Assignment 0x01 - Intro & Cryptography

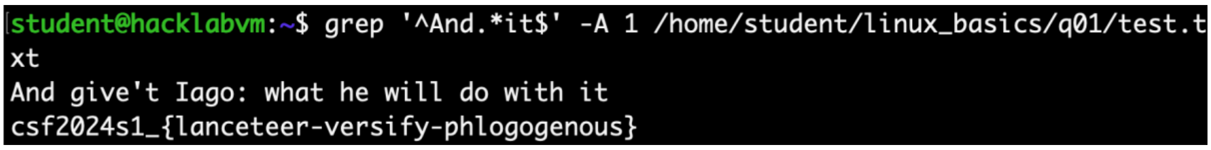
Part I - Intro

You may start this part after completing workshop 0x01.

Instructions:

- Access **/home/student/linux_basics** on HackLab VM, either by accessing the VM via SSH or via the graphical login (username=student, password=password).
- Please ensure you provide the details of what you did (command executed, script written, screenshot of any results, etc), and not just provide the answer!
- Upload your assignment as PDF by the due date

1. (1 point) There is a secret passphrase embedded in the file called "**test.txt**" in the folder /home/student/linux_basics/q01. The secret can be found **following** the line that begins with the word "**And**" and ends with "**it**". Use grep with regular expression to locate this line and the line following it. What is the passphrase?



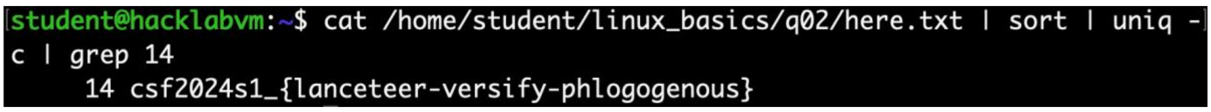
```
[student@hacklabvm:~$ grep '^And.*it$' -A 1 /home/student/linux_basics/q01/test.t
xt
And give't Iago: what he will do with it
csf2024s1_{lanceteer-versify-phlogogenous}
```

Command: grep '^And.*it$' -A 1 /home/student/linux_basics/q01/test.txt

"^" means the start of one line, "And" for find the word And, ".*" means any characters in the middle, "it" followed by "$" means the line is end with it. Then the "-A 1" means print 1 line after the matching line. The "/home/student/linux_basics/q01/test.txt" is the relatively path of the file.

The passphrase is "csf2024s1_{lanceteer-versify-phlogogenous}"

2. (1 point) In the folder /home/student/linux_basics/q02, there is a file called "**here.txt**" that contains passphrases. Find the passphrase that occur exactly **14 times**.



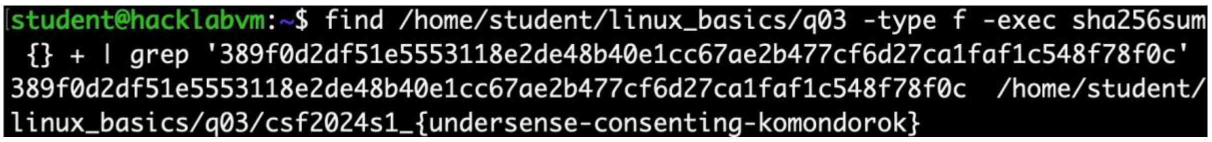
```
[student@hacklabvm:~$ cat /home/student/linux_basics/q02/here.txt | sort | uniq -
c | grep 14
     14 csf2024s1_{lanceteer-versify-phlogogenous}
```

Command: cat /home/student/linux_basics/q02/here.txt | sort | uniq -c | grep '14'

"cat /home/student/linux_basics/q02/here.txt" means get the content of the file, then use "|" pipe pass the content to the command "sort" to sort the content in order then pipe the sorted file content to “uniq -c” command to prefixes each line of the output with the count of occurrences, then it was piped to the command “grep 14” to match any lines start with the number 14.

The passphrase is “csf2024s1_{lanceteer-versify-phlogogenous}”

3. (1 point) There are lots of files in /home/student/linux_basics/q03. What is the name of the file whose SHA256 sum is 389f0d2df51e5553118e2de48b40e1cc67ae2b477cf6d27ca1faf1c548f78f0c ?



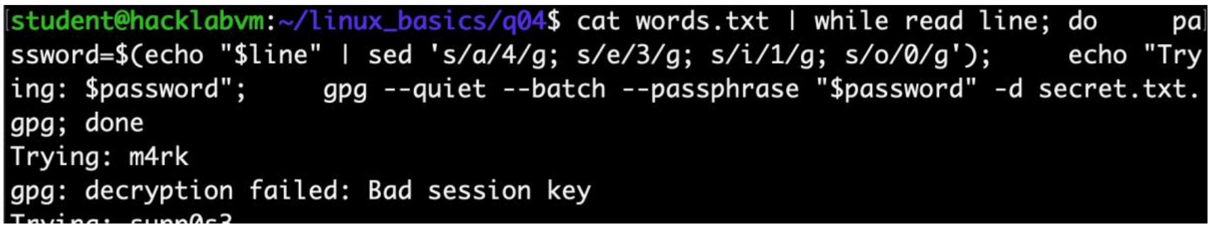
```
student@hacklabvm:~$ find /home/student/linux_basics/q03 -type f -exec sha256sum
 {} + | grep '389f0d2df51e5553118e2de48b40e1cc67ae2b477cf6d27ca1faf1c548f78f0c'
389f0d2df51e5553118e2de48b40e1cc67ae2b477cf6d27ca1faf1c548f78f0c  /home/student/
linux_basics/q03/csf2024s1_{undersense-consenting-komondorok}
```

Command: find /home/student/linux_basics/q03 -type f -exec sha256sum {} + | grep '389f0d2df51e5553118e2de48b40e1cc67ae2b477cf6d27ca1faf1c548f78f0c'

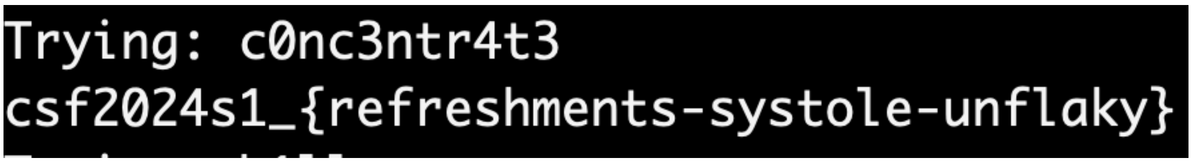
“find” command to find a file, “-type f” restrict it to be just a file rather than a dictionary or something else, “-exec” means execute the command “sha256sum {} +” for every single file, while executing, “{}” is replaced by the pathname of the current file and “+” combine all together in one line. Then pipe the result to the command “grep” to find the file with the same SHA256 sum.

