F 1F FF DB 00 43 01 05 05  .........ÿÛ.C...
00000060  05 08 07 08 0E 07 07 0E 20 16 12 16 20 20 20 20  ........ ...    
00000070  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  
00000080  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20                  
00000090  20 20 20 20 20 20 20 20 20 20 20 20 20 20 FF C0                ÿÀ
000000A0  00 11 08 02 43 03 1A 03 01 22 00 02 11 01 03 11  ....C...".......
000000B0  01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00  .ÿÄ.............
000000C0  00 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09  ................
000000D0  0A 0B FF C4 00 B5 10 00 02 01 03 03 02 04 03 05  ..ÿÄ.µ..........
000000E0  05 04 04 00 00 01 7D 01 02 03 00 04 11 05 12 21  .......}.......!
000000F0  31 41 06 13 51 61 07 22 71 14 32 81 91 A1 08 23  1A..Qa."q.2.'¡.#
00000100  42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17  B±Á.RÑð$3br.....
00000110  18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A  ...%&'()*456789:
00000120  43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A  CDEFGHIJSTUVWXYZ
00000130  63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A  cdefghijstuvwxyz
00000140  83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99  ƒ„…†‡ˆ‰Š’“”•–—˜™
00000150  9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7  š¢£¤¥¦§¨©ª²³´µ¶·
00000160  B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5  ¸¹ºÂÃÄÅÆÇÈÉÊÒÓÔÕ
00000170  D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1  Ö×ØÙÚáâãäåæçèéêñ
00000180  F2 F3 F4 F5 F6 F7 F8 F9 FA FF C4 00 1F 01 00 03  òóôõö÷øùúÿÄ.....
00000190  01 01 01 01 01 01 01 01 01 00 00 00 00 00 00 01  ................
```

2.We also find using ‘file’ command in Kali Linux that it can be jpeg file.

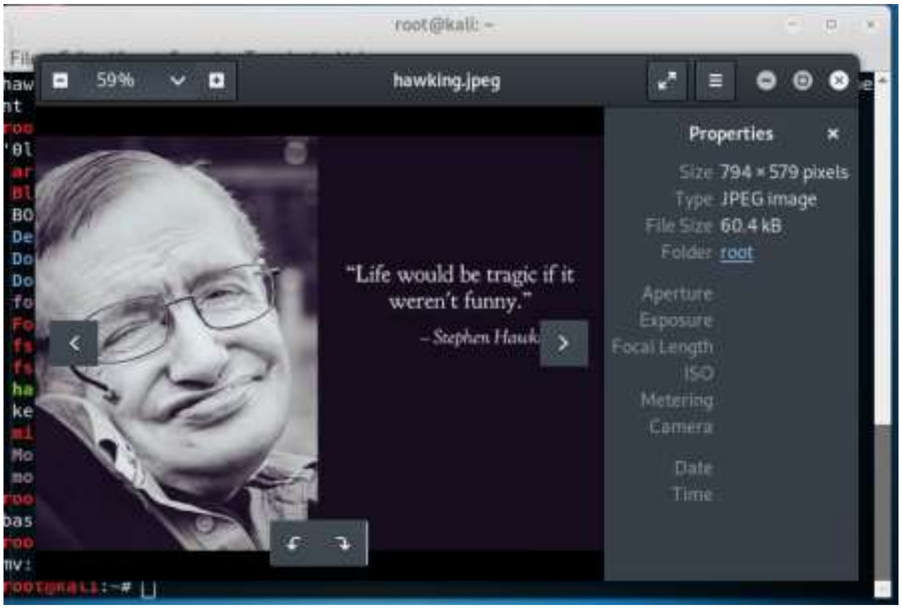


```
root@kali:~# file hawking
hawking: JPEG image data, JFIF standard 1.01, aspect ratio, density 72x72, segme
nt length 16, baseline, precision 8, 794x579, components 3
```

3.we change the name extension to .jpeg to see if it can open as .jpeg and it could.



```
root@kali:~# mv hawking hawking.jpeg | cdg-open hawking.jpeg
```



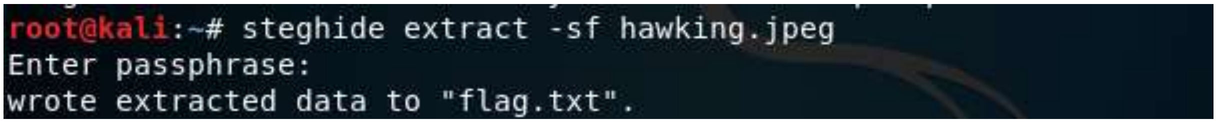
Properties

Size 794 × 579 pixels
Type JPEG image
File Size 60.4 kB
Folder root

Aperture
Exposure
Focal Length
ISO
Metering
Camera

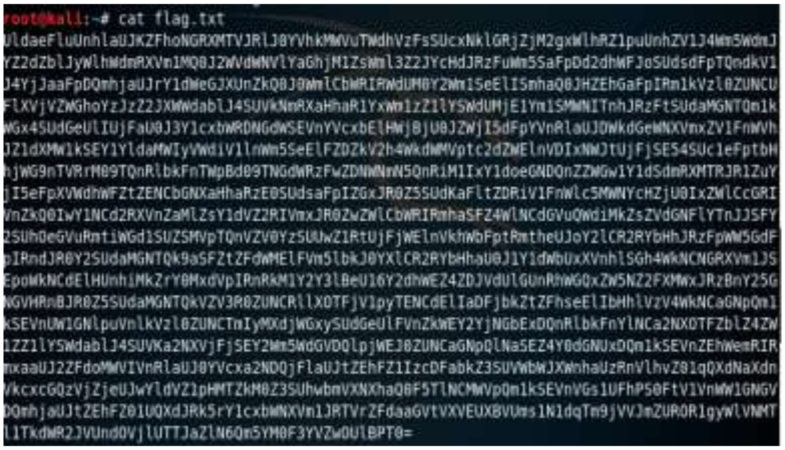
Date
Time

4.We use steghide to extract hidden files inside the .jpeg image in step3. We try different password and found 'hawking' as the passphrase.



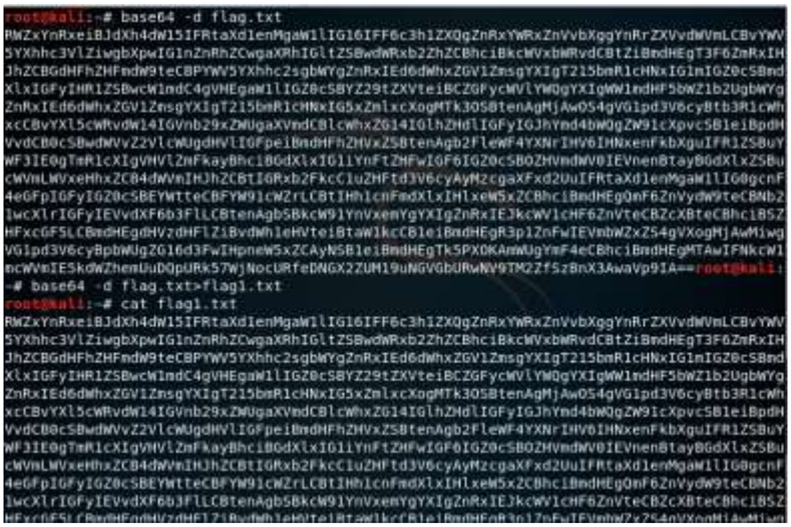
```
root@kali:~# steghide extract -sf hawking.jpeg
Enter passphrase:
wrote extracted data to "flag.txt".
```

5.We find encoded base64 in the flag.txt. We find that it is doubly encoded. After decoding it two times we found some text that is not base64.

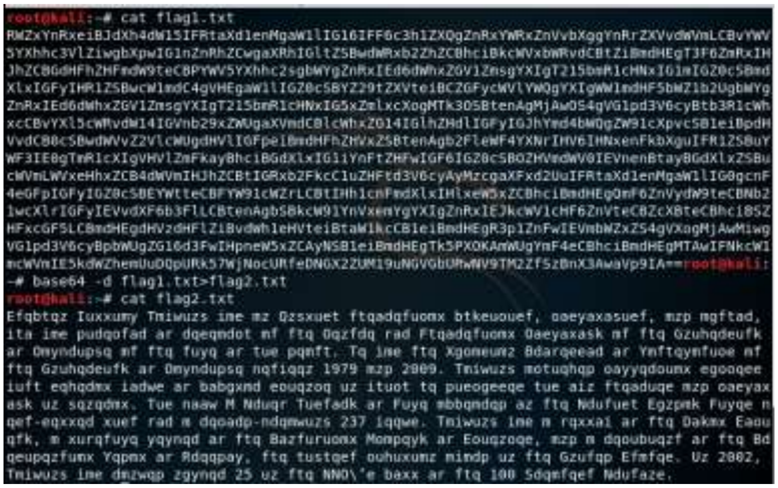


