Assignment 0x01 - Intro & Cryptography

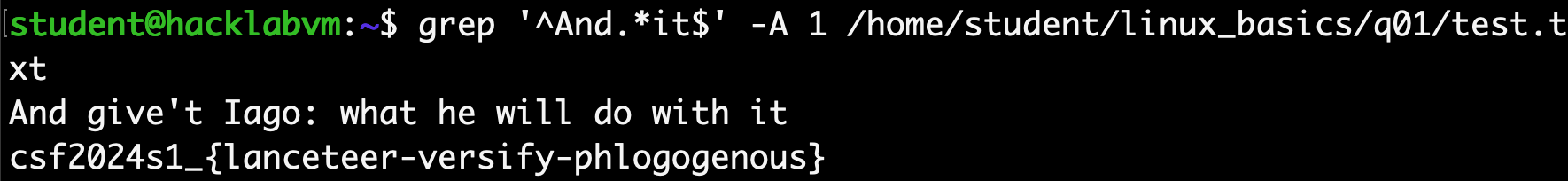
**Part I - Intro**

**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?

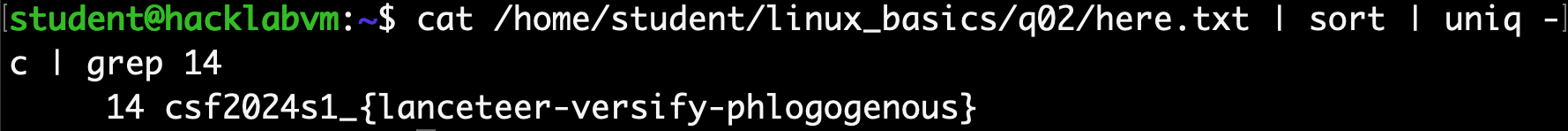


Command: grep '^And.\*it$' -A 1 /home/student/linux\_basics/q01/test.txt

“^” 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}”

1. (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**.

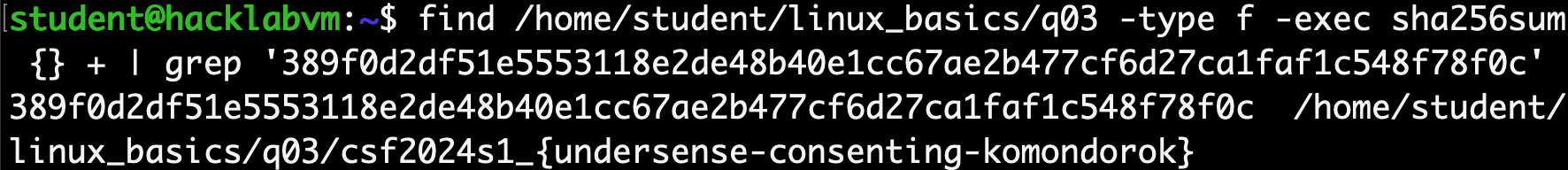


Command: cat /home/student/linux\_basics/q02/here.txt | sort | uniq -c | grep '14'

“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}”

1. (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 ?



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}”

1. (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**.