The passphrase is: “csf2024s1_{undersense-consenting-komondorok}”

4. (1 point) Generate a list of passwords from the source file words.txt. Use "l33t" conversion so that a=>4, e=>3, i=>1, and o=>0. For example "hello world" becomes "h3ll0 w0rld". There is a file encrypted using gpg (Gnu Privacy Protection) under /home/student/linux_basics/q04, using the command gpg -c --batch --passphrase <pass>. Unfortunately, we have forgotten the password. Use the "l33t" converted password list to brute-force the encrypted file. This may take a few minutes. **Please provide both the correct password and the content of the decrypted file**.



```
student@hacklabvm:~/linux_basics/q04$ cat words.txt | while read line; do     pa
ssword=$(echo "$line" | sed 's/a/4/g; s/e/3/g; s/i/1/g; s/o/0/g');     echo "Try
ing: $password";     gpg --quiet --batch --passphrase "$password" -d secret.txt.
gpg; done
Trying: m4rk
gpg: decryption failed: Bad session key
Trying: supp0s3
```



```
Trying: c0nc3ntr4t3
csf2024s1_{refreshments-systole-unflaky}
```

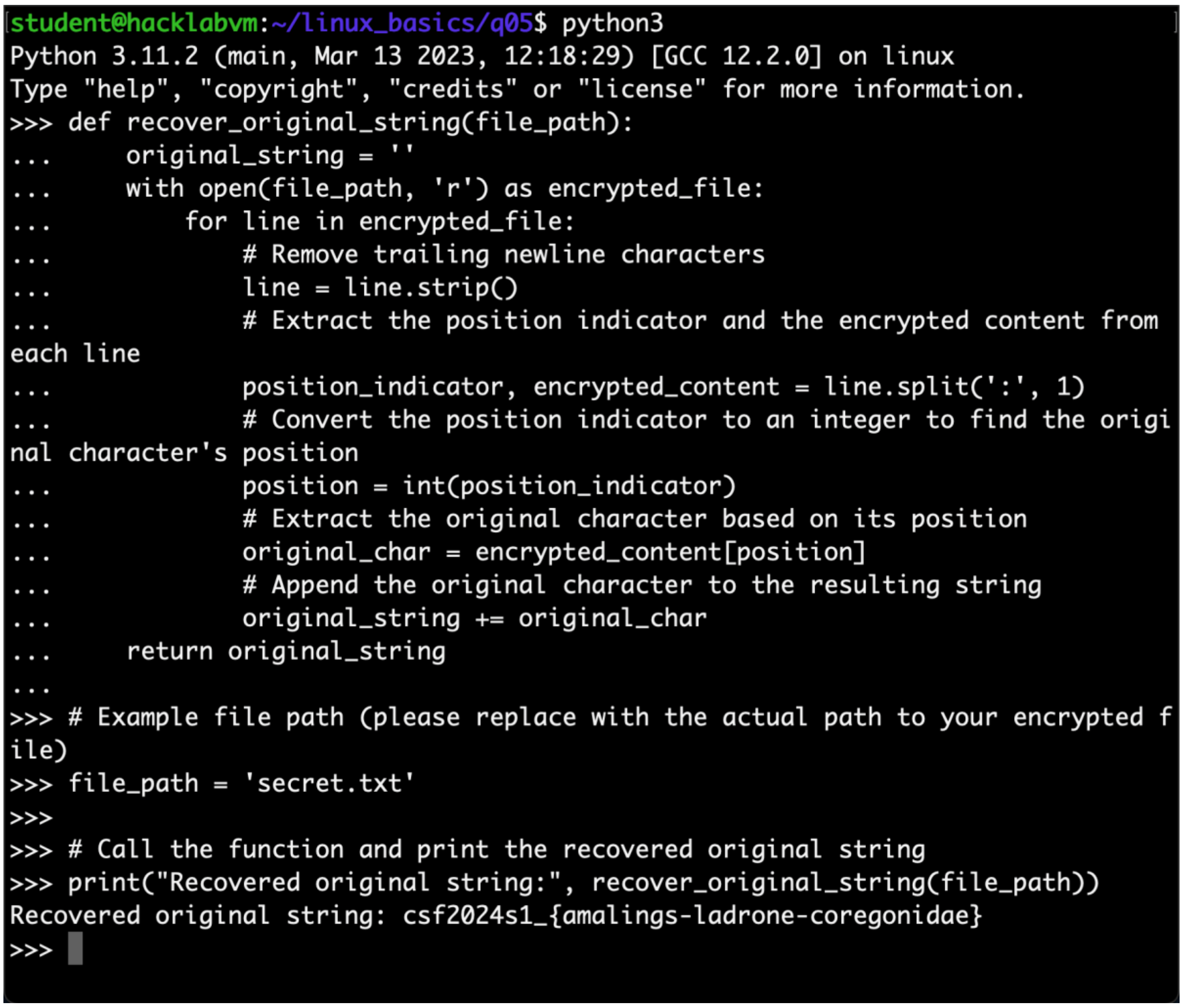
Command & Script: cat words.txt | while read line; do password=$(echo "$line" | sed 's/a/4/g; s/e/3/g; s/i/1/g; s/o/0/g'); echo "Trying: $password"; gpg --quiet --batch --passphrase "$password" -d secret.txt.gpg; done

Using "cat words.txt" to capture the list of password to the IO stream and use pipe "|" pass it to "while read line" which read line by line in the file and use "l33t" conversion it to the needed passwords one by one to the variable "password". Give a hint display on the screen "Trying ..." then use "gpg" command with the same parameters: "batch", "passphrase" and specify the file which named "secret.txt.gpg" by the parameter "-d" to brute-force the encrypted file.