```
root@kali:~# cat flag.txt
```

UldaeFluUnhlaUJKZFhoNGRXMTVJRlJ0YVhkMWVuTWdhVzFsSUcxNklGRjZjM2gxWlhRZ1puUnhZV1J4Wm5WdmJ
YZ2dZblJyWlhWdmRXVm1MQ0J2WVdWNVlYaGhjM1ZsWml3Z2JYcHdJRzFuWm5SaFpDd2dhWFJoSUdsdFpTQndkV1
J4YjJaaFpDQmhjaUJrY1dWeGJXUnZkQ0J0WmlCbWRIRWdUM0Y2Wm1SeElISmhaQ0JHZEhGaFpIRm1kVzl0ZUNCU
FlXVjVZWGhoYzJzZ2JXWWdablJ4SUVkNmRXaHhaR1YxWm1zZ1lYSWdUMjE1Ym1SMWNITnhJRzFtSUdaMGNTQm1k
WGx4SUdGeUlIUjFaU0J3Y1cxbWRDNGdWSEVnYVcxbElHWjBjU0JZWjI5dFpYVnRlaUJ0WkdGeWNXVmxZV1FnWVh
JZ1dXMW1kSEY1YldaMWIyVWdiV1llnWm5SeElFZDZkV2h4WkdWMVptc2dZWElnVDIxNWJtUjFjSE54SUcleFptbH
hjWG9nTVRrM09TQnRlbkFnTWpBd09TNGdWRzFwZDNWNmN5QnRiM1IxY1doeGNDQnZZWGw1Y1dSdmRXMTRJR1ZuY
jI5eFpXVWdhWFZtZENCbGNXaHhaRzE0SUdsaFpIZGxJR0Z5SUdKaFltZDRiV1FnWlc5MWNYcHZjU0IxZWlCcGRI
VnZkQ0IwY1NCd2RXVnZaMlZsY1dVZ2RIVmxJR0ZwZWlCbWRIRmhaSFZ4WlNCdGVuQWdiMkZsZVdGNFlYTnJJSFY
ZSUhOeGVuRmtiWGd1SUZSMVpTQnVZV0YzSUUwZ1RtUjFjWElnVkhWbFptRmtheUJoY2lCR2RYbHhJRzFpWW5GdF
pIRndJR0Y2SUdaMGNTQk9aSFZtZFdWMElFVm5lbkJ0YXlCR2RYbHhaU0J1Y1dWbUxXVnhlSGh4WkNCNGRXVm1JS
EpoWkNCdElHUnhiMkZrY0MxdVpIRnRkM1Y2Y3lBeU16Y2dhWEZ4ZDJVdUlGUnRhWGQxZW5NZ2FXMWxJRzBnY25G
NGVHRnBJR0Z5SUdaMGNTQkVZV3R0ZUNCRllXOTFjV1pyTENCdElIaDFjbkZtZFhseElIbHhlVzV4WkNCaGNpQm1
kSEVnUW1GNlpuVnlkVzl0ZUNCTmIyMXdjWGxySUdGeUlFVnZkWEY2YjNGbExDQnRlbkFnYlNCa2NXOTFZblZ4ZW
1ZZ1lYSWdablJ4SUVKa2NXVjFjSEY2Wm5WdGVDQlpjWEJ0ZUNCaGNpQlNaSEZ4Y0dGNUxDQm1kSEVnZEhWemRIR
mxaaUJ2ZFdoMWVIVnRlaUJ0YVcxa2NDQjFlaUJtZEhFZ1IzcDFabkZ3SUVWbWJXWnhaUzRnVlhvZ01qQXdNaXdn
VkcxcGQzVjZjeUJwYldVZ1pHMTZkM0Z3SUhwbmVXNXhaQ0F5TlNCMWVpQm1kSEVnVGs1UFhPS0FtV1VnWW1GNGV
DQmhjaUJtZEhFZ01UQXdJRk5rY1cxbWNXVm1JRTVrZFdaaGVtVXVEUXBVUms1N1dqTm9jVVJmZUROR1gyWlVNMT
l1TkdWR2JVUndOVjlUTTJaZlN6Qm5YM0F3YVZwd0UlBPT0=

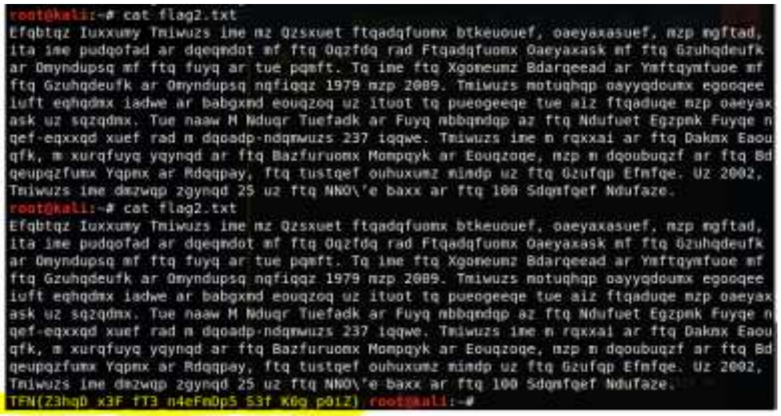


