ACM

WEEK 2 WRITE UP

Task: CRYPTOHACK – general

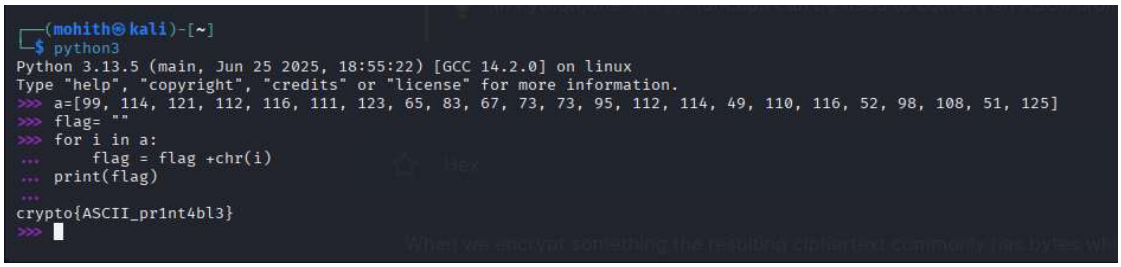
ENCRYPTON:

1.ASCII

Pre-requisites: python latest version (pyhton3)

Method of decryption:

a) Open the terminal and use the command to open python.(“python3”)
b) In python 3 use a code to convert the given values into ascii characters and you will find the flag. The code and output will be as follows



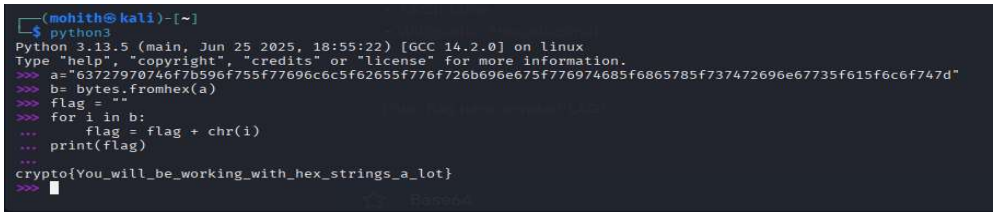
c) So ,the flag is there “crypto{ASCII_pr1nt4bl3}”

2.HEX

Pre-requisites: python latest version(python3)

Method of decryption:

a) Similar to the previous one and this time it is from hex to ascii
b) The code will be as follows



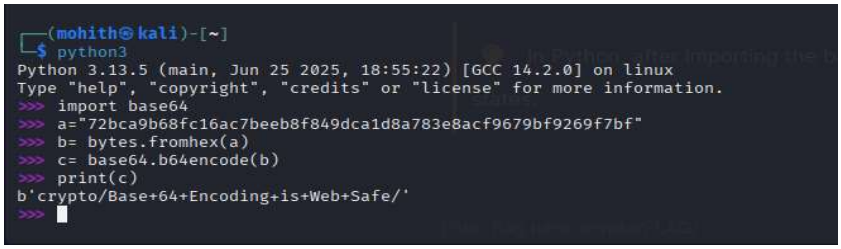
c) So, the flag will be crypto{You_will_be_working_with_hex_strings_a_lot}.

d) Yaa! It's a level up.

3.Base64

Pre-requisites: python latest version(python3)

a) Here it is a encoding into base64 from hexadecimal.
b) The data is "72bca9b68fc16ac7beeb8f849dca1d8a783e8acf9679bf9269f7bf".
c) First we need to convert this to byte data and then to base64 encoding using python program.
d) The code and output are as follows



e) So, the final flag is "crypto/Base+64+Encoding+is+Web+Safe/".

4. Bytes and Big Integers

Pre-requisites: python latest version(python3)

a) This is a decoding method from base 10 to base 16 and then to ascii to get the required flag
b) The data is as follows
"11515195063862318899931685488813747395775516287289682636499652827146372592062 69"
c) The code and the output are as follows
Code:

```
from Crypto.Util.number import long_to_bytes
num =
11515195063862318899931685488813747395775516287289682636499652827146372592062
69
decoded = long_to_bytes(num).decode()
print(decoded)
```

d) The final flag is "crypto{3nc0d1n6_4ll_7h3_w4y_d0wn}"

5.Encoding challenge

XOR:

1.XOR starter:

Pre-requisites: python3

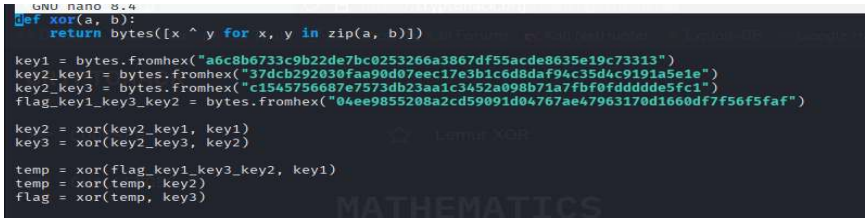
a) We need to convert the word “label” into binary and xor with “13” and again convert to the word using python code.
b) I solved this by creating a file using nano and enter the code and run that program using python3 <filename>
c) The code is

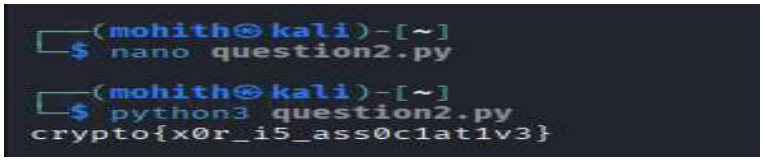
```
text = "label"
key = 13
result = ""
for letter in text:
  ascii_value = ord(letter)
  xored_value = ascii_value ^ key
  new_char = chr(xored_value)
  result += new_char
print("crypto{" + result + "}")
```

d) The output is “crypto{aloha}”.

2.XOR properties:

a) To solve this question, I used a code to solve and the code is as below in nano



b) The output is 

c) So, the final flag is “crypto{x0r_i5_ass0c1at1v3}”.

3.Favourite Byte:

a) To solve this we have 256 numbers for hexadecimal system as it is a byte(8-bits) and there are 2 types of bits(1 and 0).

b) So, the code will be as follows

```
a = bytes.fromhex("73626960647f6b206821204f21254f7d694f7624662065622127234f726927756d")
for i in range(0, 256):
    b = ''
    for x in a:
        b += chr(x ^ i)
    if "crypto" in b:
        print(b)
        break
```

And the output is

```
┌──(mohith㉿kali)-[~]
└─$ nano question4.py

┌──(mohith㉿kali)-[~]
└─$ python3 question4.py
crypto{0x10_15_my_f4v0ur173_by7e}
```

c) The final flag is “crypto{0x10_15_my_f4v0ur173_by7e}”

4. You either know, XOR you don't

a) this also can be solved using python code and the code is stores in another file and executed using python3 <filename>.written the code using github references as my knowledge in python is not upto the mark.

The code is as below

```
GNU nano 8.4
import binascii
from pwn import *

str = "0e0b213f26041e480b26217f27342e175d0e070a3c5b103e2526217f27342e175d0e077e263451150104"
encoded = binascii.unhexlify(str)
print(encoded)
print(encoded)
flag = "crypto{"
key = "myXORkey"
print(xor(encoded,key))
```

b) And the final flag is “crypto{1f_y0u_Kn0w_En0uGH_y0u_Kn0w_1t_4ll}”.

5. Lemur XOR:

a) This is a xor between two images so after so much of searching I found the idea of the commands from the reference in github. The tool used is “gmic”.

b) So, the commands are “gmic -input flag_7ae18c704272532658c10b5faad06d74.png \
-input lemur_ed66878c338e662d3473f0d98eedbd0d.png \
-blend xor \
-o result.png”

c) now the output image is displayed using display “<filename>”.

d) the final flag is “crypto{X0Rly_n0t!}”.