Password: c0nc3ntr4t3

The passphrase is: "csf2024s1_{refreshments-systole-unflaky}"

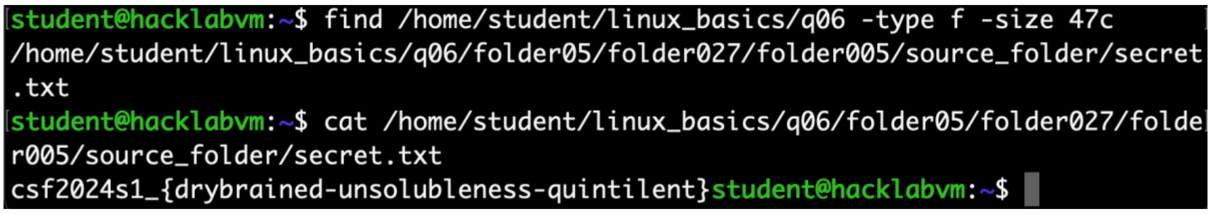
5. (1 point) Find the flag hidden (encoded?) in the file /home/student/linux_basics/q05/secret.txt. Looks like the cyber.py (Hint: the file cyber.py was used to generate the file...)


```
student@hacklabvm:~/linux_basics/q05$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> def recover_original_string(file_path):
...     original_string = ''
...     with open(file_path, 'r') as encrypted_file:
...         for line in encrypted_file:
...             # Remove trailing newline characters
...             line = line.strip()
...             # Extract the position indicator and the encrypted content from
each line
...             position_indicator, encrypted_content = line.split(':', 1)
...             # Convert the position indicator to an integer to find the origi
nal character's position
...             position = int(position_indicator)
...             # Extract the original character based on its position
...             original_char = encrypted_content[position]
...             # Append the original character to the resulting string
...             original_string += original_char
...     return original_string
...
>>> # Example file path (please replace with the actual path to your encrypted f
ile)
>>> file_path = 'secret.txt'
>>>
>>> # Call the function and print the recovered original string
>>> print("Recovered original string:", recover_original_string(file_path))
Recovered original string: csf2024s1_{amalings-ladrone-coregonidae}
>>>
```

After looking at the “cyber.py” file, I notified that the first 4 numbers indicate where the original character should be in each line, then I write a python scrip to reverse the path to restore the passphrase.

The passphrase is: “csf2024s1_{amalings-ladrone-coregonidae}”

6. (1 point) There are lots of sub-directories and files under /home/student/linux_basics/q06. Find the file containing the secret. The file has size of exactly **47** bytes long.

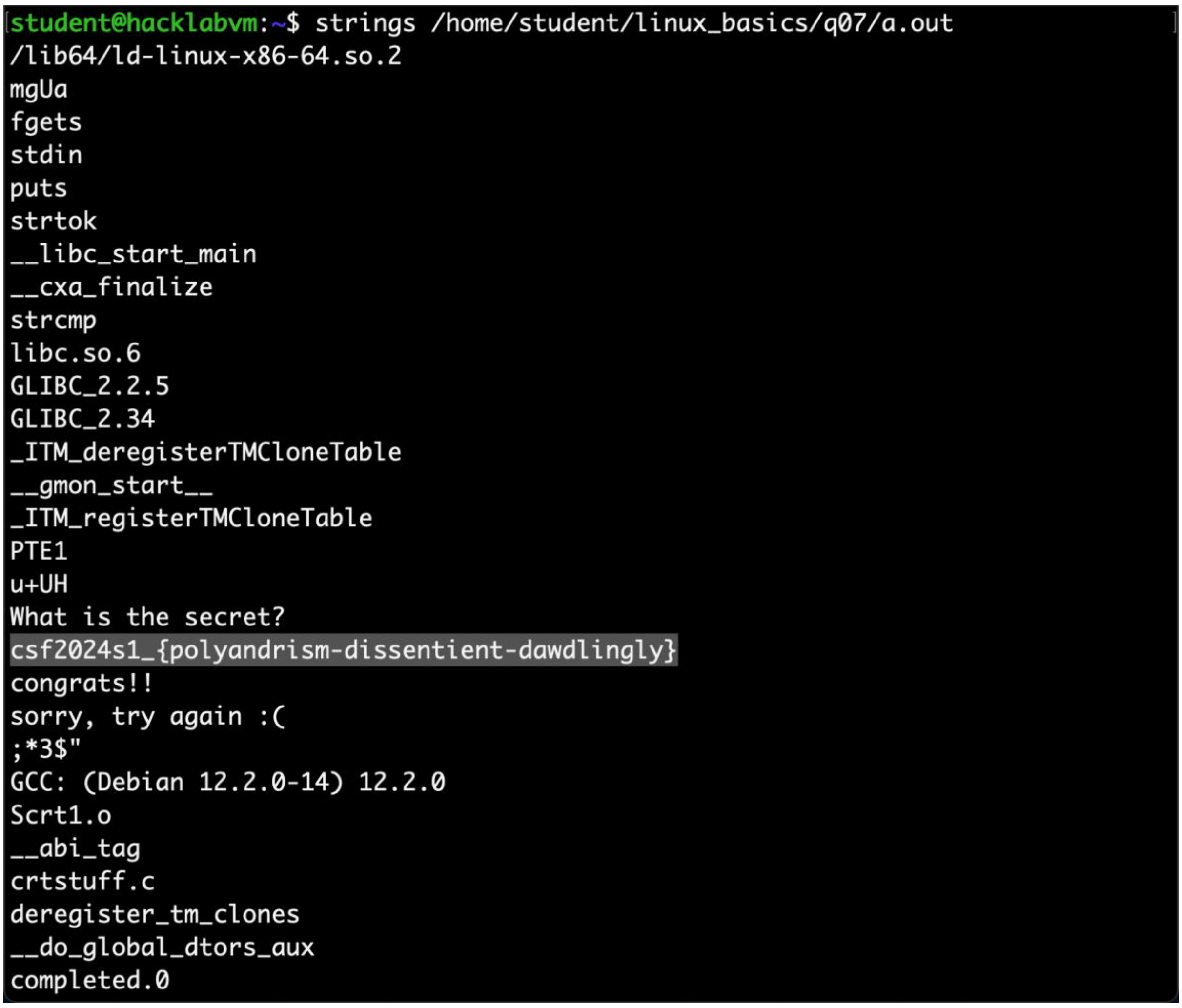

```
student@hacklabvm:~$ find /home/student/linux_basics/q06 -type f -size 47c
/home/student/linux_basics/q06/folder05/folder027/folder005/source_folder/secret
.txt
student@hacklabvm:~$ cat /home/student/linux_basics/q06/folder05/folder027/folde
r005/source_folder/secret.txt
csf2024s1_{drybrained-unsolubleness-quintilent}student@hacklabvm:~$
```

Using the “find” command with the parameters “-type f” to find a file and “-size 47c” to find the file which is exactly 47 bytes long (c stands for bytes).