```
root@kali:~# base64 -d flag.txt
RWZxYnRxeiBJdXh4dW15IFRtaXd1enMgaW1lIG16IFF6c3h1ZXQgZnRxYWRxZnVvbXggYnRrZXVvdWVmLCBvYWV
5YXhhc3VlZiwgbXpwIG1nZnRhZCwgaXRhIGltZSBwdWRxb2ZhZCBhciBkcWVxbWRvdCBtZiBmdHEgT3F6ZmRxIH
JhZCBGdHFhZHFmdW9teCBPYWV5YXhhc2sgbWYgZnRxIEd6dWhxZGV1ZmsgYXIgT215bmR1cHNxIG1mIGZ0cSBmd
XlxIGFyIHR1ZSBwcW1mdC4gVHEgaW1lIGZ0cSBYZ29tZXVteiBCZGFycWVlYWQgYXIgWW1mdHF5bWZ1b2UgbWYg
ZnRxIEd6dWhxZGV1ZmsgYXIgT215bmR1cHNxIG5xZmlxcXogMTk3OSBtenAgMjAwOS4gVG1pd3V6cyBtb3R1cWh
xcCBvYXl5cWRvdW14IGVnb29xZWUgaXVmdCBlcWhxZG14IGlhZHdlIGFyIGJhYmd4bWQgZW91cXpvcSB1eiBpdH
VvdCB0cSBwdWVvZ2VlcWUgdHVlIGFpeiBmdHFhZHVxZSBtenAgb2FleWF4YXNrIHV6IHNxenFkbXguIFR1ZSBuY
WF3IE0gTmR1cXIgVHVlZmFkayBhciBGdXlxIG1iYnFtZHFwIGF6IGZ0cSBOZHVmdWV0IEVnenBtayBGdXlxZSBu
cWVmLWVxeHhxZCB4dWVmIHJhZCBtIGRxb2FkcC1uZHFtd3V6cyAyMzcgaXFxd2UuIFRtaXd1enMgaW1lIG0gcnF
4eGFpIGFyIGZ0cSBEYWtteCBFYW91cWZrLCBtIHh1cnFmdXlxIHlxeW5xZCBhciBmdHEgQmF6ZnVydW9teCBNb2
1wcXlrIGFyIEVvdXF6b3FlLCBtenAgbSBkcW91YnVxemYgYXIgZnRxIEJkcWV1cHF6ZnVteCBZcXBteCBhciBSZ
HFxcGF5LCBmdHEgdHVzdHFlZiBvdWh1eHVteiBtaW1kcCB1eiBmdHEgR3p1ZnFwIEVmbWZxZS4gVXogMjAwMiwg
VG1pd3V6cyBpbWUgZG16d3FwIHpneW5xZCAyNSB1eiBmdHEgTk5PXOKAmWUgYmF4eCBhciBmdHEgMTAwIFNkcW1
mcWVmIE5kdWZhemUuDQpURk57WjNocURfeDNGX2ZUM19uNGVGbURwNV9TM2ZfSzBnX3AwaVp9IA==root@kali:
~# base64 -d flag.txt>flag1.txt
root@kali:~# cat flag1.txt
RWZxYnRxeiBJdXh4dW15IFRtaXd1enMgaW1lIG16IFF6c3h1ZXQgZnRxYWRxZnVvbXggYnRrZXVvdWVmLCBvYWV
5YXhhc3VlZiwgbXpwIG1nZnRhZCwgaXRhIGltZSBwdWRxb2ZhZCBhciBkcWVxbWRvdCBtZiBmdHEgT3F6ZmRxIH
JhZCBGdHFhZHFmdW9teCBPYWV5YXhhc2sgbWYgZnRxIEd6dWhxZGV1ZmsgYXIgT215bmR1cHNxIG1mIGZ0cSBmd
XlxIGFyIHR1ZSBwcW1mdC4gVHEgaW1lIGZ0cSBYZ29tZXVteiBCZGFycWVlYWQgYXIgWW1mdHF5bWZ1b2UgbWYg
ZnRxIEd6dWhxZGV1ZmsgYXIgT215bmR1cHNxIG5xZmlxcXogMTk3OSBtenAgMjAwOS4gVG1pd3V6cyBtb3R1cWh
xcCBvYXl5cWRvdW14IGVnb29xZWUgaXVmdCBlcWhxZG14IGlhZHdlIGFyIGJhYmd4bWQgZW91cXpvcSB1eiBpdH
VvdCB0cSBwdWVvZ2VlcWUgdHVlIGFpeiBmdHFhZHVxZSBtenAgb2FleWF4YXNrIHV6IHNxenFkbXguIFR1ZSBuY
WF3IE0gTmR1cXIgVHVlZmFkayBhciBGdXlxIG1iYnFtZHFwIGF6IGZ0cSBOZHVmdWV0IEVnenBtayBGdXlxZSBu
cWVmLWVxeHhxZCB4dWVmIHJhZCBtIGRxb2FkcC1uZHFtd3V6cyAyMzcgaXFxd2UuIFRtaXd1enMgaW1lIG0gcnF
4eGFpIGFyIGZ0cSBEYWtteCBFYW91cWZrLCBtIHh1cnFmdXlxIHlxeW5xZCBhciBmdHEgQmF6ZnVydW9teCBNb2
1wcXlrIGFyIEVvdXF6b3FlLCBtenAgbSBkcW91YnVxemYgYXIgZnRxIEJkcWV1cHF6ZnVteCBZcXBteCBhciBSZ
HFxcGF5LCBmdHEgdHVzdHFlZiBvdWh1eHVteiBtaW1kcCB1eiBmdHEgR3p1ZnFwIEVmbWZxZS4gVXogMjAwMiwg
```



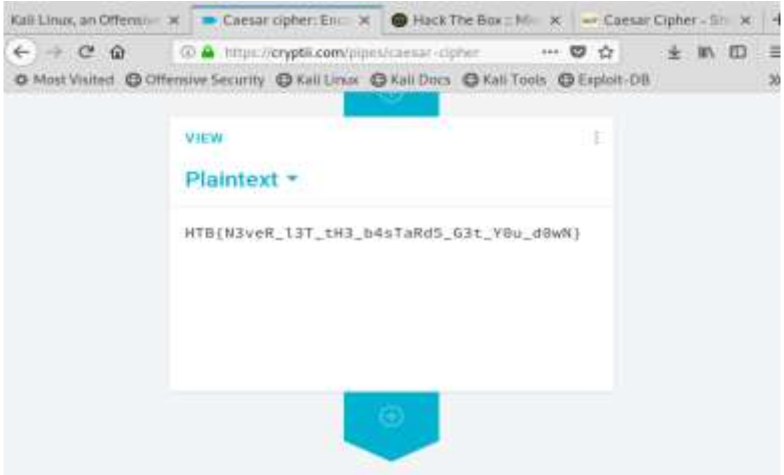
```
root@kali:~# cat flag1.txt
RWZxYnRxeiBJdXh4dW15IFRtaXd1enMgaW1lIG16IFF6c3h1ZXQgZnRxYWRxZnVvbXggYnRrZXVvdWVmLCBvYWV
5YXhhc3VlZiwgbXpwIG1nZnRhZCwgaXRhIGltZSBwdWRxb2ZhZCBhciBkcWVxbWRvdCBtZiBmdHEgT3F6ZmRxIH
JhZCBGdHFhZHFmdW9teCBPYWV5YXhhc2sgbWYgZnRxIEd6dWhxZGV1ZmsgYXIgT215bmR1cHNxIG1mIGZ0cSBmd
XlxIGFyIHR1ZSBwcW1mdC4gVHEgaW1lIGZ0cSBYZ29tZXVteiBCZGFycWVlYWQgYXIgWW1mdHF5bWZ1b2UgbWYg
ZnRxIEd6dWhxZGV1ZmsgYXIgT215bmR1cHNxIG5xZmlxcXogMTk3OSBtenAgMjAwOS4gVG1pd3V6cyBtb3R1cWh
xcCBvYXl5cWRvdW14IGVnb29xZWUgaXVmdCBlcWhxZG14IGlhZHdlIGFyIGJhYmd4bWQgZW91cXpvcSB1eiBpdH
VvdCB0cSBwdWVvZ2VlcWUgdHVlIGFpeiBmdHFhZHVxZSBtenAgb2FleWF4YXNrIHV6IHNxenFkbXguIFR1ZSBuY
WF3IE0gTmR1cXIgVHVlZmFkayBhciBGdXlxIG1iYnFtZHFwIGF6IGZ0cSBOZHVmdWV0IEVnenBtayBGdXlxZSBu
cWVmLWVxeHhxZCB4dWVmIHJhZCBtIGRxb2FkcC1uZHFtd3V6cyAyMzcgaXFxd2UuIFRtaXd1enMgaW1lIG0gcnF
4eGFpIGFyIGZ0cSBEYWtteCBFYW91cWZrLCBtIHh1cnFmdXlxIHlxeW5xZCBhciBmdHEgQmF6ZnVydW9teCBNb2
1wcXlrIGFyIEVvdXF6b3FlLCBtenAgbSBkcW91YnVxemYgYXIgZnRxIEJkcWV1cHF6ZnVteCBZcXBteCBhciBSZ
HFxcGF5LCBmdHEgdHVzdHFlZiBvdWh1eHVteiBtaW1kcCB1eiBmdHEgR3p1ZnFwIEVmbWZxZS4gVXogMjAwMiwg
VG1pd3V6cyBpbWUgZG16d3FwIHpneW5xZCAyNSB1eiBmdHEgTk5PXOKAmWUgYmF4eCBhciBmdHEgMTAwIFNkcW1
mcWVmIE5kdWZhemUuDQpURk57WjNocURfeDNGX2ZUM19uNGVGbURwNV9TM2ZfSzBnX3AwaVp9IA==root@kali:
~# base64 -d flag1.txt>flag2.txt
root@kali:~# cat flag2.txt
Efqbtqz Iuxxumy Tmiwuzs ime mz Qzsxuet ftqadqfuomx btkeuouef, oaeyaxasuef, mzp mgftad,
ita ime pudqofad ar dqeqmdot mf ftq Oqzfdq rad Ftqadqfuomx Oaeyaxask mf ftq Gzuhqdeufk
ar Omyndupsq mf ftq fuyq ar tue pqmft. Tq ime ftq Xgomeumz Bdarqeead ar Ymftqymfuoe mf
ftq Gzuhqdeufk ar Omyndupsq nqfiqqz 1979 mzp 2009. Tmiwuzs motuqhqp oayyqdoumx egooqee
iuft eqhqdmx iadwe ar babgxmd eouqzoq uz ituot tq pueogeeqe tue aiz ftqaduqe mzp oaeyax
ask uz sqzqdmx. Tue naaw M Nduqr Tuefadk ar Fuyq mbbqmdqp az ftq Ndufuet Egzpmk Fuyqe n
qef-eqxxqd xuef rad m dqoadp-ndqmwuzs 237 iqqwe. Tmiwuzs ime m rqxxai ar ftq Dakmx Eaou
qfk, m xurqfuyq yqynqd ar ftq Bazfuruomx Mompqyk ar Eouqzoqe, mzp m dqoubuqzf ar ftq Bd
qeupqzfumx Yqpmx ar Rdqqpay, ftq tustqef ouhuxumz mimdp uz ftq Gzufqp Efmfqe. Uz 2002,
Tmiwuzs ime dmzwqp zgynqd 25 uz ftq NNO\'e baxx ar ftq 100 Sdqmfqef Ndufaze.
```

6)We find something that could relate to the flag at the very bottom of the encrypted text.