A screen shot of a computer

Description automatically generated



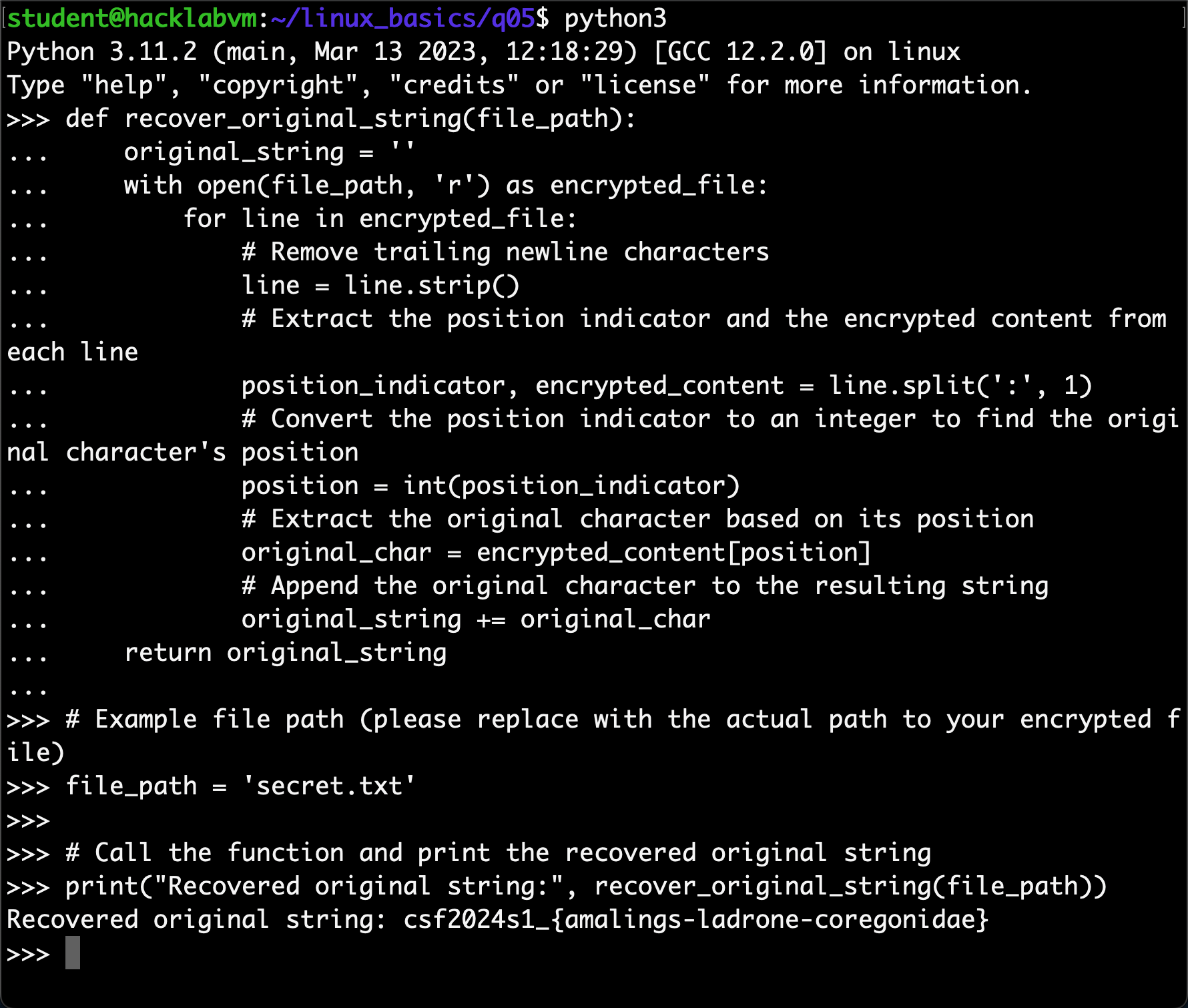
Command & Script: cat words.txt | while read line; do password=$(echo "$line" | sed 's/a/4/g; s/e/3/g; s/i/1/g; s/o/0/g'); echo "Trying: $password"; gpg --quiet --batch --passphrase "$password" -d secret.txt.gpg; done

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}”

1. (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...)



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}”

1. (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.