Task: Over the wire-krypton

Level 0-1:

a) Decoded version of “S1JZUFRPTklTR1JFQVQ=” is “KRYPTONISGREAT”.
b) So I tried to connect to port 2231using ssh command given below..
   Ssh -p 2231 krypton1@krypton.labs.overthewire.org
   This will move you to krypton1@bandit
c) Now we will find the file for other levels in /krypton/.

level 1-2:

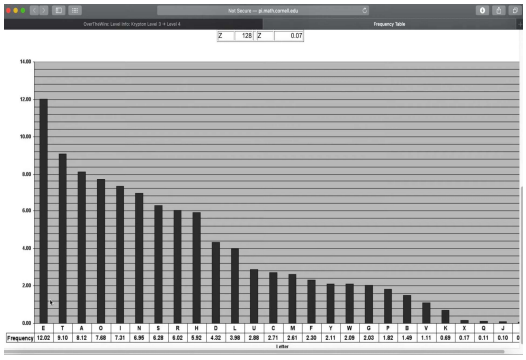
a) Here the password is encrypted by simple rotation in the files of /krypton2/.
b) In /krypton/krypton1 we will find two files and one consists of encryption algorithm and other consists of cipher text
   The cipher text is “YRIRY GJB CNFFJBEQ EBGGRA”.
   The plain text after decryption is “LEVEL TWO PASSWORD ROTTEN”.
c) Now this is used for next level

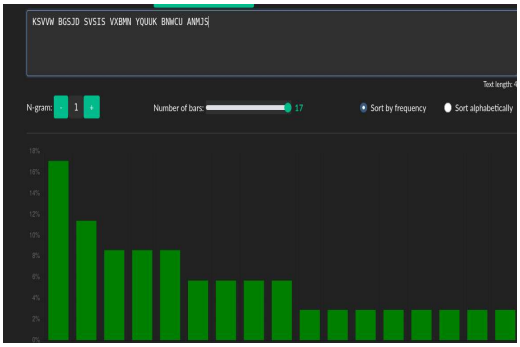
Level 2-3:

a) Now I tried to do in krypton1@bandit but it did not work later on observing I found that we need to do in kryton2@bandit so use exit and change the ssh command accordingly and move to krypton2@bandit
b) Here navigate according to commands given in question so I found some cipher text(“GNGZFGXFEZX”).As the method of encryption is Caesar cipher I tried all 26 possibilities and none made a sense.
c) Now navigate too krytpon/krypton2/ and cat krypton3 we will find some other cipher text(“OMQEMDUEQMEK”) and on decrypting the plaintext is(“CAESARISEASY”).

Level 3-4:

a) Similar navigation as previous one in krytpon3 we will find a file krypton4 and the cipher text is (“KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS”).
b) Using frequency analysis if we solve that we will get the plain text (“WELL DONE THE LEVEL FOUR PASSWORD IS BRUTE”).
   The frequency analysis is done as follows:
   I. the basic frequency analysis table of English alphabets is compared with the frequency analysis of cipher text





now the frequency analysis is done for the file found1.found2,found3 and the for the file krypton4.

So the whole analysis was done in an online frequency analyser like given down

For the file found 1:

```
0   -1.323  in cryptography a caesar cipher also known as a caesars cipher the shift cipher caesars code or caesar shift is one of the simplest and most widely known encryption techniques it is a type of
substitution cipher in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet for example with a shift of a would be replaced by db would
become e and soon the method is named after julius caesar who used it to communicate with his generals the encryption step performed by a caesar cipher is often incorporated as part of more
complex schemes such as the vigenre cipher and still has modern application in the rot system as with all single alphabet substitution ciphers the caesar cipher is easily broken and in practice
offers essentially no communication security shakespeare produced most of his known work between and his early plays were mainly comedies and histories genre she raised to the peak of
sophistication and artistry by the end of the sixteenth century next he wrote mainly tragedies until about including hamlet king lear and macbeth considered some of the finest examples in the
english language in his last phase he wrote tragicomedies also known as romances and collaborated with other playwrights many of his plays were published in editions of varying quality and
accuracy during his lifetime and in two of his former theatrical colleagues published the first folio a collected edition of his dramatic works that included all but two of the plays now
recognised as shakespeares although no attendance records for the period survive most biographers agree that shakespeare was educated at the kings new school in stratford a free school
chartered in about a quarter of a mile from his home grammar schools varied in quality during the elizabethan era but the curriculum was dictated by law throughout england and the school would
have provided an intensive education in latin grammar and the classics at the age of shakespeare married the year old anne hathaway the consistory court of the diocese of worcester issued a
marriage licence on november two of hathaways neighbours posted bonds the next day as surety that there were no impediments to the marriage the couple may have arranged the ceremony in some
haste since the worcester chancellor allowed the marriage banns to be read once instead of the usual three times annes pregnancy could have been the reason for this six months after the
marriage she gave birth to a daughter susanna who was baptised on may twins son hamnet and daughter judith followed almost two years later and were baptised on february ham net died of unknown
causes at the age of and was buried on august after the birth of the twins there are few historical traces of shakespeare until he is mentioned as part of the london theatre scene in because of
this gap scholars refer to the years between and as shakespeares lost years biographers attempting to account for this period have reported many apocryphal stories nicholas rowe shakespeares
first biographer recounted a stratford legend that shakespeare fled the town for london to escape prosecution for deer poaching another eighteenth century story has shakespeare starting his
theatrical career minding the horses of theatre patrons in london john aubrey reported that shakespeare had been a country school master some twentieth century scholars have suggested that
shakespeare may have been employed as a schoolmaster by alexander hoghton of lancashire a catholic landowner who named a certain william shakesh aftein his will no evidence substantiates such
stories other than hearsay collected after his death whereupon le grand arose with a grave and stately air and brought me the beetle from a glass case in which it was enclosed it was a
beautiful scarabaeus and at that time unknown to naturalists of course a great prize in a scientific point of view there were two round black spots near one extremity of the back and a long one
near the other the scales were exceedingly hard and glossy with all the appearance of burnished gold the weight of the insect was very remarkable and taking all things into consideration i
could hardly blame jupiter for his opinion respecting it well done the level four password is brute
```

I have entered all the cipher text from all the files in the same place so that the krypton4 ‘s cipher text is at the last line .

The password for level 4 is “BRUTE”.

Level 4-5:

a) this level is about vigenere cipher . Login using previous credentials and in this we will find a file named krypton5 and it consists of cipher text (“HCIKV RJOX”)and on using the data given the plain text is (“CLEAR TEXT”).
b) Given the length of key is 6 so using vigenere cipher the pass word is broken from the cipher text in first 2 files (only few data was take for calculations) and solvation was done..

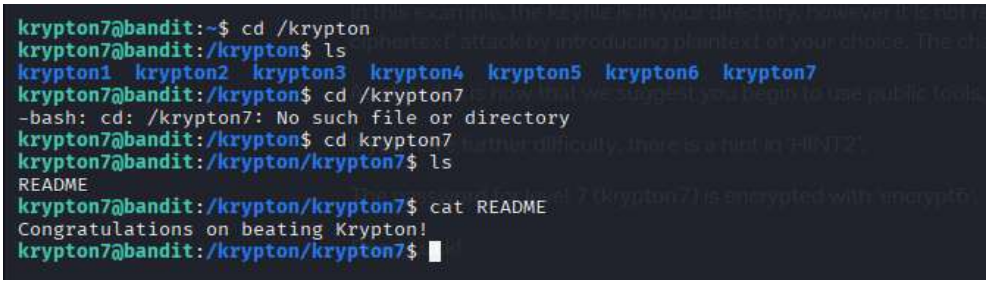
Level 5-6:

a) Move to krypton5 using previous plain text
b) Now we have the cipher test “BELOS Z” and the encryption method is polyalphabetic cipher so I tried it with all method and found that vigenere cipher is the encryption method.
c) And the plain text is “RANDOM”

Level 6-7:

a) This is a stream cipher encryption that uses bit by bit encryption
b) And the encryption was done by 8 bit LFSR.
c) So the cipher text is “PNUKLYLWRQKGKBE” and the decryption was done using stream cipher.
d) The plain text is “LFSRISNOTRANDOM”
e) This is a decryption using the repeated key vigenere cipher
f) I created a file with some same letter repeated for few times and found the key that is repeating in order an found the cipher text in krypton7 file.
g) So upon decryption and using it for the next login I found the following this