```
root@kali:~# cat flag2.txt
Efqbtqz Iuxxumy Tmiwuzs ime mz Qzsxuet ftqadqfuomx btkeuouef, oaeyaxasuef, mzp mgftad,
ita ime pudqofad ar dqeqmdot mf ftq Oqzfdq rad Ftqadqfuomx Oaeyaxask mf ftq Gzuhqdeufk
ar Omyndupsq mf ftq fuyq ar tue pqmft. Tq ime ftq Xgomeumz Bdarqeead ar Ymftqymfuoe mf
ftq Gzuhqdeufk ar Omyndupsq nqfiqqz 1979 mzp 2009. Tmiwuzs motuqhqp oayyqdoumx egooqee
iuft eqhqdmx iadwe ar babgxmd eouqzoq uz ituot tq pueogeeqe tue aiz ftqaduqe mzp oaeyax
ask uz sqzqdmx. Tue naaw M Nduqr Tuefadk ar Fuyq mbbqmdqp az ftq Ndufuet Egzpmk Fuyqe n
qef-eqxxqd xuef rad m dqoadp-ndqmwuzs 237 iqqwe. Tmiwuzs ime m rqxxai ar ftq Dakmx Eaou
qfk, m xurqfuyq yqynqd ar ftq Bazfuruomx Mompqyk ar Eouqzoqe, mzp m dqoubuqzf ar ftq Bd
qeupqzfumx Yqpmx ar Rdqqpay, ftq tustqef ouhuxumz mimdp uz ftq Gzufqp Efmfqe. Uz 2002,
Tmiwuzs ime dmzwqp zgynqd 25 uz ftq NNO\'e baxx ar ftq 100 Sdqmfqef Ndufaze.
root@kali:~# cat flag2.txt
Efqbtqz Iuxxumy Tmiwuzs ime mz Qzsxuet ftqadqfuomx btkeuouef, oaeyaxasuef, mzp mgftad,
ita ime pudqofad ar dqeqmdot mf ftq Oqzfdq rad Ftqadqfuomx Oaeyaxask mf ftq Gzuhqdeufk
ar Omyndupsq mf ftq fuyq ar tue pqmft. Tq ime ftq Xgomeumz Bdarqeead ar Ymftqymfuoe mf
ftq Gzuhqdeufk ar Omyndupsq nqfiqqz 1979 mzp 2009. Tmiwuzs motuqhqp oayyqdoumx egooqee
iuft eqhqdmx iadwe ar babgxmd eouqzoq uz ituot tq pueogeeqe tue aiz ftqaduqe mzp oaeyax
ask uz sqzqdmx. Tue naaw M Nduqr Tuefadk ar Fuyq mbbqmdqp az ftq Ndufuet Egzpmk Fuyqe n
qef-eqxxqd xuef rad m dqoadp-ndqmwuzs 237 iqqwe. Tmiwuzs ime m rqxxai ar ftq Dakmx Eaou
qfk, m xurqfuyq yqynqd ar ftq Bazfuruomx Mompqyk ar Eouqzoqe, mzp m dqoubuqzf ar ftq Bd
qeupqzfumx Yqpmx ar Rdqqpay, ftq tustqef ouhuxumz mimdp uz ftq Gzufqp Efmfqe. Uz 2002,
Tmiwuzs ime dmzwqp zgynqd 25 uz ftq NNO\'e baxx ar ftq 100 Sdqmfqef Ndufaze.
TFN{Z3hqD_x3F_fT3_n4eFmDp5_S3f_K0g_p0iZ} root@kali:~#
```

7)We decode the text starting from “TFN{..”  as highlighted in step6 using Caesar cipher decoder and find the flag.

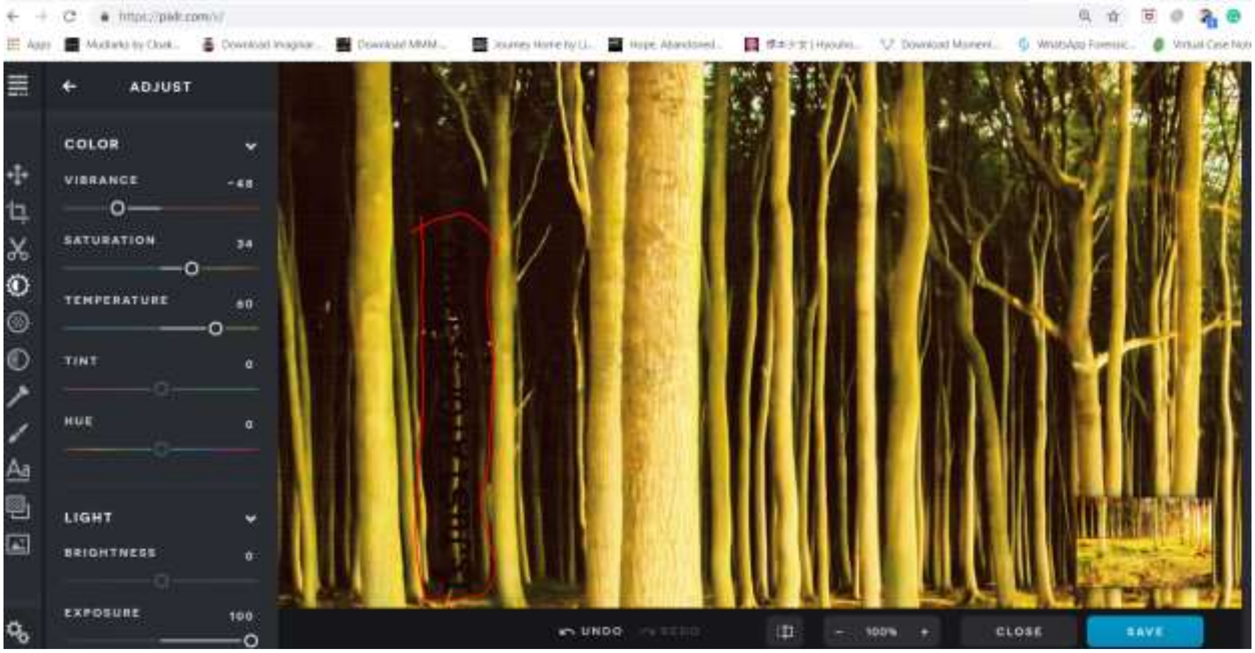