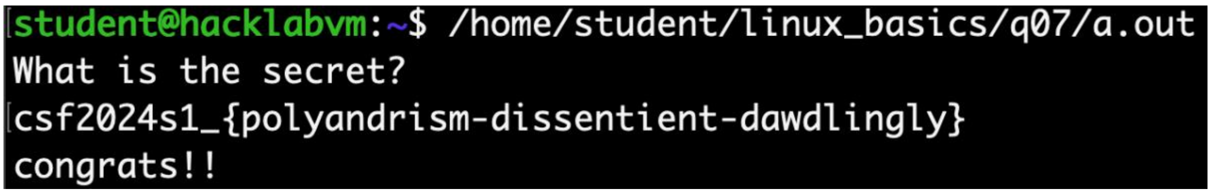
Then use the “cat” command to see the content inside the founded file.

The passphrase is: “csf2024s1_{drybrained-unsolubleness-quintilent}”

7. (1 point) Execute /home/student/linux_basics/q07/a.out and try to guess the secret.



```
student@hacklabvm:~$ strings /home/student/linux_basics/q07/a.out
/lib64/ld-linux-x86-64.so.2
mgUa
fgets
stdin
puts
strtok
__libc_start_main
__cxa_finalize
strcmp
libc.so.6
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
What is the secret?
csf2024s1_{polyandrism-dissentient-dawdlingly}
congrats!!
sorry, try again :(
;*3$"
GCC: (Debian 12.2.0-14) 12.2.0
Scrt1.o
__abi_tag
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
```

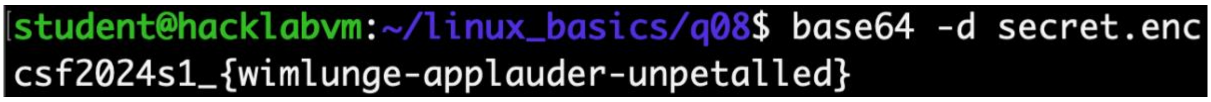


```
student@hacklabvm:~$ /home/student/linux_basics/q07/a.out
What is the secret?
csf2024s1_{polyandrism-dissentient-dawdlingly}
congrats!!
```

Using the “strings” command to read the binary file, the answer shows in the file!

The passphrase is: “csf2024s1_{polyandrism-dissentient-dawdlingly}”

8. (1 point) There is an encoded secret in /home/student/linux_basics/q08 folder. Decode to get the secret.



```
[student@hacklabvm:~/linux_basics/q08$ base64 -d secret.enc
csf2024s1_{wimlunge-applauder-unpetalled}
```

Try to use a common encode method “base64”, the answer comes out!

The passphrase is: “csf2024s1_{wimlunge-applauder-unpetalled}”

Part II - Cryptography

You may start this part after completing workshop 0x02.

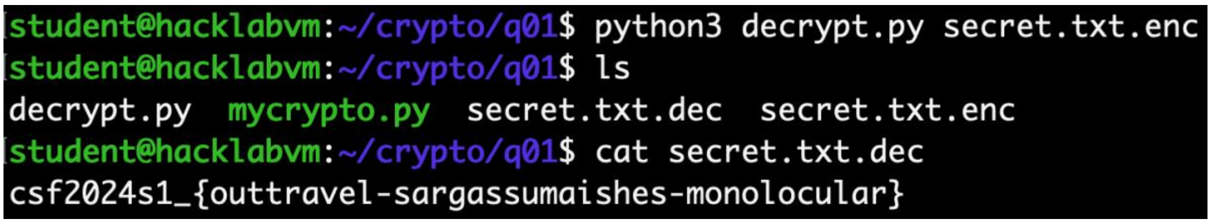
1. (1 point) Go to /home/student/crypto/q01 on the HacklabVM (either via the graphical interface or via SSH). The file secret.txt.enc has been "encrypted" using this simple crypter code below (see mycrypto.py in the same directory). "Decrypt" the file and find the secret. Is this a good encryption? What is the key space?

```
#!/usr/bin/python3

import argparse

import os

import sys

import random

def mycrypto (filename):

    with open(filename, 'r') as f, open(filename + '.enc', 'w') as o:

        blob = f.read()

        for b in blob:

            key = random.randrange(255)

            x = ord(b) ^ key

            o.write(chr(x))


def main():

    parser = argparse.ArgumentParser(description='Encrypt (?) a file')

    parser.add_argument('filename', metavar='filename', type=str, help='file to encrypt')

    parser.add_argument('--seed', metavar='seed',type=int,default=312024,help='seed')

    args = parser.parse_args()


    if not os.path.isfile(args.filename):
        print('The file  does not exist')

        sys.exit()


    random.seed(args.seed)

    mycrypto(args.filename)


if __name__== "__main__":

        main()
```

It is not a good encryption because it uses the same seed to encrypt the secret text that if the hacker got the seed, the person could crack all the cyphertexts using it very easy. The key space is A ⊕ B ⊕ B = A, so just write a similar script to use the same seed to do the XOR operation again can restore the original passphrase. Here is my script in python:

```
“ #!/usr/bin/python3

import argparse

import os

import sys

import random

def mydecrypt(filename, seed=312024):

  random.seed(seed)

  with open(filename, 'r') as f, open(filename.replace('.enc', '.dec'), 'w') as o:

    blob = f.read()

    for b in blob:

      key = random.randrange(255)

      x = ord(b) ^ key

      o.write(chr(x))

def main():

  parser = argparse.ArgumentParser(description='Decrypt a file')

  parser.add_argument('filename', metavar='filename', type=str, help='file to
decrypt')
```

```
    args = parser.parse_args()

    if not os.path.isfile(args.filename):

        print('The file does not exist')

        sys.exit()

    mydecrypt(args.filename)

if __name__ == "__main__":

    main() "
```



```
student@hacklabvm:~/crypto/q01$ python3 decrypt.py secret.txt.enc
student@hacklabvm:~/crypto/q01$ ls
decrypt.py  mycrypto.py  secret.txt.dec  secret.txt.enc
student@hacklabvm:~/crypto/q01$ cat secret.txt.dec
csf2024s1_{outtravel-sargassumaishes-monolocular}
```

The passphrase is: "csf2024s1_{outtravel-sargassumaishes-monolocular}"

2. (1 point) A short, plaintext message has been encrypted using Textbook RSA (using the public exponent) with the same parameters used in the workshop. The encrypted message can be found in /home/student/crypto/q02 folder. Decrypt the secret.

```
n=0x9B51C20306EDE535C8FCAADBC3F3515E52A0D005703DD449BEC66B23E2932313
p=0xC5A047A7C52ED3A2875F7D76C47B555F
q=0xC93268355C09197BBF1659B5522FFACD
e=0x010001
d=0x0D067636BAC6088AD2281E4BFFCACFEFEF9BC1A69FB9E701063DFBAAB436E4C1
```

Notes

- The numbers above are all in hexadecimal
- Use the sample code from workshop
- You can use the following helper functions in Python3 to byte-wise convert between string and integer.