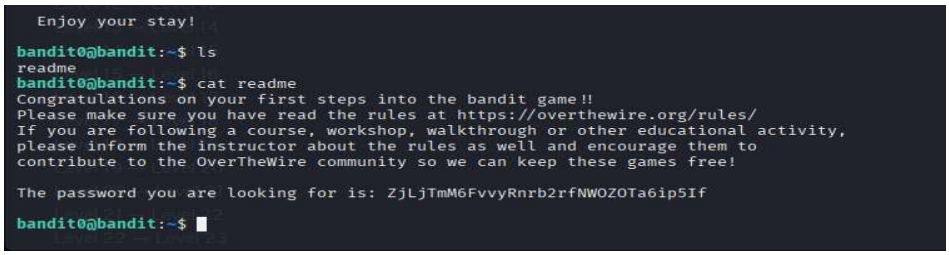
PROOF OF COMPLETION:



Task: OVER THE WIRE – BANDIT:

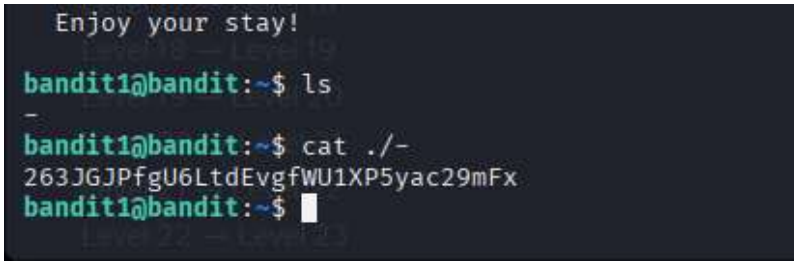
Level 0:

a) This is an introduction level that gives the password for next level and that should be done after logging into the bandit shell using ssh command



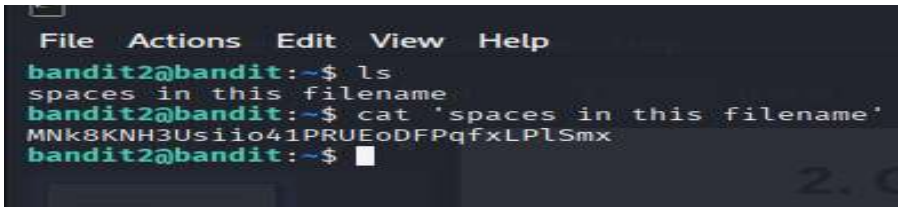
Level 0-1:

a) Next log into the bandit 1 using the pwd found in level 0
b) Commands used: ls , cat ./<filename>



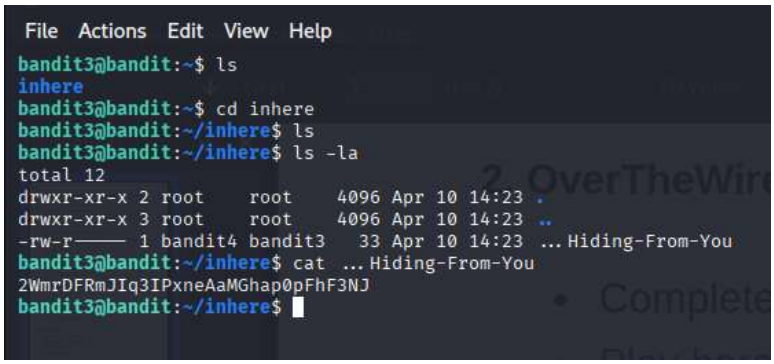
Level 1-2:

a) Next one will be done using the commands in the picture and the pwd for next login is “MNk8KNH3Usiio41PRUEoDFPqfxLPlSmx”



Level 2-3:

a) This level has a hidden file that is found using the command " ls -la". The information that is found in bandit3 is "2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ".



Level 3-4:

a) In this level we have 9 files in the directory and the required data is found in 7th file.

We can also filter using the type of data "file ./*".

And in the human readable file we can access the data.

Level 4-5:

a) Here we need to classify using the "find" command.

b) In the find manual we found that we can filter the file using “find -<file specifications>.

c) That can be done as below

```
File  Actions  Edit  View  Help
bandit5@bandit:~$ ls
inhere
bandit5@bandit:~$ cd inhere
bandit5@bandit:~/inhere$ ls
maybehere00  maybehere02  maybehere04  maybehere06  maybehere08  maybehere10  maybehere12  maybehere14  maybehere16  maybehere18
maybehere01  maybehere03  maybehere05  maybehere07  maybehere09  maybehere11  maybehere13  maybehere15  maybehere17  maybehere19
bandit5@bandit:~/inhere$ find -readable -size 1033c ! -executable
./maybehere07/.file2
bandit5@bandit:~/inhere$ cd /maybehere07
-bash: cd: /maybehere07: No such file or directory
bandit5@bandit:~/inhere$ cd maybehere07
bandit5@bandit:~/inhere/maybehere07$ ls
-file1  -file2  -file3  spaces file1  spaces file2  spaces file3
bandit5@bandit:~/inhere/maybehere07$ cat file2
cat: file2: No such file or directory
bandit5@bandit:~/inhere/maybehere07$ cat .file2
HWasnPhtq9AVKe0dmk45nxy20cvUa6EG
```

Level 5-6:

a) In the question they gave that the owner is bandit7

b) So I used the command “find -user bandit7 -group bandit6 size 33c”.

c) Here I tried It with 3 combinations from “-users bandit7” & “ -group bandit6” and also based on the size.

d) The first two tries gave so many errors

e) In third attempt, I found the path for the bandit7 password.

```
find: ‘./var/spool/cron/crontabs’: Permission denied
find: ‘./var/spool/bandit24’: Permission denied
find: ‘./var/log/chrony’: Permission denied
find: ‘./var/log/amazon’: Permission denied
find: ‘./var/log/unattended-upgrades’: Permission denied
find: ‘./var/log/private’: Permission denied
find: ‘./var/tmp’: Permission denied
find: ‘./var/lib/udisks2’: Permission denied
find: ‘./var/lib/update-notifier/package-data-downloads/partial’: Permission denied
find: ‘./var/lib/polkit-1’: Permission denied
./var/lib/dpkg/info/bandit7.password
find: ‘./var/lib/apt/lists/partial’: Permission denied
find: ‘./var/lib/chrony’: Permission denied
find: ‘./var/lib/amazon’: Permission denied
find: ‘./var/lib/ubuntu-advantage/apt-esm/var/lib/apt/lists/partial’: Permission denied
find: ‘./var/lib/snapd/cookie’: Permission denied
find: ‘./var/lib/snapd/void’: Permission denied
find: ‘./var/lib/private’: Permission denied
find: ‘./drifter/drifter14_src/axTLS’: Permission denied
find: ‘./tmp’: Permission denied
bandit6@bandit:/$ cd ./var/lib/dpkg/info/bandit7.password
-bash: cd: ./var/lib/dpkg/info/bandit7.password: Not a directory
bandit6@bandit:/$ cat ./var/lib/dpkg/info/bandit7.password
morbNTDkSW6jIlUc0ymOdMaLnOlFVAaj
bandit6@bandit:/$
```

Level 6-7:

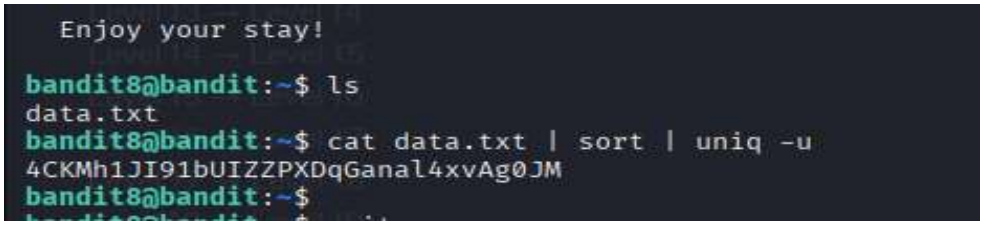
a) We will have a data.txt and that has so many words so according to the question we will search for the word millionth.

```
  Enjoy your stay!

bandit7@bandit:~$ ls
data.txt
bandit7@bandit:~$ cat data.txt | grep millionth
millionth       dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc
bandit7@bandit:~$
```

So , the data is here fro next login.