```
HTB{N3veR_l3T_tH3_b4sTaRd5_G3t_Y0u_d0wN}
```

Flag: HTB{N3veR_l3T_tH3_b4sTaRd5_G3t_Y0u_d0wN}

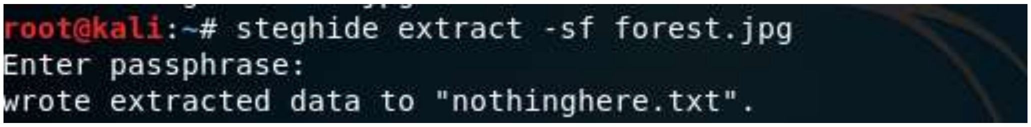
## 18.0 Forest

1.We try to find hidden things inside the photo by using Pixlr online photo editor. We adjust the brightness of the photo to maximum .We find a text appeared in one of the trees in the photo. The text is IsJuS1Af0r3sTbR0. However, this is not the flag.



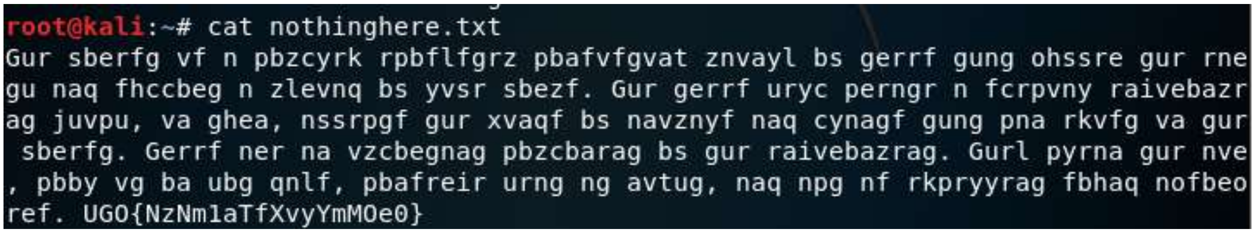
VIBRANCE -48
SATURATION 34
TEMPERATURE 60
TINT 0
HUE 0
BRIGHTNESS 0
EXPOSURE 100

2. We try using steghide to extract hidden files inside the image. We use the text obtained in step1 as a passphrase.



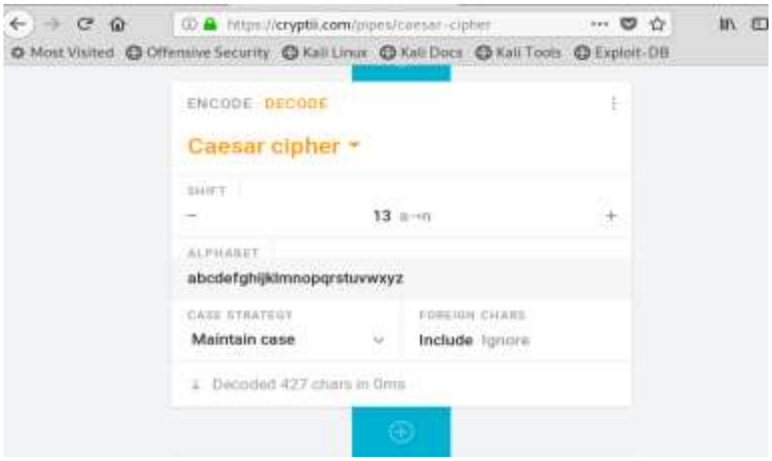
```
root@kali:~# steghide extract -sf forest.jpg
Enter passphrase:
wrote extracted data to "nothinghere.txt".
```

3.We found an encrypted text in the extracted nothinghere.txt.



```
root@kali:~# cat nothinghere.txt
Gur sberfg vf n pbzcyrk rpbflfgrz pbafvfgvat znvayl bs gerrf gung ohssre gur rne
gu naq fhccbeg n zlevnq bs yvsr sbezf. Gur gerrf uryc perngr n fcrpvny raivebazr
ag juvpu, va ghea, nssrpgf gur xvaqf bs navznyf naq cynagf gung pna rkvfg va gur
 sberfg. Gerrf ner na vzcbegnag pbzcbarag bs gur raivebazrag. Gurl pyrna gur nve
, pbby vg ba ubg qnlf, pbafreir urng ng avtug, naq npg nf rkpryyrag fbhaq nofbeo
ref. UGO{NzNm1aTfXvyYmMOe0}
```

4.We use Caesar cipher decoder and increase the shift until we find the text readable. In this case, it is shift 13. We found the flag.



ENCODE DECODE

Caesar cipher

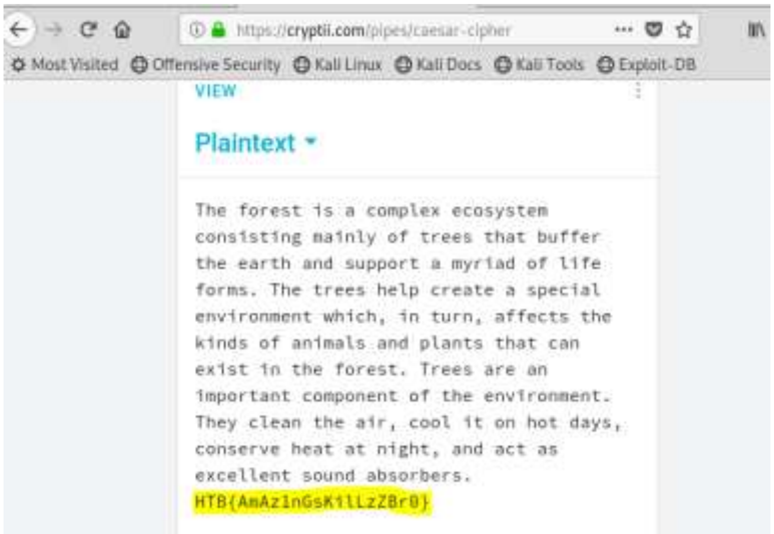
SHIFT – 13 a→n +

ALPHABET abcdefghijklmnopqrstuvwxyz

CASE STRATEGY Maintain case

FOREIGN CHARS Include Ignore

Decoded 427 chars in 0ms



VIEW

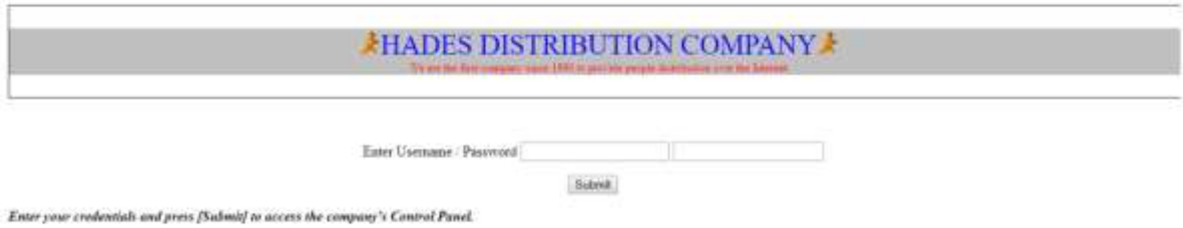
Plaintext