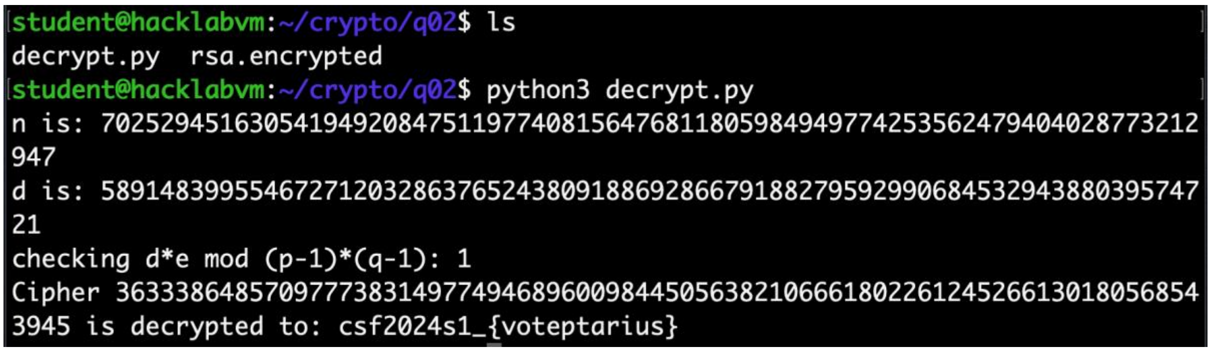
```
import binascii


# convert string to integer using

def string_to_int(string):

    return int.from_bytes(binascii.a2b_qp(string),byteorder='big')
```

```
# convert into back to string

def int_to_string(number):

   bin = number.to_bytes((number.bit_length() + 7) // 8, byteorder='big')

   return binascii.b2a_qp(bin).decode("utf-8")
```

Follow the helper functions and the sample code from workshop, I write a script to do the decrypt process just like the workshop script.

```
“import binascii

# Copy supplied RSA parameters

p = int("0xC5A047A7C52ED3A2875F7D76C47B555F", 16)  # First prime

q = int("0xC93268355C09197BBF1659B5522FFACD", 16)  # Second prime

e = int("0x010001", 16)  # A number that is co-prime with (p-1)*(q-1)

d =
int("0x0D067636BAC6088AD2281E4BFFCACFEFEF9BC1A69FB9E701063DFBAA
B436E4C1", 16)  # Decryption key

enc =
int("0x50543d1e0fda637c109bb32c706dbaec8d8d20ce001cca02f8576a4852a0
72c9", 16)  # Encrypted message

n = p * q  # Modulus

# Print n, d and check d*e mod (p-1)*(q-1)

print("n is: " + str(n))

print("d is: " + str(d))

print("checking d*e mod (p-1)*(q-1): " + str((d * e) % ((p - 1) * (q - 1))))

# Decrypt message using private exponent d

plain = pow(enc, d, n)


# Convert plain (in int) to str byte-wise by first representing int as hex, then
converting that to ascii

decrypted_message_hex = format(plain, 'x')  # Convert to hex format

decrypted_message = binascii.unhexlify(decrypted_message_hex)  # Convert
hex to bytes
```

print("Cipher " + str(enc) + " is decrypted to: " + decrypted_message.decode())"



```
student@hacklabvm:~/crypto/q02$ ls
decrypt.py  rsa.encrypted
student@hacklabvm:~/crypto/q02$ python3 decrypt.py
n is: 70252945163054194920847511977408156476811805984949774253562479404028773212
947
d is: 58914839955467271203286376524380918869286679188279592990684532943880395747
21
checking d*e mod (p-1)*(q-1): 1
Cipher 36333864857097773831497749468960098445056382106661802261245266130180568543
3945 is decrypted to: csf2024s1_{voteptarius}
```

The passphrase is: "csf2024s1_{voteptarius}"

3. (1 point) Find the message hidden inside this encrypted bitmap image Download encrypted bitmap image. You don't need to decrypt.

   Use a simple scripe to create a normal bmp file with the same parameter
   "from PIL import Image

   # Create a 2000x2000 image, "p" stand for 256 color

   img = Image.new('P', (2000, 2000))

   # Use black to fill the blank

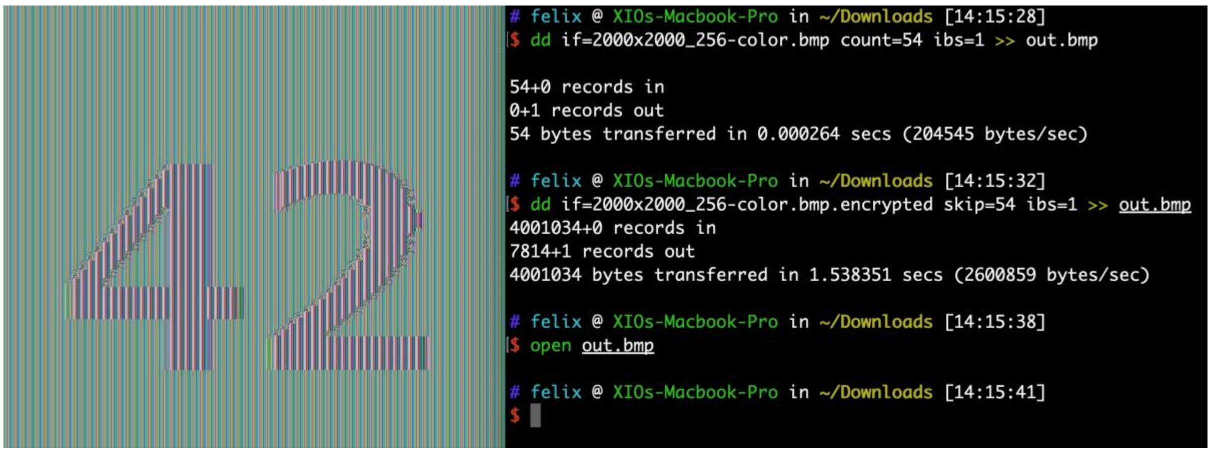
   img.paste(0, (0, 0, 2000, 2000))

   # Save the image

   img.save('2000x2000_256-color.bmp')

   print("Success")"

   Then use the similar command as the workshop did to show the pattern of the image which can be recognized.