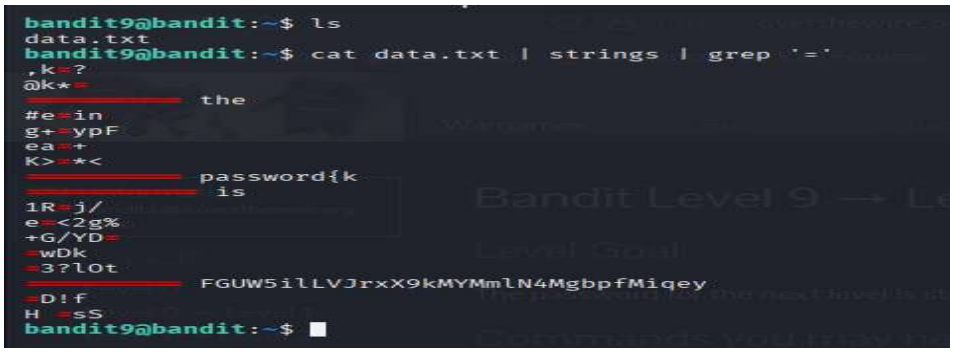
Level 7-8:

a) For this we will find the unique characters using the command “cat data.txt | sort | uniq -u”
b) We will find the password for next level.



Level 8-9:

a) This level we will sort the readable data using strings command and the process goes like this.



Level 9-10:

a) This level has a base64 encrypted data and found data in data.txt file and solved using base64 decoder.

The decoded text is “The password is dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr”.

Level 10-11:

a) This level the data is encrypted by rotating the letters by 13 positions.
b) The decryption will be done using a rot13 decrypter.
c) The plain text is as follows “The password is 7x16WNeHIi5YkIhWsfFIqoognUTyj9Q4”.

```
  Enjoy your stay!

bandit11@bandit:~$ ls
data.txt
bandit11@bandit:~$ cat data.txt
Gur cnffjbeq vf 7k16JArUVv5LxVuJfsSVdbbtaHGlw9D4
bandit11@bandit:~$
```

Level 11-12:

a) This round is of continuous decompression of the file “data.txt”
b) This involves in 3 types of decompressions they are 1) gzip 2) bzip2 and 3) tar things .
c) So, now we have to move the file from the present type to the new type using “mv” and the final password is “FO5dwFsc0cbaliH0h8J2eUks2vdTDwAn”.
The proof of completion:

```
bandit12@bandit:~$ ls
data.txt
bandit12@bandit:~$ mkdir /tmp/hunter
bandit12@bandit:~$ cp data.txt /tmp/hunter
bandit12@bandit:~$ cd /tmp/hunter
bandit12@bandit:/tmp/hunter$ ls
data.txt
bandit12@bandit:/tmp/hunter$ xxd -r data.txt > data
bandit12@bandit:/tmp/hunter$ ls
data  data.txt
bandit12@bandit:/tmp/hunter$ file data
data: gzip compressed data, was "data2.bin", last modified: Thu Apr 10 14:22:57 2025, max compression, from Unix, or
iginal size modulo 2^32 585
bandit12@bandit:/tmp/hunter$ mv data file.gz
bandit12@bandit:/tmp/hunter$ gzip -d file.gz
bandit12@bandit:/tmp/hunter$ ls
data.txt  file
bandit12@bandit:/tmp/hunter$ file file
file: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/hunter$ mv file file.bz2
bandit12@bandit:/tmp/hunter$ ls
data.txt  file.bz2
bandit12@bandit:/tmp/hunter$ bzip2 -d file.bz2
bandit12@bandit:/tmp/hunter$ ls
data.txt  file
bandit12@bandit:/tmp/hunter$ file file
file: gzip compressed data, was "data4.bin", last modified: Thu Apr 10 14:22:57 2025, max compression, from Unix, or
iginal size modulo 2^32 20480
bandit12@bandit:/tmp/hunter$ mv file file.gz
bandit12@bandit:/tmp/hunter$ ls
data.txt  file.gz
bandit12@bandit:/tmp/hunter$ gzip -d file.gz
bandit12@bandit:/tmp/hunter$ ls
data.txt  file
bandit12@bandit:/tmp/hunter$ file file
file: POSIX tar archive (GNU)
bandit12@bandit:/tmp/hunter$ mv file file.tar
bandit12@bandit:/tmp/hunter$ tar xf file.tar
bandit12@bandit:/tmp/hunter$ ls
data5.bin  data.txt  file.tar
bandit12@bandit:/tmp/hunter$ file data5.bin
data5.bin: POSIX tar archive (GNU)
bandit12@bandit:/tmp/hunter$ rm file.tar
bandit12@bandit:/tmp/hunter$ ls
data5.bin  data.txt
bandit12@bandit:/tmp/hunter$ rm data.txt
bandit12@bandit:/tmp/hunter$ ls
data5.bin
bandit12@bandit:/tmp/hunter$ file data5.bin
data5.bin: POSIX tar archive (GNU)
bandit12@bandit:/tmp/hunter$ mv data5.bin data.tar
bandit12@bandit:/tmp/hunter$ tar xf data.tar
```

```
bandit12@bandit:/tmp/hunter$ ls
data6.bin  data.tar
bandit12@bandit:/tmp/hunter$ file data6.bin
data6.bin: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/hunter$ mv data6.bin data.bz2
bandit12@bandit:/tmp/hunter$ bzip -d data.bz2
Command 'bzip' not found, but there are 21 similar ones.
bandit12@bandit:/tmp/hunter$ bzip2 -d data.bz2
bandit12@bandit:/tmp/hunter$ ls
data  data.tar
bandit12@bandit:/tmp/hunter$ file data
data: POSIX tar archive (GNU)
bandit12@bandit:/tmp/hunter$ mv data data.tar
bandit12@bandit:/tmp/hunter$ ls
data.tar
bandit12@bandit:/tmp/hunter$ tar xf data.tar
bandit12@bandit:/tmp/hunter$ ls
data8.bin  data.tar
bandit12@bandit:/tmp/hunter$ file data8.bin
data8.bin: gzip compressed data, was "data9.bin", last modified: Thu Apr 10 14:22:57 2025, max compression, from Uni
x, original size modulo 2^32 49
bandit12@bandit:/tmp/hunter$ mv data8.bin data.gz
bandit12@bandit:/tmp/hunter$ gzip -d data.gz
bandit12@bandit:/tmp/hunter$ ls
data  data.tar
bandit12@bandit:/tmp/hunter$ file data
data: ASCII text
bandit12@bandit:/tmp/hunter$ cat data.txt
cat: data.txt: No such file or directory
bandit12@bandit:/tmp/hunter$ cat data
The password is FO5dwFsc0cbaIiH0h8J2eUks2vdTDwAn
bandit12@bandit:/tmp/hunter$
```

Level 12-13:

a) This level we need to move to bandit 13 and find the private key and copy it the exit bandit 13