```
The forest is a complex ecosystem
consisting mainly of trees that buffer
the earth and support a myriad of life
forms. The trees help create a special
environment which, in turn, affects the
kinds of animals and plants that can
exist in the forest. Trees are an
important component of the environment.
They clean the air, cool it on hot days,
conserve heat at night, and act as
excellent sound absorbers.
HTB{AmAz1nGsKilLzZBr0}
```

Flag: HTB{AmAz1nGsKilLzZBr0}

19.0  HDC

1.We come across a login page.



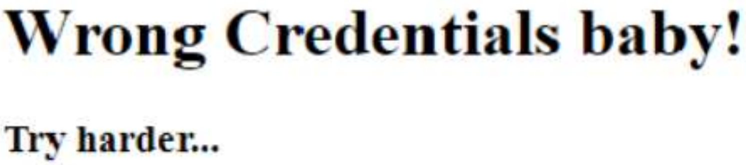
HADES DISTRIBUTION COMPANY

[illegible]

Enter Username / Password

*Enter your credentials and press [Submit] to access the company's Control Panel.*

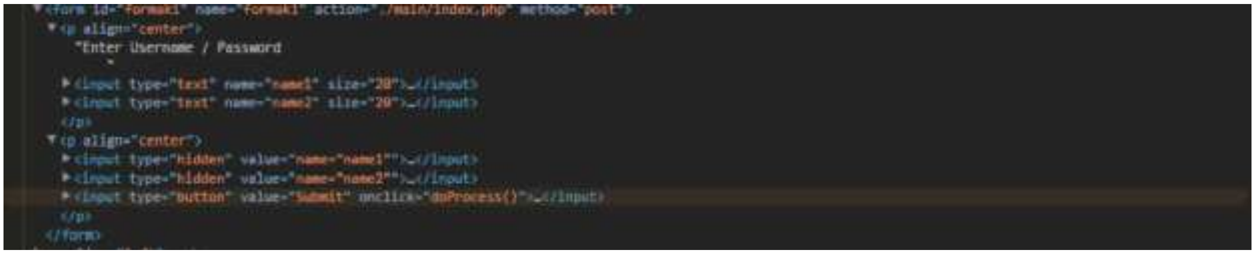
2.There will be an error message if you submit wrong username and password.



Wrong Credentials baby!

Try harder...

3.After inspecting the code of the login page we find 2 hidden inputs and a doProcess() function.



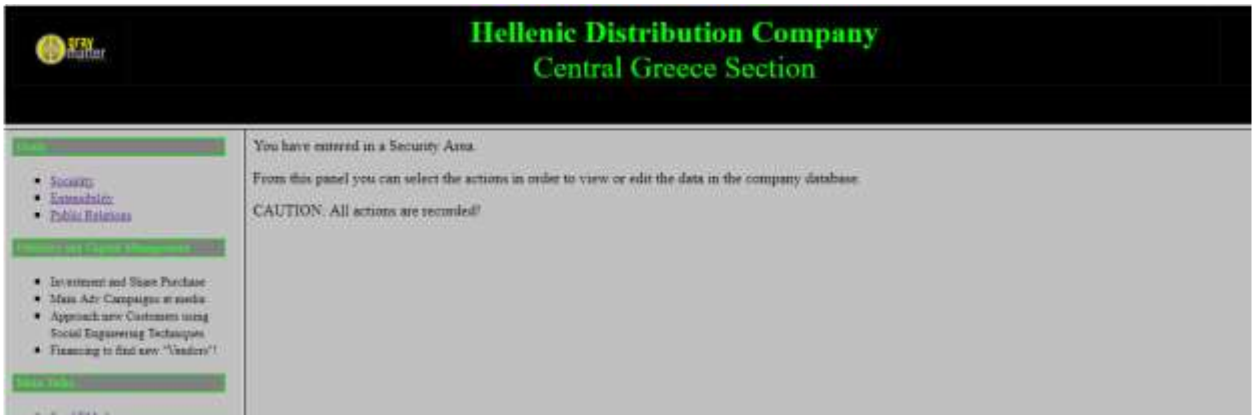
```
▼<form id="formaki" name="formaki" action="./main/index.php" method="post">
  ▼<p align="center">
      "Enter Username / Password
      "
    ▶<input type="text" name="name1" size="20">…</input>
    ▶<input type="text" name="name2" size="20">…</input>
    </p>
  ▼<p align="center">
    ▶<input type="hidden" value="name="name1"">…</input>
    ▶<input type="hidden" value="name="name2"">…</input>
    ▶<input type="button" value="Submit" onclick="doProcess()">…</input>
    </p>
  </form>
```

4. Open the developer tools and searching for “doProcess” in the entire document (Ctrl + Shift + F) will lead us to the following code snippet.

```
function doProcess() {
    var form = document.createElement("form");
    form.setAttribute("method", "post");
    form.setAttribute("action", "main/index.php");
    form.setAttribute("target", "view");
    var hiddenField = document.createElement("input");
    hiddenField.setAttribute("type", "hidden");
    hiddenField.setAttribute("name", "name1");
    hiddenField.setAttribute("value", "TXlMaXR0bGU");
    var hiddenField2 = document.createElement("input");
    hiddenField2.setAttribute("type", "hidden");
    hiddenField2.setAttribute("name", "name2");
    hiddenField2.setAttribute("value", "cDB3bmll");
    form.appendChild(hiddenField2);
    form.appendChild(hiddenField);
    form.appendChild(hiddenField2);
    document.body.appendChild(form);
      window.open('', 'view');
    form.submit();
}
```

There seems to be suspicious text as highlighted above.

5. We entered the text highlighted above, “TXlMaXR0bGU” and “cDB3bmll” as username and password respectively and we got in.

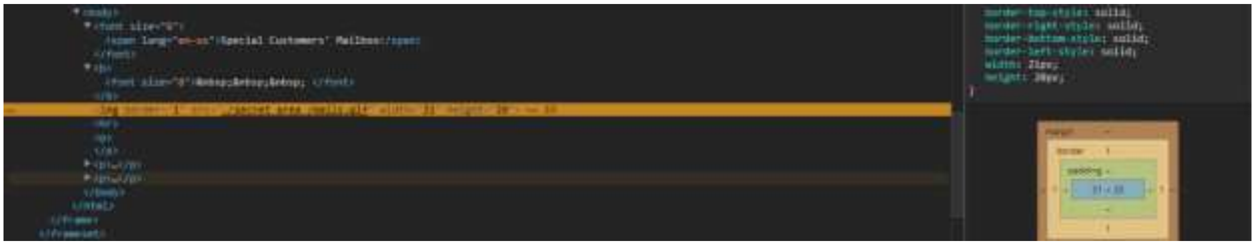

gray matter
Hellenic Distribution Company
Central Greece Section
Security
Public Relations
Investment and Share Purchase
Mass Adv Campaigns at media
Approach new Customers using Social Engineering Techniques
Financing to find new "Vendors"!

You have entered in a Security Area.

From this panel you can select the actions in order to view or edit the data in the company database.

CAUTION. All actions are recorded!

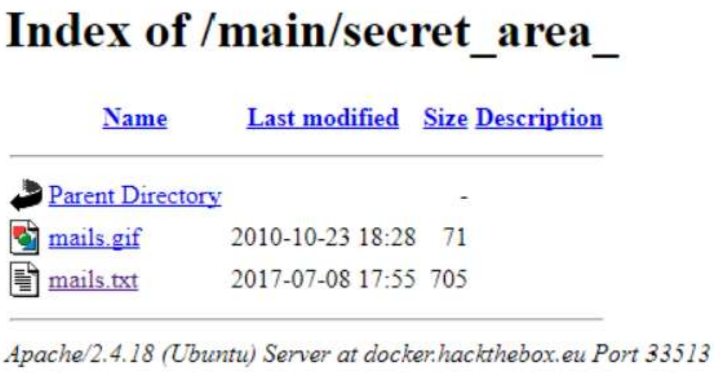
6. If we inspect the iframe  in the ‘Customers Area’ we will see that there is an image tag with its source property set to “ ./secret_area_/mails.gif”.



```
<body>
  <font size="6">
    <span lang="en-us">Special Customers' Mailbox</span>
  </font>
  <p>
    <font size="3">&nbsp;&nbsp;&nbsp; </font>
  </p>
  <img border="1" src="./secret_area_/mails.gif" width="21" height="20"> == $0
  <br>
  <p>
  </p>
  <p>…</p>
  <p>…</p>
</body>
</html>
</frame>
</frameset>
```