A computer screen with white text

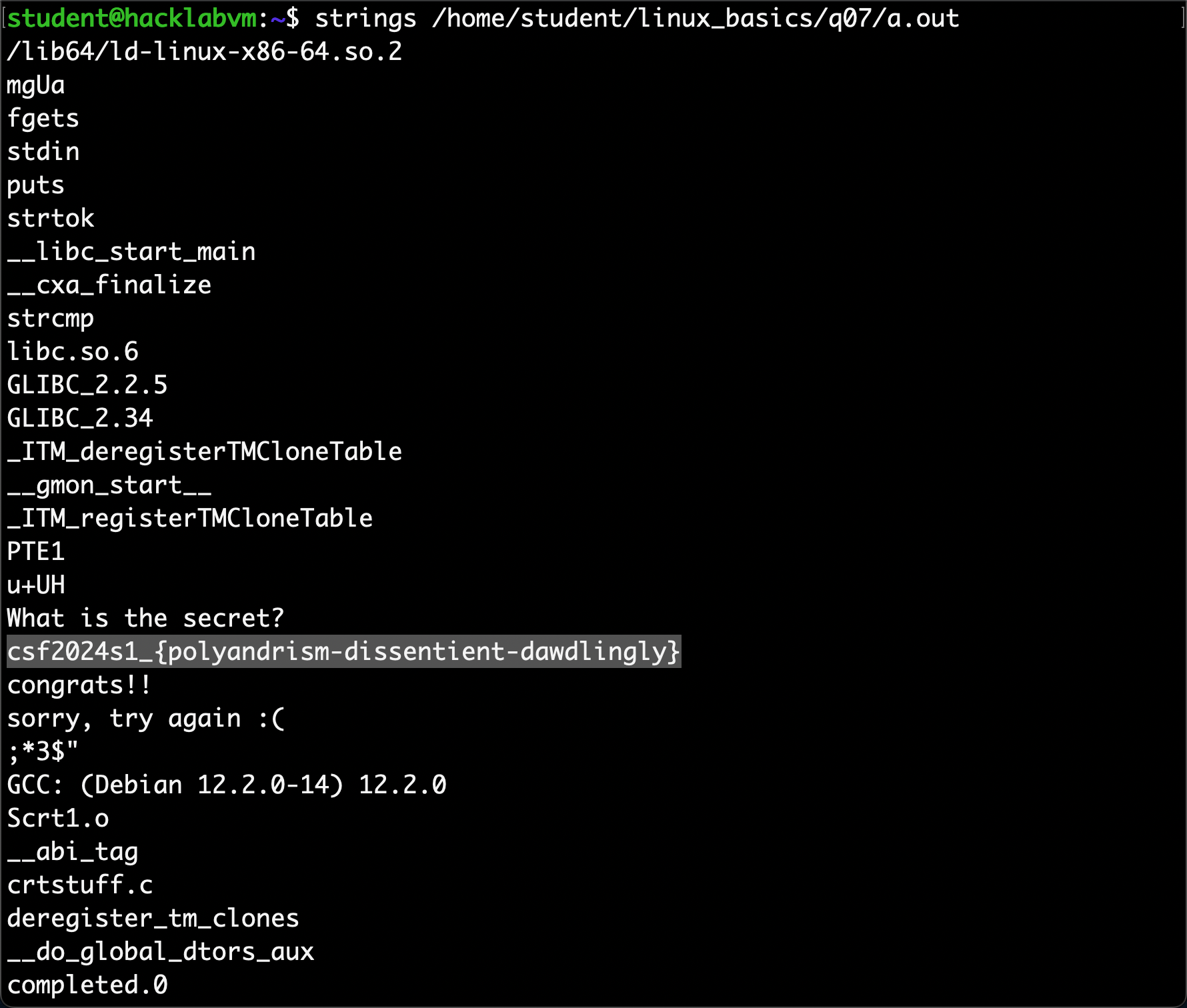
Description automatically generated

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}”

1. (1 point) Execute /home/student/linux\_basics/q07/a.out and try to guess the secret.



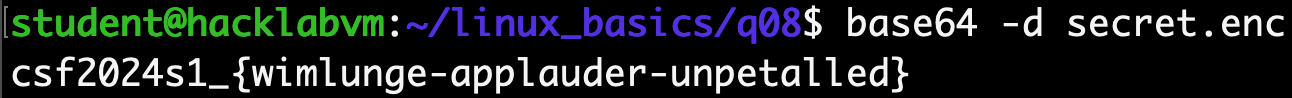
A black background with white text

Description automatically generated

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

The passphrase is: “csf2024s1\_{polyandrism-dissentient-dawdlingly}”

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



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() ”

A screen shot of a computer code

Description automatically generated

The passphrase is: “csf2024s1\_{outtravel-sargassumaishes-monolocular}“

1. (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("0x0D067636BAC6088AD2281E4BFFCACFEFEF9BC1A69FB9E701063DFBAAB436E4C1", 16) # Decryption key

enc = int("0x50543d1e0fda637c109bb32c706dbaec8d8d20ce001cca02f8576a4852a072c9", 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())”A computer screen with numbers and letters

Description automatically generated

The passphrase is: “csf2024s1\_{voteptarius}”

1. (1 point) Find the message hidden inside this [encrypted bitmap image](https://myuni.adelaide.edu.au/courses/95262/files/14689587?wrap=1)[Download encrypted bitmap image](https://myuni.adelaide.edu.au/courses/95262/files/14689587/download?download_frd=1).  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.