b) This level gives the ssh private key for the next level

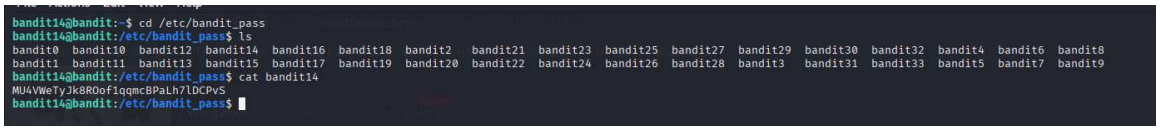
```
  Enjoy your stay!

bandit13@bandit:~$ ls
sshkey.private
bandit13@bandit:~$ cat sshkey.private
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxkkOE83W2cOT7IWhFc9aPaaQmQDdgzuXCv+ppZHa++buSkN+
gg0tcr7Fw8NLGa5+Uzec2rEg0WmeevB13AIoYp0MZyETq46t+jk9puNwZwIt9XgB
ZufGtZEwWbFWw/vVLNwOXBe4UWStGRWzgPpEeSv5Tb1VjLZIBdGphTIK22Amz6Zb
ThMsiMnyJafEwJ/T8PQO3myS91vUHEuoOMAzoUID4kN0MEZ3+XahyK0HJVq68KsV
ObefXG1vvA3GAJ29kxJaqvRfgYnqZryWN7w3CHjNU4c/2Jkp+n8L0SnxaNA+WYA7
jiPyTF0is8uzMlYQ4l1Lzh/8/MpvhCQF8r22dwIDAQABAoIBAQC6dWBjhyEOzjeA
J3j/RWmap9M5zfJ/wb2bfidNpwbB8rsJ4sZIDZQ7XuIh4LfygoAQSS+bBw3RXvzE
pvJt3SmU8hIDuLsCjL1VnBY5pY7Bju8g8aR/3FyjyNAqx/TLfzlLYfOu7i9Jet67
xAh0tONG/u8FB5I3LAI2Vp6OviwvdWeC4nOxCthldpuPKNLA8rmMMVRTKQ+7T2VS
nXmwYckKUcUgzoVSpiNZaS0zUDypdpy2+tRH3MQa5kqN1YKjvF8RC47woOYCktsD
o3FFpGNFec9Taa3Msy+DfQQhHKZFKIL3bJDONtmrVvtYK40/yeU4aZ/HA2DQzwhe
ol1AfiEhAoGBAOnVjosBkm7sblK+n4IEwPxs8sOmhPnTDUy5WGrpSCrXOmsVIBUf
laL3ZGLx3xCIwtCnEucB9DvN2HZkupc/h6hTKUYLqXuyLD8njTrbRhLgbC9QrKrS
M1F2fSTxVqPtZDlDMwjNR04xHA/fKh8bXXyTMqOHNJTHHNhbh3McdURjAoGBANkU
1hqfnw7+aXncJ9bjysr1ZWbqOE5Nd8AFgfwaKuGTTVX2NsUQnCMWdOp+wFak40JH
PKWkJNdBG+ex0H9JNQsTK3X5PBMAS8AfX0GrKeuwKWA6erytVTqjOfLYcdp5+z9s
8DtVCxDuVsM+i4X8UqIGOlvGbtKEVokHPFXP1q/dAoGAcHg5YX7WEehCgCYTzpO+
xysX8ScM2qS6xuZ3MqUWAxUWkh7NGZvhe0sGy9iOdANzwKw7mUUFViaCMR/t54W1
GC83sOs3D7n5Mj8×3NdO8xFit7dT9a245TvaoYQ7KgmqpSg/ScKCw4c3eiLava+J
3btnJeSIU+8ZXq9XjPRpKwUCgYA7z6LiOQKxNeXH3qHXcnHok855maUj5fJNpPbY
iDkyZ8ySF8GlcFsky8Yw6fWCqfG3zDrohJ5l9JmEsBh7SadkwsZhvecQcS9t4vby
9/8X4jS0P8ibfcKS4nBP+dT81kkkg5Z5MohXBORA7VWx+ACohcDEkprsQ+w32xeD
qT1EvQKBgQDKm8ws2ByvSUVs9GjTilCajFqLJ0eVYzRPaY6f++Gv/UVfAPV4c+S0
kAWpXbv5tbkkzbS0eaLPTKgLzavXtQoTtKwrjpolHKIHUz6Wu+n4abfAIRFubOdN
/+aLoRQ0yBDRbdXMsZN/jvY44eM+xRLdRVyMmdPtP8belRi2E2aEzA==
-----END RSA PRIVATE KEY-----
bandit13@bandit:~$
```

c) Now create a file and login using the command “ssh bandit14@bandit.labs.overthewire.org -p 2220 -i private.key”.

d) If it doesn’t login modify the file permission to 700 using chmod

Level 13-14:

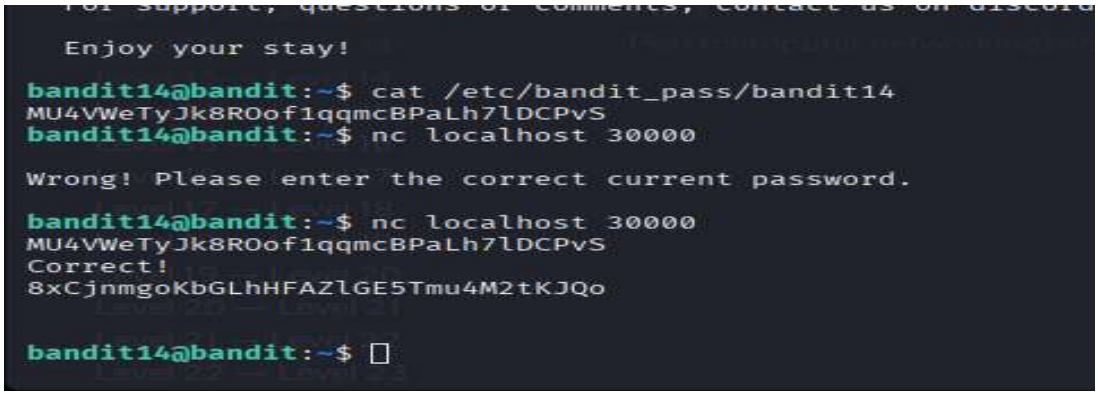
a) We will login into bandit14 and according to instructions we will navigate to /etc/bandit_pass/bandit14.



Level 14-15:

a) We will find the required data in the localhost 30000 in bandit 14 only.

o



b) So the data is obtained by entering the present password.

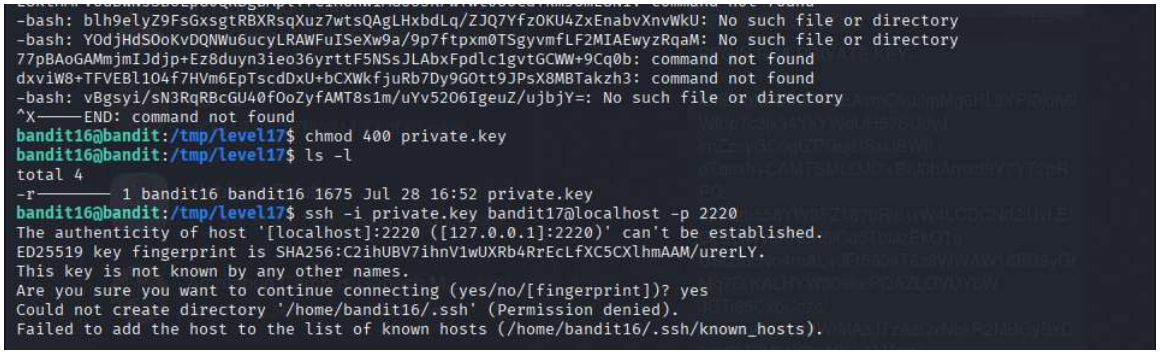
Level 15-16:

a) For this level we login into bandit 15 from the above password and here we will use openssl and s_client and connect to port 30001 using the below command " openssl s_client -connect localhost-30001".

b) "kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx"

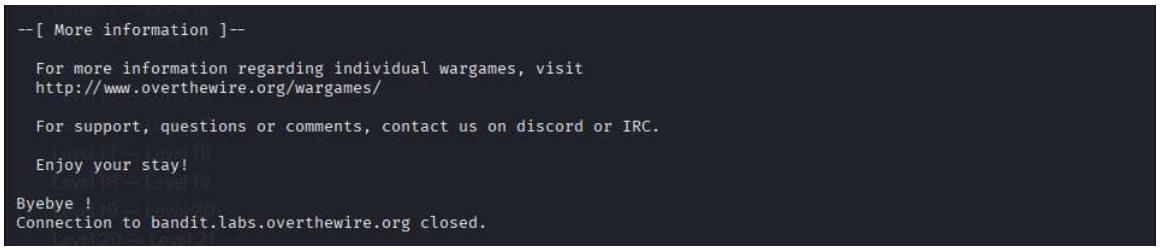
Level 16-17:

a) For this level we need to scanthe proths that are open and need to find the port that is listening to our bandit sever and enter that localhost.
b) Enter the prvious level password and there we will receive a rsa key.
c) Use that key to opn and find the password in /etc/bandit_pass/bandit17
d) And login to bandit 17 using that password.