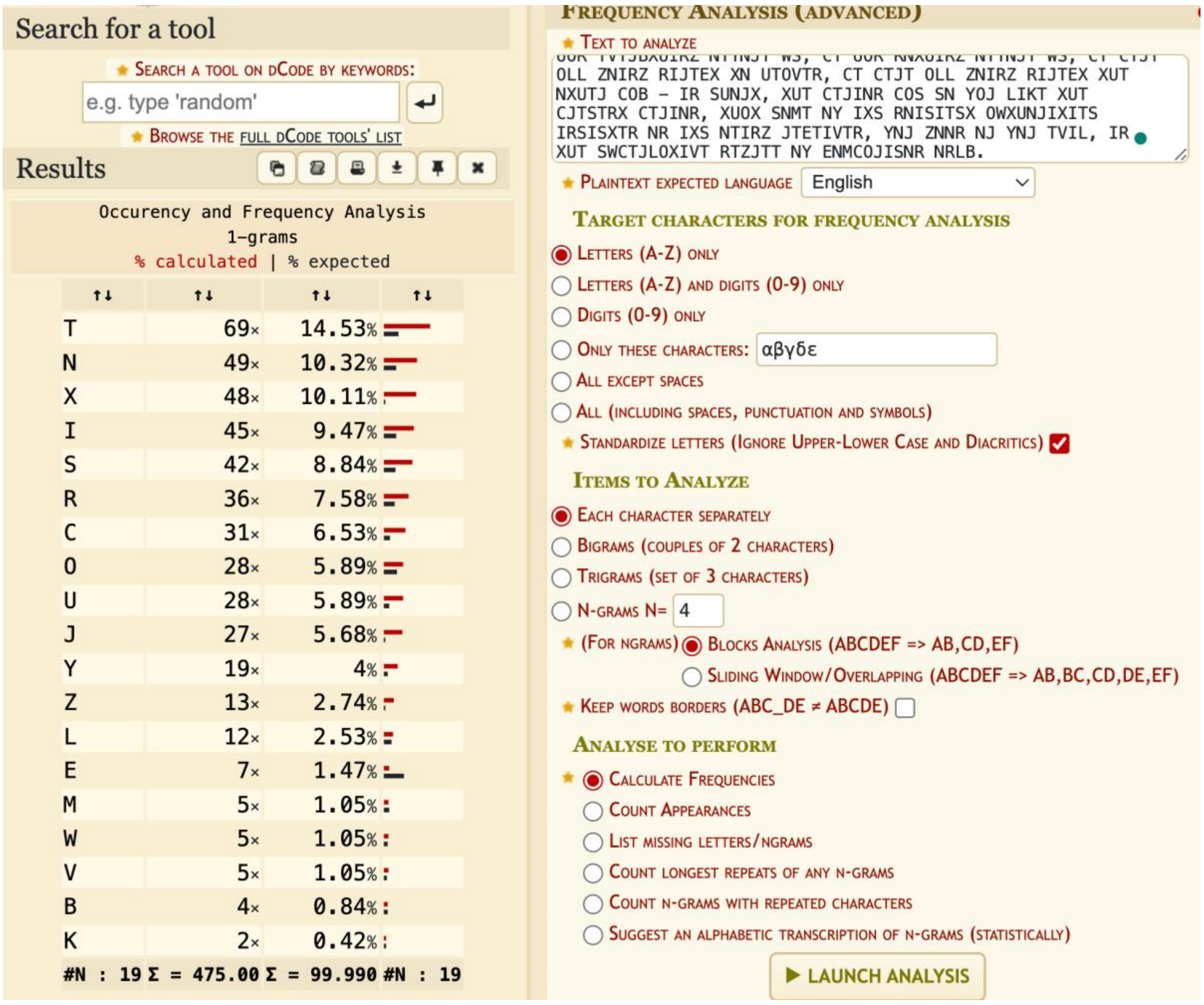


```
# felix @ XIOs-Macbook-Pro in ~/Downloads [14:15:28]
$ dd if=2000x2000_256-color.bmp count=54 ibs=1 >> out.bmp

54+0 records in
0+1 records out
54 bytes transferred in 0.000264 secs (204545 bytes/sec)

# felix @ XIOs-Macbook-Pro in ~/Downloads [14:15:32]
$ dd if=2000x2000_256-color.bmp.encrypted skip=54 ibs=1 >> out.bmp
4001034+0 records in
7814+1 records out
4001034 bytes transferred in 1.538351 secs (2600859 bytes/sec)

# felix @ XIOs-Macbook-Pro in ~/Downloads [14:15:38]
$ open out.bmp

# felix @ XIOs-Macbook-Pro in ~/Downloads [14:15:41]
$
```

So the answer is 42.

4. (1 points) Go to /home/student/crypto/q04 and find the file ciphertext.txt, which was created using one of the classical ciphers. Find out what the original text is. What is the key used?

   The content is look like a substitution cipher, first of all, use the online tool mentioned in the lecture to test the text.



Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS: e.g. type 'random'

BROWSE THE FULL DCODE TOOLS' LIST

Results

Occurency and Frequency Analysis
1-grams
% calculated | % expected

↑↓	↑↓	↑↓	↑↓
T	69×	14.53%	
N	49×	10.32%	
X	48×	10.11%	
I	45×	9.47%	
S	42×	8.84%	
R	36×	7.58%	
C	31×	6.53%	
O	28×	5.89%	
U	28×	5.89%	
J	27×	5.68%	
Y	19×	4%	
Z	13×	2.74%	
L	12×	2.53%	
E	7×	1.47%	
M	5×	1.05%	
W	5×	1.05%	
V	5×	1.05%	
B	4×	0.84%	
K	2×	0.42%	
#N : 19	Σ = 475.00	Σ = 99.990	#N : 19

FREQUENCY ANALYSIS (ADVANCED)

TEXT TO ANALYZE

```
OLL ZNIRZ RIJTEX XN UTOVTR, CT CTJT OLL ZNIRZ RIJTEX XUT
NXUTJ COB – IR SUNJX, XUT CTJINR COS SN YOJ LIKT XUT
CJTSTRX CTJINR, XUOX SNMT NY IXS RNISITSX OWXUNJIXITS
IRSISXTR NR IXS NTIRZ JTETIVTR, YNJ ZNNR NJ YNJ TVIL, IR
XUT SWCTJLOXIVT RTZJTT NY ENMCOJISNR NRLB.
```

PLAINTEXT EXPECTED LANGUAGE: English

TARGET CHARACTERS FOR FREQUENCY ANALYSIS

[✓ LETTERS (A-Z) ONLY]
LETTERS (A-Z) AND DIGITS (0-9) ONLY
DIGITS (0-9) ONLY
ONLY THESE CHARACTERS: αβγδε
ALL EXCEPT SPACES
ALL (INCLUDING SPACES, PUNCTUATION AND SYMBOLS)
STANDARDIZE LETTERS (IGNORE UPPER-LOWER CASE AND DIACRITICS) ✓

ITEMS TO ANALYZE

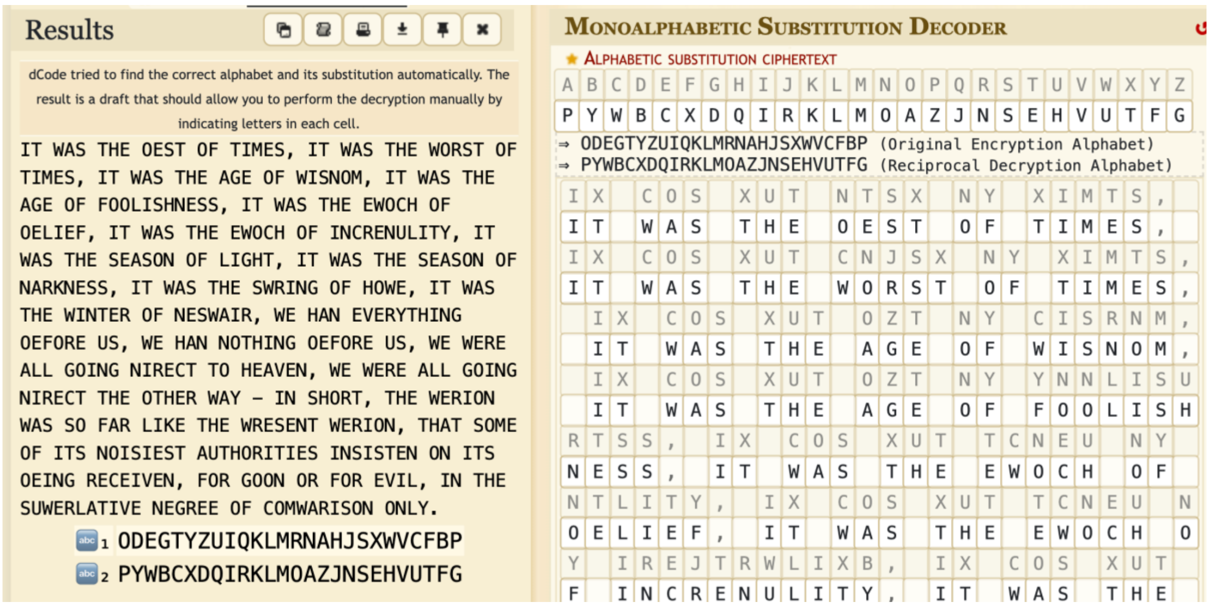
[✓ EACH CHARACTER SEPARATELY]
BIGRAMS (COUPLES OF 2 CHARACTERS)
TRIGRAMS (SET OF 3 CHARACTERS)
N-GRAMS N= 4
(FOR NGRAMS) [✓ BLOCKS ANALYSIS (ABCDEF => AB,CD,EF)]
SLIDING WINDOW/OVERLAPPING (ABCDEF => AB,BC,CD,DE,EF)
KEEP WORDS BORDERS (ABC_DE ≠ ABCDE)

ANALYSE TO PERFORM

[✓ CALCULATE FREQUENCIES]
COUNT APPEARANCES
LIST MISSING LETTERS/NGRAMS
COUNT LONGEST REPEATS OF ANY N-GRAMS
COUNT N-GRAMS WITH REPEATED CHARACTERS
SUGGEST AN ALPHABETIC TRANSCRIPTION OF N-GRAMS (STATISTICALLY)

▶ LAUNCH ANALYSIS

Then use the online tool in the same website to decrypt the text.


Results

dCode tried to find the correct alphabet and its substitution automatically. The result is a draft that should allow you to perform the decryption manually by indicating letters in each cell.

```
IT WAS THE OEST OF TIMES, IT WAS THE WORST OF
TIMES, IT WAS THE AGE OF WISNOM, IT WAS THE
AGE OF FOOLISHNESS, IT WAS THE EWOCH OF
OELIEF, IT WAS THE EWOCH OF INCRENULITY, IT
WAS THE SEASON OF LIGHT, IT WAS THE SEASON OF
NARKNESS, IT WAS THE SWRING OF HOWE, IT WAS
THE WINTER OF NESWAIR, WE HAN EVERYTHING
OEFORE US, WE HAN NOTHING OEFORE US, WE WERE
ALL GOING NIRECT TO HEAVEN, WE WERE ALL GOING
NIRECT THE OTHER WAY – IN SHORT, THE WERION
WAS SO FAR LIKE THE WRESENT WERION, THAT SOME
OF ITS NOISIEST AUTHORITIES INSISTEN ON ITS
OEING RECEIVEN, FOR GOON OR FOR EVIL, IN THE
SUWERLATIVE NEGREE OF COMWARISON ONLY.
```