```
border-top-style: solid;
border-right-style: solid;
border-bottom-style: solid;
border-left-style: solid;
width: 21px;
height: 20px;
}
```

7.Visit http://docker.hackthebox.eu:XXX/main/secret_area_/, replace ‘XXX’ with your port instance number. We find some interesting files.

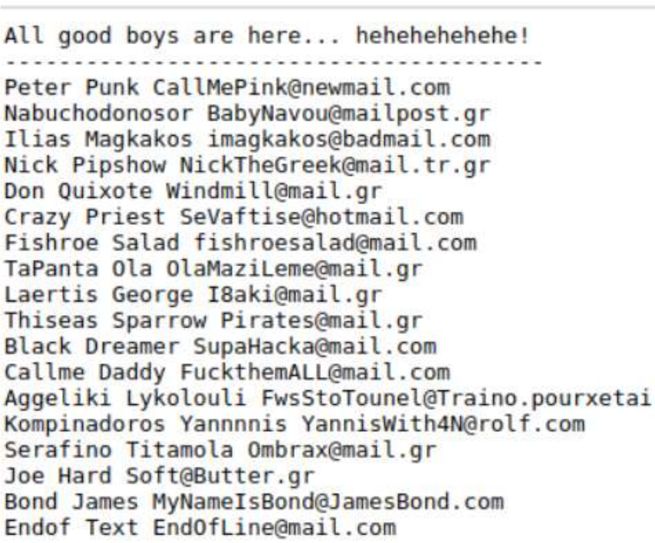


# Index of /main/secret_area_

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory |  | - |  |
| mails.gif | 2010-10-23 18:28 | 71 |  |
| mails.txt | 2017-07-08 17:55 | 705 |  |

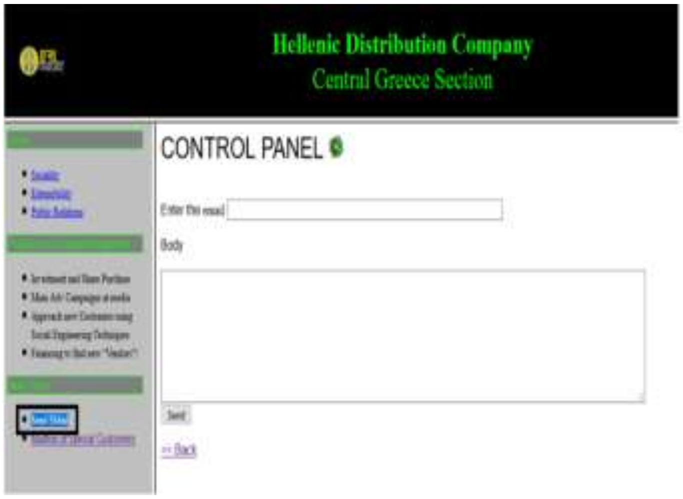
*Apache/2.4.18 (Ubuntu) Server at docker.hackthebox.eu Port 33513*

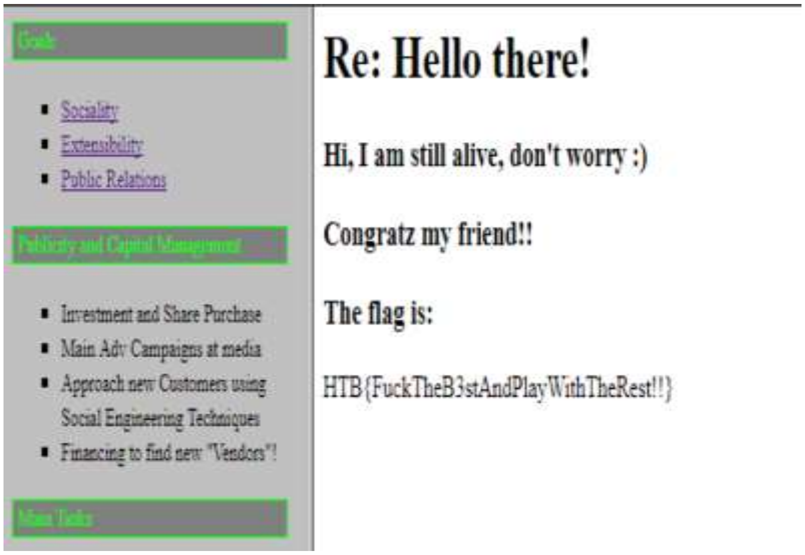
8.We find a list of emails. It might be that one of the email owners is the suspect.



```
All good boys are here... hehehehehehe!
---------------------------------------
Peter Punk CallMePink@newmail.com
Nabuchodonosor BabyNavou@mailpost.gr
Ilias Magkakos imagkakos@badmail.com
Nick Pipshow NickTheGreek@mail.tr.gr
Don Quixote Windmill@mail.gr
Crazy Priest SeVaftise@hotmail.com
Fishroe Salad fishroesalad@mail.com
TaPanta Ola OlaMaziLeme@mail.gr
Laertis George I8aki@mail.gr
Thiseas Sparrow Pirates@mail.gr
Black Dreamer SupaHacka@mail.com
Callme Daddy FuckthemALL@mail.com
Aggeliki Lykolouli FwsStoTounel@Traino.pourxetai
Kompinadoros Yannnnis YannisWith4N@rolf.com
Serafino Titamola Ombrax@mail.gr
Joe Hard Soft@Butter.gr
Bond James MyNameIsBond@JamesBond.com
Endof Text EndOfLine@mail.com
```

9)We can send email to each one of the emails listed in step 8 until we find a message leading us to the suspect by using ‘Send Email’.


Hellenic Distribution Company
Central Greece Section
CONTROL PANEL
Enter the email
Body
Send
<< Back
Sociality
Extensibility
Public Relations
Investment and Share Purchase
Main Adv Campaigns at media
Approach new Customers using Social Engineering Techniques
Financing to find new "Vendors"!
Send Email


Goals
Sociality
Extensibility
Public Relations
Publicity and Capital Management
Investment and Share Purchase
Main Adv Campaigns at media
Approach new Customers using Social Engineering Techniques
Financing to find new "Vendors"!
Main Tasks

Re: Hello there!

Hi, I am still alive, don't worry :)

Congratz my friend!!

The flag is:

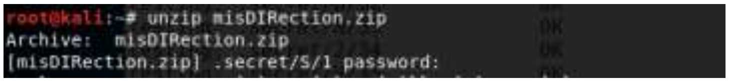
HTB{FuckTheB3stAndPlayWithTheRest!!}

The bad guy is ‘fishroesalad@mail.com’.

Flag: HTB{FuckTheB3stAndPlayWithTheRest!!}

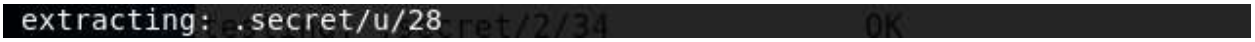
## 20.0 misDIrection

1.First, we unzip the file using the ‘hackthebox’ password.



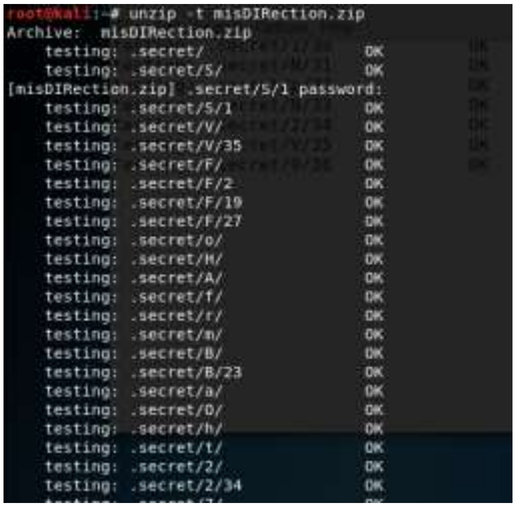
```
root@kali:~# unzip misDIRection.zip
Archive:  misDIRection.zip
[misDIRection.zip] .secret/S/1 password:
```

2.The terminal shows that a file is done being extracted.However, when we lock at the file browser, the file is not there. It seems that it is  hidden.



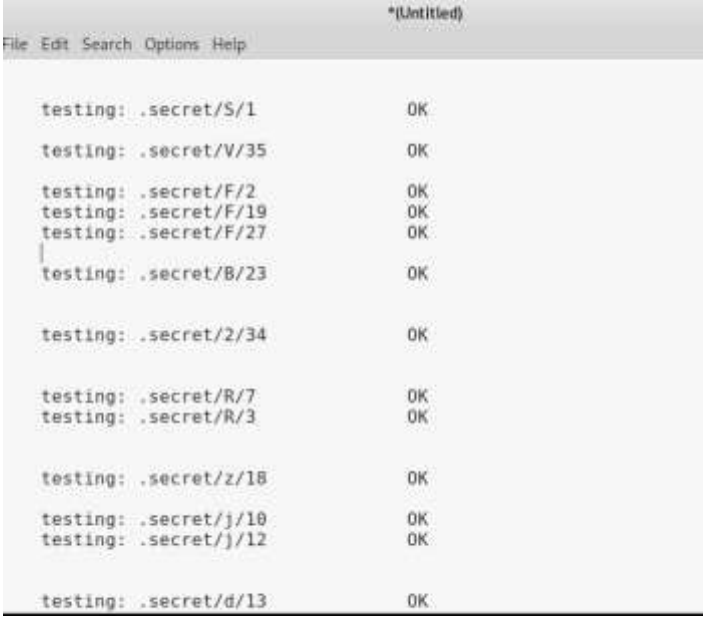
```
 extracting: .secret/u/28
```

3.We use  ‘unzip –t’ command to find out what the files extracted are.



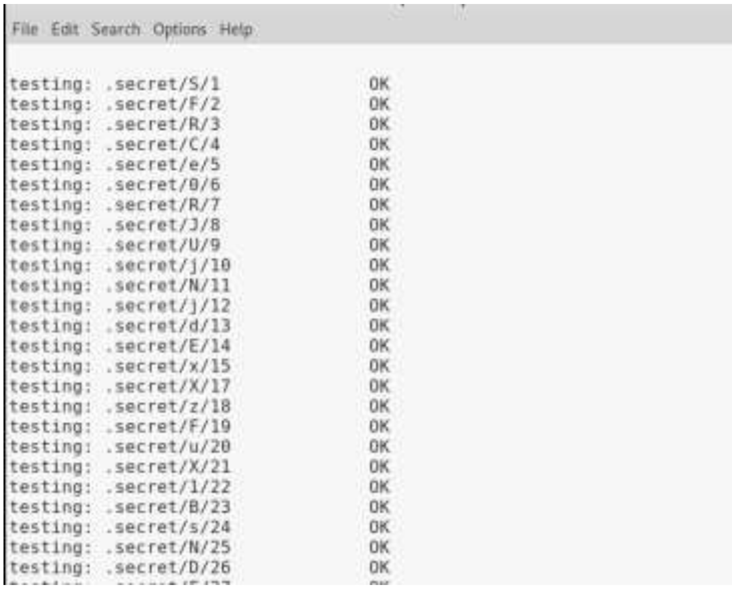
```
root@kali:~# unzip -t misDIRection.zip
Archive:  misDIRection.zip
    testing: .secret/                 OK
    testing: .secret/S/               OK
[misDIRection.zip] .secret/S/1 password:
    testing: .secret/S/1              OK
    testing: .secret/V/               OK
    testing: .secret/V/35             OK
    testing: .secret/F/               OK
    testing: .secret/F/2              OK
    testing: .secret/F/19             OK
    testing: .secret/F/27             OK
    testing: .secret/o/               OK
    testing: .secret/H/               OK
    testing: .secret/A/               OK
    testing: .secret/f/               OK
    testing: .secret/r/               OK
    testing: .secret/m/               OK
    testing: .secret/B/               OK
    testing: .secret/B/23             OK
    testing: .secret/a/               OK
    testing: .secret/D/               OK
    testing: .secret/h/               OK
    testing: .secret/t/               OK
    testing: .secret/2/               OK
    testing: .secret/2/34             OK
```

3)We notice that some of the files were empty and some have a single number for a file name. We copied the output in step3 and paste it to Leafpad.



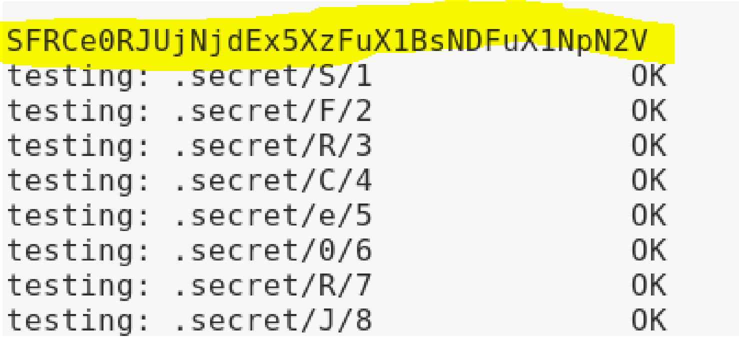
```
    testing: .secret/S/1              OK