Level 17-18:

a) We wil find two file and if we find the difference between two files we will get two passwords and we will find the correct one and see bye bye! at last

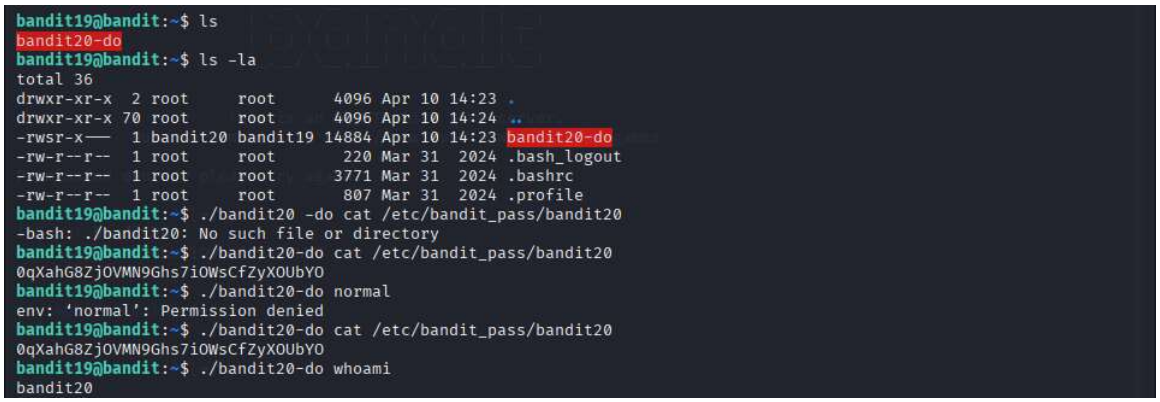


Level 18-19:

a) To login into this level first we need to get the password from bandit18 using the command “ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme”.

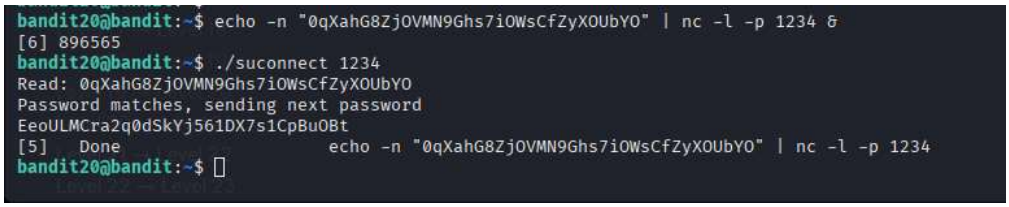
Level 19-20:

a) Here we need use “./bandit20-d cat /etc/bandit_pass/bandit20” and we will find the key here itself
b) Later we can find the file using “./bandit20-do whoami”



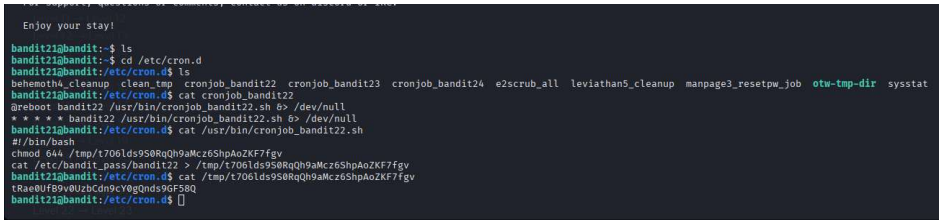
Level 20-21:

a) This level I have use “nc” to solve and then suconnect as the things given in question is are not working.



Level 21-22:

a) This level we need to use cron as cron is found in /etc/cron.d.
b) And the whole thing went as below in picture.



Level 22-23:

a) Here we will use the path and find a data and using that we will find the necessary data.

```
bandit22@bandit:/etc/cron.d$ cat /usr/bin/cronjob_bandit23.sh
#!/bin/bash

myname=$(whoami)
mytarget=$(echo I am user $myname | md5sum | cut -d ' ' -f 1)

echo "Copying passwordfile /etc/bandit_pass/$myname to /tmp/$mytarget"

cat /etc/bandit_pass/$myname > /tmp/$mytarget
bandit22@bandit:/etc/cron.d$ echo "I am user bandit23" | md5sum | cut -d ' ' -f1
8ca319486bfbbc3663ea0fbe81326349
bandit22@bandit:/etc/cron.d$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349
0Zf11ioIjMVN551jX3CmStKLYqjk54Ga
bandit22@bandit:/etc/cron.d$ exit
```

Level 23-24:

a) This we need to create some directories and some files and modify the file permissions to sole this level.

```
for i in * .*;
do
    if [ "$i" != "." -a "$i" != ".." ];
    then
        echo "Handling $i"
        owner="$(stat --format "%U" ./$i)"
        if [ "${owner}" = "bandit23" ]; then
            timeout -s 9 60 ./$i
        fi
        rm -f ./$i
    fi
done

bandit23@bandit:/etc/cron.d$ mkdir /tmp/last
bandit23@bandit:/etc/cron.d$ cd /tmp/last
bandit23@bandit:/tmp/last$ nano last.py
Unable to create directory /home/bandit23/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

bandit23@bandit:/tmp/last$ ls
last.py
bandit23@bandit:/tmp/last$ cano last.py
Command 'cano' not found, did you mean:
  command 'cado' from deb cado (0.9.6-1build1)
  command 'canu' from deb canu (2.0+dfsg-2)
  command 'nano' from deb nano (7.2-2ubuntu0.1)
Try: apt install <deb name>
bandit23@bandit:/tmp/last$ nano last.py
Unable to create directory /home/bandit23/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

bandit23@bandit:/tmp/last$ ls
last.py  last.sh
bandit23@bandit:/tmp/last$ ls -l last.sh
-rw-rw-r-- 1 bandit23 bandit23 68 Jul 29 03:44 last.sh
bandit23@bandit:/tmp/last$ chmod +x last.sh
bandit23@bandit:/tmp/last$ ls -la
total 32
drwxrwxr-x   2 bandit23 bandit23  4096 Jul 29 03:44 .
drwxrwx-wt 298 root     root     20480 Jul 29 03:46 ..
-rw-rw-r--   1 bandit23 bandit23    69 Jul 29 03:43 last.py
-rwxrwxr-x   1 bandit23 bandit23    68 Jul 29 03:44 last.sh
bandit23@bandit:/tmp/last$ chmod +w .
bandit23@bandit:/tmp/last$ chmod 777 .
bandit23@bandit:/tmp/last$ cp last.sh /var/spool/bandit24/foo
bandit23@bandit:/tmp/last$ cp last.sh /var/spool/bandit24/
cp: cannot create regular file '/var/spool/bandit24/last.sh': Operation not permitted
bandit23@bandit:/tmp/last$ ls
last.py  last.sh  password.txt
bandit23@bandit:/tmp/last$ cat password.txt
gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8
bandit23@bandit:/tmp/last$
```

Level 24-25:

a) We will use the file list.txt in brute. We will find password using the command “cat list.txt | nc localhost 30002”.

b) That gives the password

```
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Correct!
The password of user bandit25 is iCi86ttT4KSNe1armKiwbQNmB3YJP3q4

bandit24@bandit:/tmp/brute$
```

Task – 1:

1.over the wire – Natas

Level 0:

the credentials for next level are

user name: natas0

password: natas0

Level 0--1:

i) Open the url given on the screen and use the user name and password to get into that page.

ii) Right click and inspect the page. So that you will get the user name and pass word for next page.as below

```
<!--The password for natas1 is 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq -->
```

iii) Now this continuous with different restrictions like right click is blocked for next page and etc....

iv) The credentials for next level are –
User name: natas 1
Password: 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq

Level 1--2:

Here right clicking is blocked so use the more option in the browser to find the developer tools where we will find the pass word for next level.

```
" You can find the password for the next level on this page, but
rightclicking has been blocked! "
<!--The password for natas2 is TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI --> == $0
```

User name: natas 2
Password: TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI

Level 2--3:

Here in the developer space we will find a thing as below in the body part .

```
" There is nothing on this page "
<img src="files/pixel.png"> == $0
```

Now open that files/pixel.png and go to files place to find user.txt file which consists of following thing

```
# username:password
alice:BYNdCesZqW
bob:jw2ueICLvT
charlie:G5vCxkVV3m
natas3:3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH
eve:zo4mJWyNj2
mallory:9urtcpzBmH
```

User name: natas3
Password: 3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH

Level 3-4:

This time it was little complicated to find the credentials

Inspect the page and we will find this

```
" There is nothing on this page "
<!-- No more information leaks!! Not even Google will find it this time... -
-> == $0
```

So on detailed analysis I found that we need to search for a directory of google
That is http://natas3.natas.labs.overthewire.org/robots.txt
We will find these details

```
User-agent: *
Disallow: /s3cr3t/
```

But we need to go to /s3cr3t/ i.e http://natas3.natas.labs.overthewire.org/s3cr3t/
So that we get the below page

Index of /s3cr3t

Name	Last modified	Size	Description
Parent Directory		-	
users.txt	2025-04-10 14:18	40	