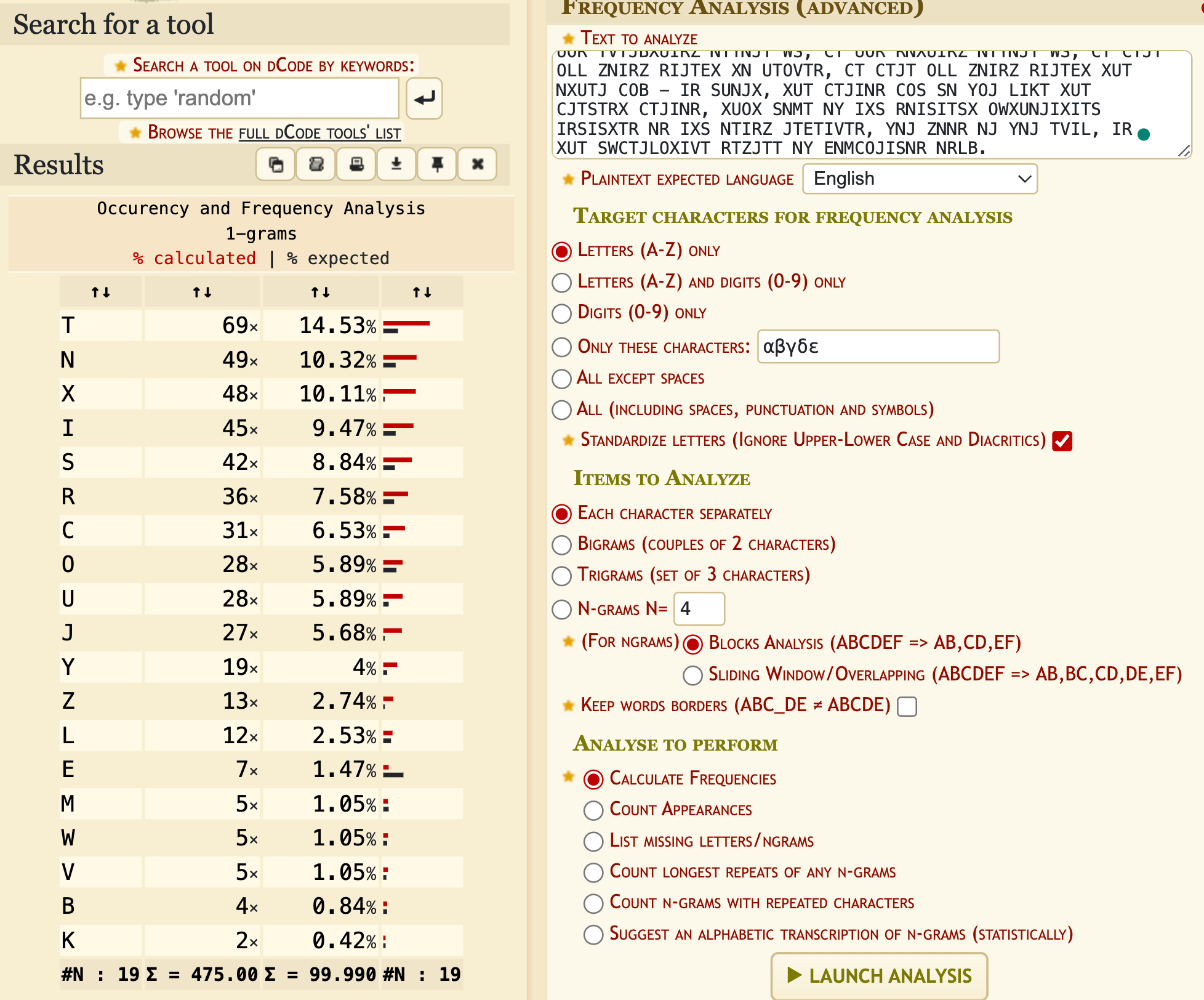
A screen shot of a computer

Description