    testing: .secret/V/35             OK

    testing: .secret/F/2              OK
    testing: .secret/F/19             OK
    testing: .secret/F/27             OK

    testing: .secret/B/23             OK


    testing: .secret/2/34             OK


    testing: .secret/R/7              OK
    testing: .secret/R/3              OK


    testing: .secret/z/18             OK

    testing: .secret/j/10             OK
    testing: .secret/j/12             OK


    testing: .secret/d/13             OK
```

4)We have to arrange the file in a certain order. So, delete the folders which did not contain a number. Arrange the files according to numbers. For example, 1 is S and 2 is F and so on.



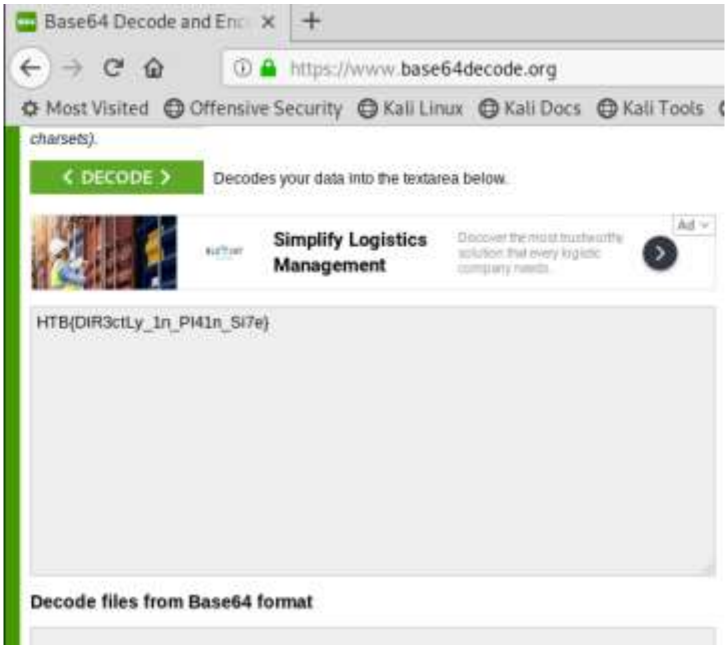
```
testing: .secret/S/1              OK
testing: .secret/F/2              OK
testing: .secret/R/3              OK
testing: .secret/C/4              OK
testing: .secret/e/5              OK
testing: .secret/0/6              OK
testing: .secret/R/7              OK
testing: .secret/J/8              OK
testing: .secret/U/9              OK
testing: .secret/j/10             OK
testing: .secret/N/11             OK
testing: .secret/j/12             OK
testing: .secret/d/13             OK
testing: .secret/E/14             OK
testing: .secret/x/15             OK
testing: .secret/X/17             OK
testing: .secret/z/18             OK
testing: .secret/F/19             OK
testing: .secret/u/20             OK
testing: .secret/X/21             OK
testing: .secret/1/22             OK
testing: .secret/B/23             OK
testing: .secret/s/24             OK
testing: .secret/N/25             OK
testing: .secret/D/26             OK
```

5)Pay attention to the letters in the middle as we arrange the lines. Use Ctrl+F to find text better. We find a mysterious word after arranging the letters.



```
SFRCe0RJUjNjdEx5XzFuX1BsNDFuX1NpN2V
testing:  .secret/S/1              OK
testing:  .secret/F/2              OK
testing:  .secret/R/3              OK
testing:  .secret/C/4              OK
testing:  .secret/e/5              OK
testing:  .secret/0/6              OK
testing:  .secret/R/7              OK
testing:  .secret/J/8              OK
```

6)We tried decoding it with base64 and found the flag.


Base64 Decode and Enc
https://www.base64decode.org
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
charsets).
DECODE
Decodes your data into the textarea below.
Simplify Logistics Management
Ad
Decode files from Base64 format

```
HTB{DIR3ctLy_1n_Pl41n_Si7e}
```

Flag: HTB{DIR3ctLy_1n_Pl41n_Si7e}