Apache/2.4.58 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80

Where the user .txt consists of login credentials for next level
Username: natas4
password: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ

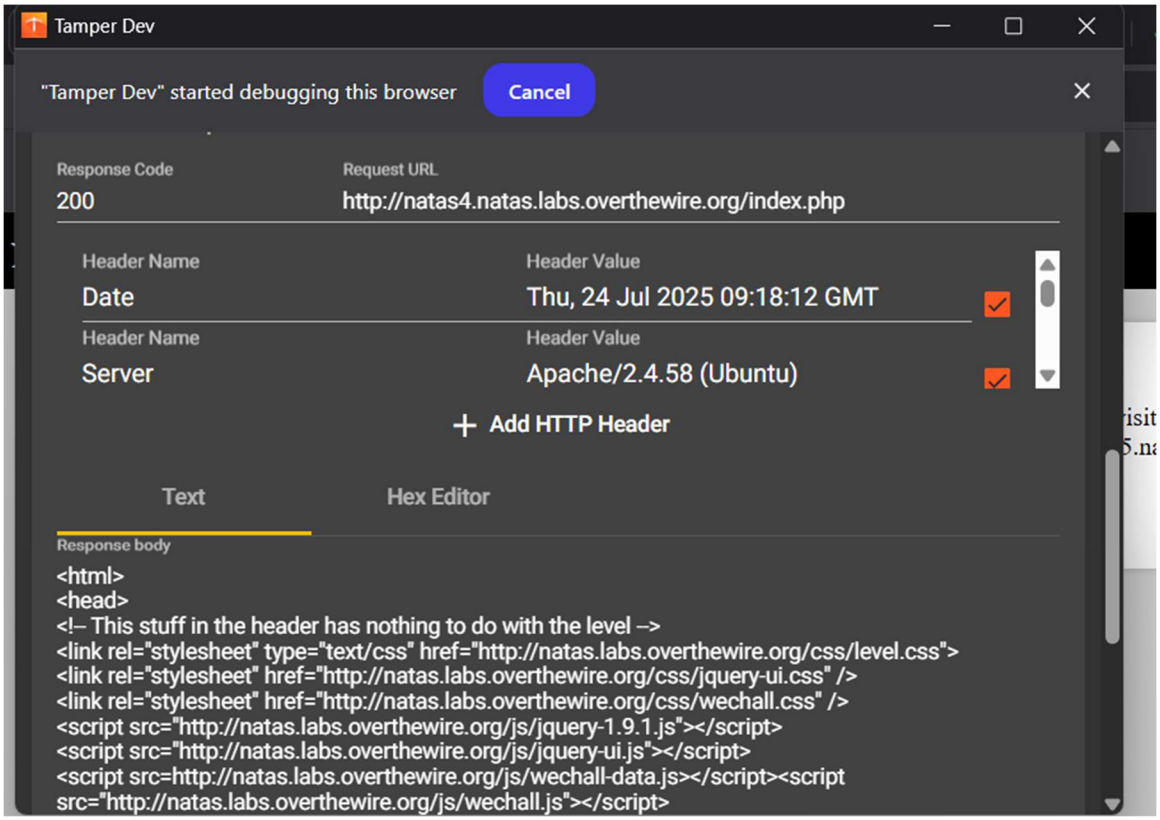
Level 4-5:

Pre-requisites:

- An extension that can tamper the refresh request.

Now using the url we will find the page and refresh the page to go into index.php.

i) We need to make changes in the header of the page to the given url so that we will be allowed .
It looks like this.



ii) change the header and send the request so that we are allowed to access and the credentials that are required for next login will be found in the body of the response

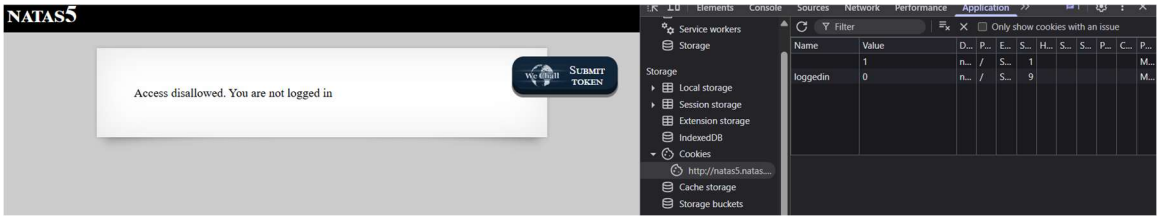
Access granted. The password for natas5 is 0n35PkggAPm2zbEpOU802c0x0Msn1ToK

iii) the credentials are username: natas5
password:0n35PkggAPm2zbEpOU802c0x0Msn1ToK

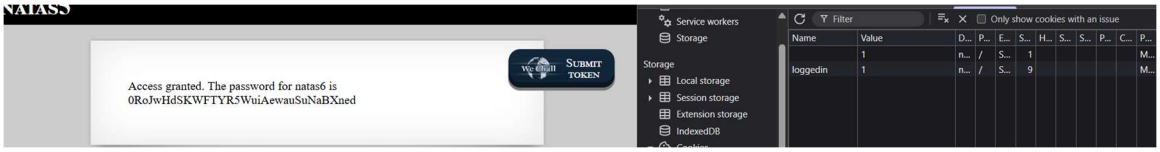
Level 5-6:

i) using the previous credentials, we will be moving to next web page here we will observe “the access is disallowed”

ii) So, on inspecting the page and on various analysis I found that the cookies value is “0”.



iii) so, on trying I found that the value should be greater than “0” and changed it to 1. Refresh the page to get the credentials for next page.



iv) The credentials are

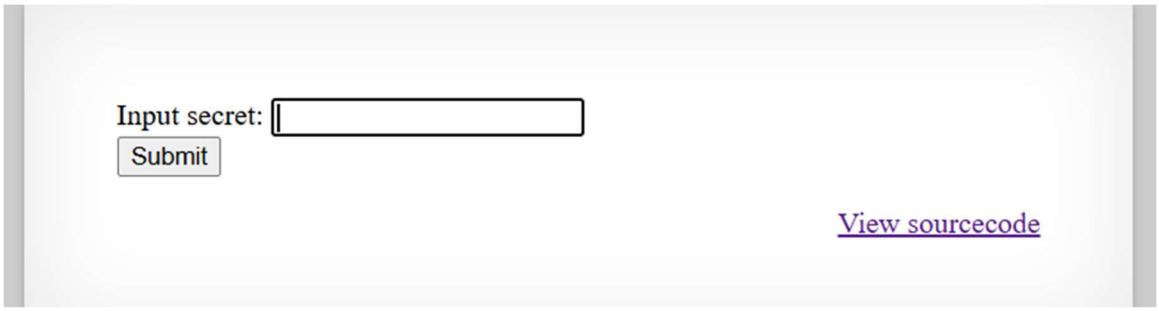
Username: natas6

Password:0RoJwHdSKWFTYR5WuiAewauSuNaBXned

Level 6-7:

i) Using previous credentials we will be directed to next page.

ii) We'll find a input segment asking for input. First



iii) and upon opening the page source we will find the below code

iv) here it includes "includes/secret.inc" so upon we will add that extension to out page url and see what comes up.

```
<?

include "includes/secret.inc";

    if(array_key_exists("submit", $_POST)) {
        if($secret == $_POST['secret']) {
        print "Access granted. The password for natas7 is <censored>";
    } else {
        print "Wrong secret";
    }
    }
?>
```

v) We found that secret code is as follows

```
<?
$secret = "FOEIUWGHFEEUHOFUOIU";
?>
```

vi) Now we will login with the credentials which we received in previous level

vii) Use the credentials given there and login to natas7

viii) The credentials are username: natas7

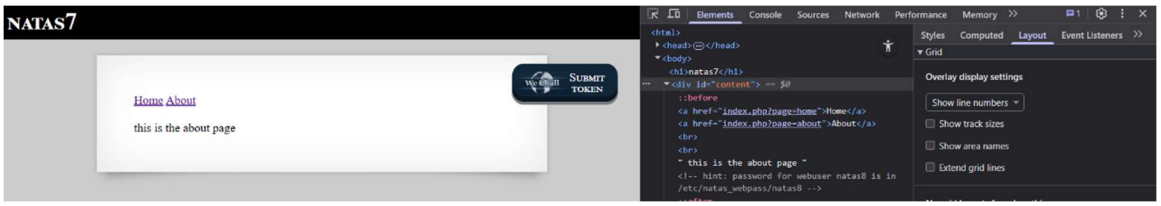
ix) Password: bmg8SvU1LizuWjx3y7xkNERkHxGre0GS