automatically generated

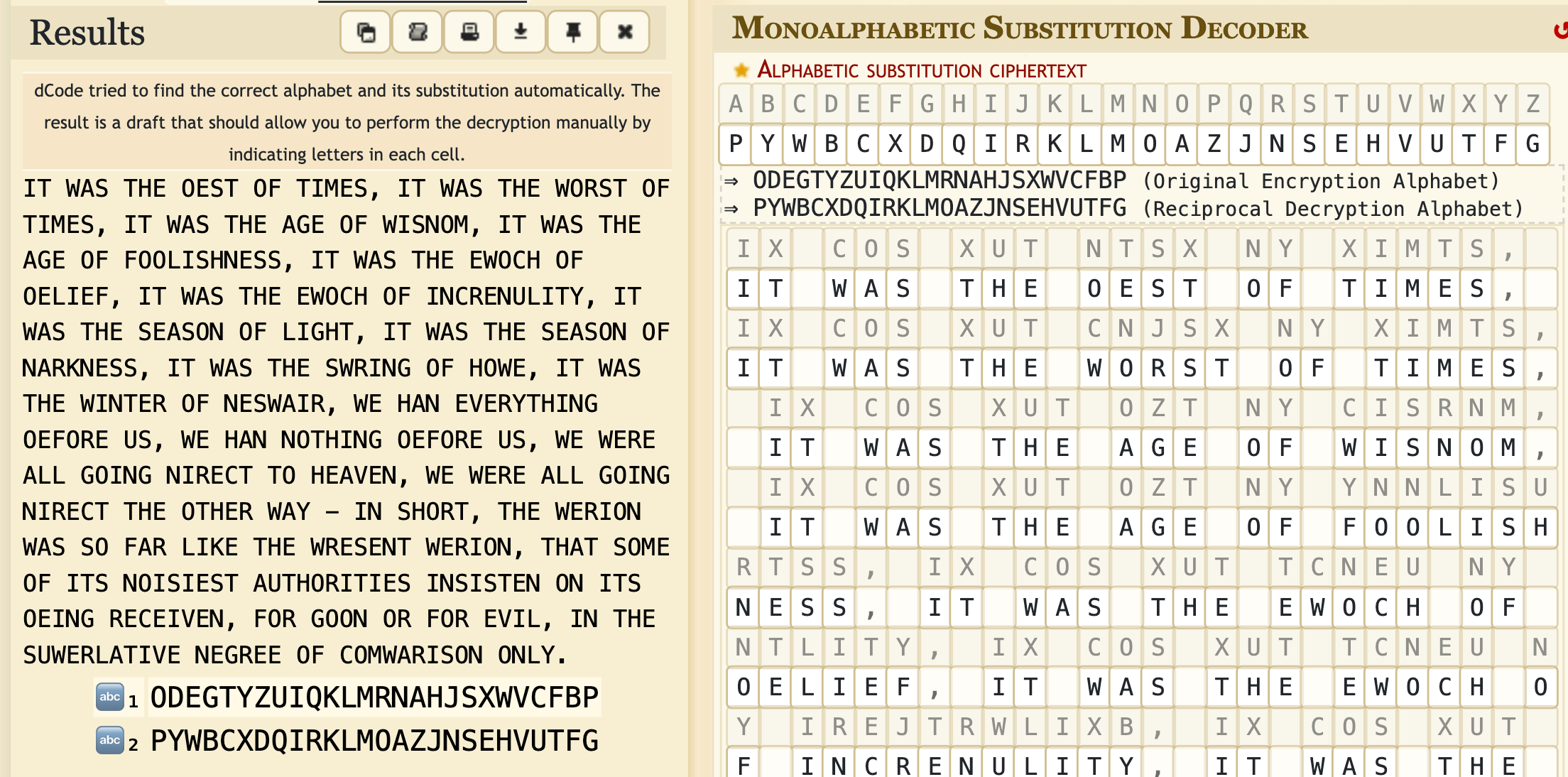
So the answer is 42.