1 ODEGTYZUIQKLMRNAHJSXWVCFBP
2 PYWBCXDQIRKLMOAZJNSEHVUTFG

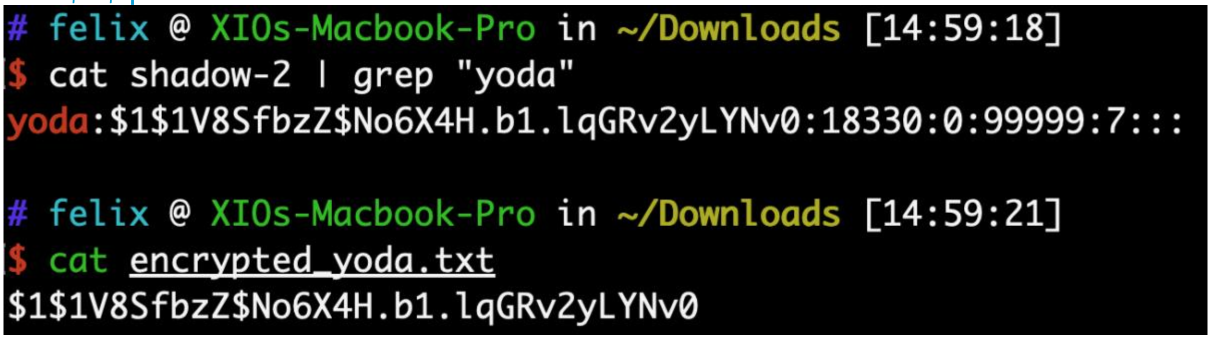
Monoalphabetic Substitution Decoder

★ Alphabetic substitution ciphertext

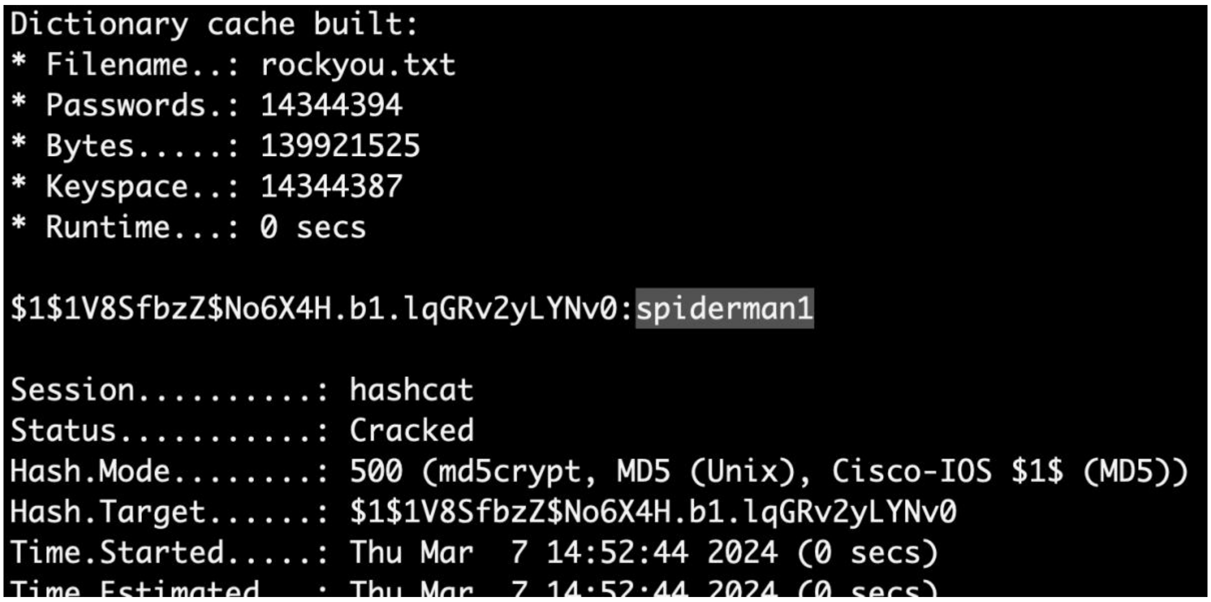
```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
P Y W B C X D Q I R K L M O A Z J N S E H V U T F G
⇒ ODEGTYZUIQKLMRNAHJSXWVCFBP (Original Encryption Alphabet)
⇒ PYWBCXDQIRKLMOAZJNSEHVUTFG (Reciprocal Decryption Alphabet)
I X   C O S   X U T   N T S X   N Y   X I M T S ,
I T   W A S   T H E   O E S T   O F   T I M E S ,
I X   C O S   X U T   C N J S X   N Y   X I M T S ,
I T   W A S   T H E   W O R S T   O F   T I M E S ,
  I X   C O S   X U T   O Z T   N Y   C I S R N M ,
  I T   W A S   T H E   A G E   O F   W I S N O M ,
  I X   C O S   X U T   O Z T   N Y   Y N N L I S U
  I T   W A S   T H E   A G E   O F   F O O L I S H
R T S S ,   I X   C O S   X U T   T C N E U   N Y
N E S S ,   I T   W A S   T H E   E W O C H   O F
N T L I T Y ,   I X   C O S   X U T   T C N E U   N
O E L I E F ,   I T   W A S   T H E   E W O C H   O
Y   I R E J T R W L I X B ,   I X   C O S   X U T
F   I N C R E N U L I T Y ,   I T   W A S   T H E
```

5. (1 point) Download this shadow file Download shadow file(/etc/shadow) from an old Linux system. Crack the password for the user account called "yoda" (the last entry).

- Use hashcat and the rockyou.txt (found under /usr/share/wordlists). You can use other tools if you like.
- The $1$ prefix of the hash means it's MD5



```
# felix @ XIOs-Macbook-Pro in ~/Downloads [14:59:18]
$ cat shadow-2 | grep "yoda"
yoda:$1$1V8SfbzZ$No6X4H.b1.lqGRv2yLYNv0:18330:0:99999:7:::

# felix @ XIOs-Macbook-Pro in ~/Downloads [14:59:21]
$ cat encrypted_yoda.txt
$1$1V8SfbzZ$No6X4H.b1.lqGRv2yLYNv0
```



```
Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344394
* Bytes.....: 139921525
* Keyspace..: 14344387
* Runtime...: 0 secs

$1$1V8SfbzZ$No6X4H.b1.lqGRv2yLYNv0:spiderman1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 500 (md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5))
Hash.Target......: $1$1V8SfbzZ$No6X4H.b1.lqGRv2yLYNv0
Time.Started.....: Thu Mar  7 14:52:44 2024 (0 secs)
Time.Estimated...: Thu Mar  7 14:52:44 2024 (0 secs)
```

First, I copy the middle part (between two ":") as the hash value to be cracked, store it in a new file called encrypted_yoda.txt.

Then use the command "hashcat -m 500 encrypted_yoda.txt rockyou.txt" to crack the password by using the password list "rockyou.txt".

At last, get the password: **spiderman1**