1. (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.



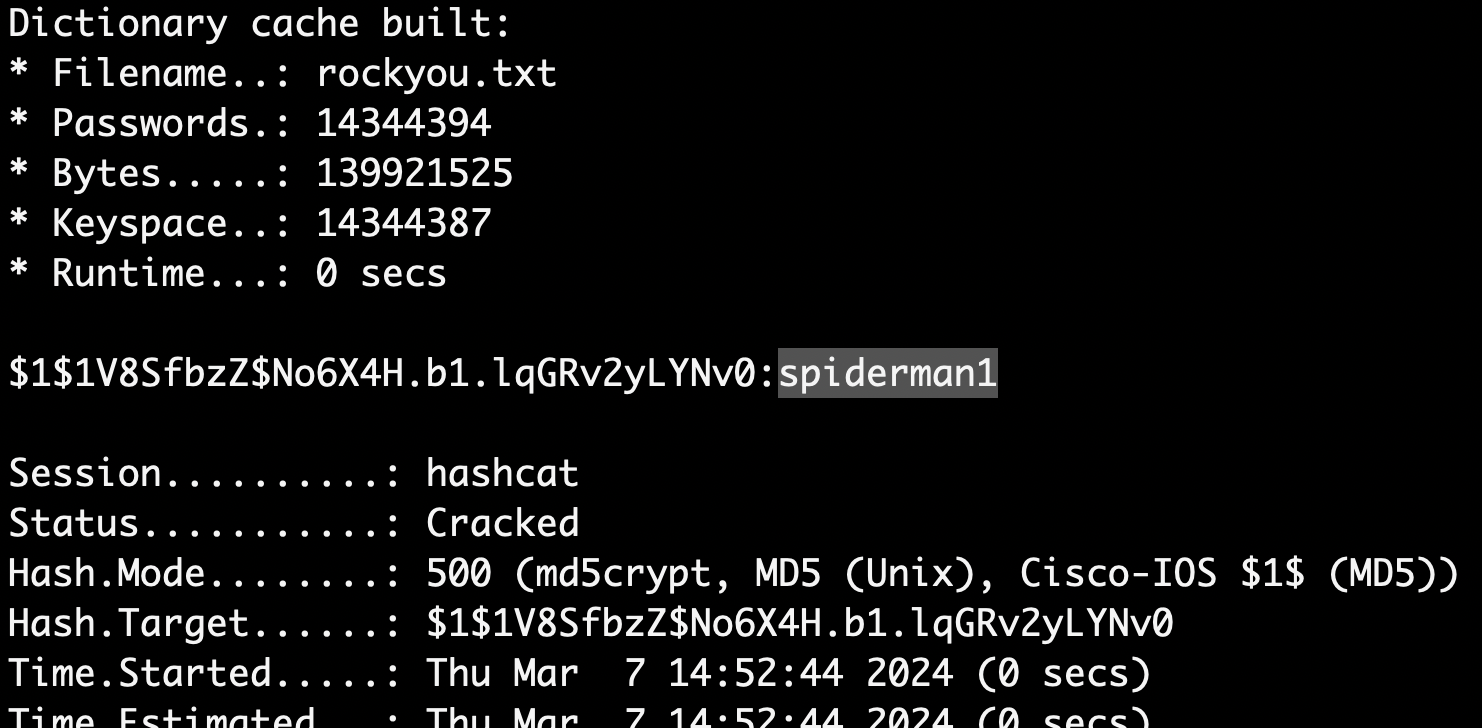
Then use the online tool in the same website to decrypt the text.



1. (1 point) Download this [shadow file](https://myuni.adelaide.edu.au/courses/95262/files/14689590?wrap=1)[Download shadow file](https://myuni.adelaide.edu.au/courses/95262/files/14689590/download?download_frd=1)(/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 MD5A screen shot of a computer code

  Description automatically generated



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**