Level 7-8:

i) In natas7 we will find 2 things "home" and "about"

ii) On inspecting the source page, we will find the path to the credentials for natas8

iii) The source page is as follows



iv) The path is "/etc/natas_webpass/natas8". So, the url is http://natas7.natas.labs.overthewire.org/index.php/etc/natas_webpass/natas8

v) There we'll see the user credentials and the final credentials are as follows:
Username: natas8
Password: xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q

Level 8-9:

i) Again we will be asked for secret code and we will observe a link for source page.

ii) On navigating to source page we will observe the following details.

```
<?

$encodedSecret = "3d3d516343746d4d6d6c315669563362";

function encodeSecret($secret) {
    return bin2hex(strrev(base64_encode($secret)));
}

if(array_key_exists("submit", $_POST)) {
    if(encodeSecret($_POST['secret']) == $encodedSecret) {
    print "Access granted. The password for natas9 is <censored>";
    } else {
    print "Wrong secret";
    }
}
?>
```

Encoded secret:3d3d516343746d4d6d6d315669563362

Decoded secret: (base on the function a) from hex, b) reverse the string and do the base64 decoding)

Secret: oubWYf2kBq

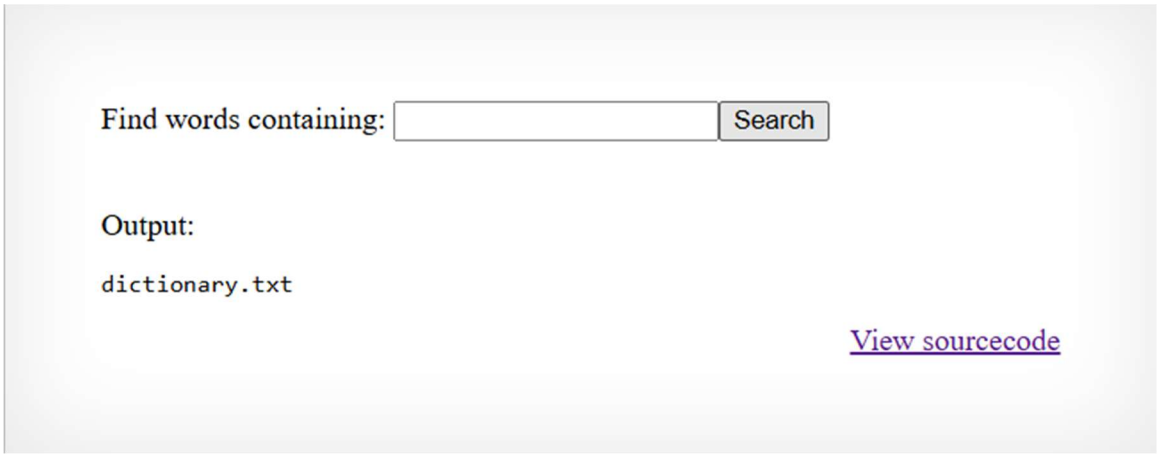
iii) Upon giving the secret input we will get the credentials as follows:
Username: natas9
Password: ZE1ck82lmdGIoErlhQgWND6j2Wzz6b6t

Level 9-10:

i) On logging in to next page we will find as follows

ii) Upon searching from many sources I found that in the input if we give ";" will end the previous instruction and after that command we will get what we need

iii) So, I tried "; ls" I got dictionary.txt like below



iv) Let's test ;cat /etc/natas_webpass/natas10

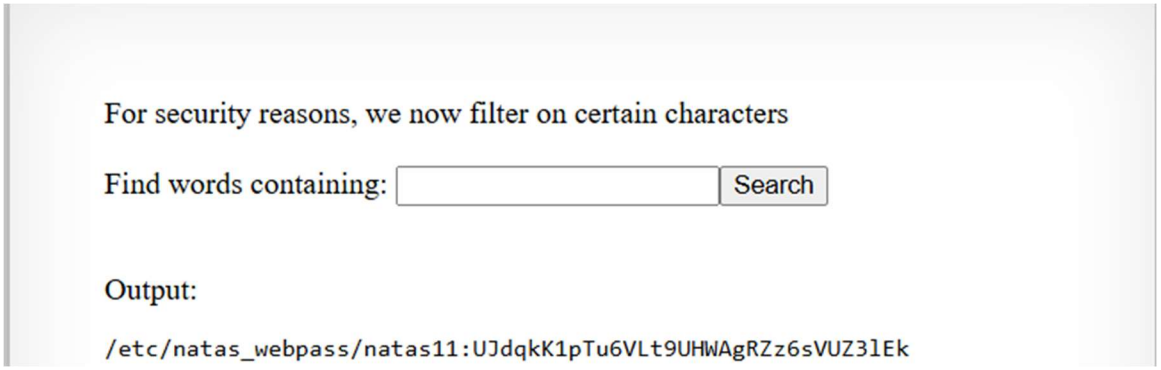
v) The output is as follows

vi) So, the login credentials are
Username: natas10
Password: t7I5VHvpa14sJTUGV0cbEsbYfFP2dmOu

Level 10-11:

i) next page has given that filtered out some words; also, that doesn't take any inputs like "/ [;|&]/"

ii) so, I tried; ls as previous question so it doesn't work. After trying from some possibilities, I found that "<anyletter /....>" is working so I gave an input of "<a /etc/natas_webpass/natas11>" and the output is as follows



iii) finally, the credentials are
username: natas11
password: UJdqkK1pTu6VLt9UHWAgRZz6sVUZ3lEk

level 11-12:

i) for this level we need to find the cookie data and need to decrypt according to the encryption in source page and find the key to solve this level

ii) the key is "eDWo"

iii) cookie's modified data is "HmYkBwozJw4WNyAAFyB1UltmLgoWZntPRyYwDAooOB1HeWlNRiIxCQMiMU0Y"

iv) And the final password is "yZdkjAYZRd3R7tq7T5kXMjMJlOIkzDeB".

Level:12-13:

i) I have uploaded a jpg image in burpesuite browser and then created a .jpg with embedded php

ii) Changed .jpg to .php and found the below password.
trbs5pCjCrkuSknBBKHhaBxq6Wm1j3LC

Level 13-14:

i) uploaded .bmp with php and used burpesuite to solve change the header to .php type and found the below password
"z3UYcr4v4uBpeX8f7EZbMHlzK4UR2XtQ"

level 14-15:

i) I saw a login form and viewed the code. It used:
SELECT * FROM users WHERE username="..." AND password="...";

ii) I tried entering:
" OR 1=1" which always returns true and ignores the rest.

iii) It worked and revealed the password.

iv) "natas15: SdqIqBsFcz3yotlNYErZSZwblkm0lrvx"

Level 15-16:

i) The site only checked if a user existed—no output if the password was correct.

ii) I wrote a Python script to brute-force each character of the password by checking the message "This user exists."

iii) The password is "hPkjKYviLQctEW33QmuXL6eDVfMW4sGo"

Level 16-17:

i) the server used grep to check inputs.

ii) I found it tough, but I got help to write a Python script that tested character-by-character guesses.

iii) After a lot of retries, I got the full password.
EqjHJbo7LFNb8vwhHb9s75hokh5TF0OC

Level 17-18:

i. the output was hidden in comments. So I used SQL injection.

ii. A script tested each character and measured how long the response took.

iii. If the response was delayed, that meant the character was correct. And the password is 6OG1PbKdVjyBlpxgD4DDbRG6ZLlCGgCJ

Level 18-19:

i. The source gave a hint that there is an admin id among the possibilities.

ii. Used burpsuite to find the id's

iii. So the password for natas19: tnwER7PdfWkxsG4FNWUtoAZ9VyZTJqJr

Level 19-20:

i. it used session IDs but encoded them like:
PHPSESSID=3234362d61646d696e

ii. I decoded it to 246-admin, which led me to try different ID-admin combinations encoded in ASCII hex.

iii. I filtered responses for "password" and found the correct one.

iv. So the password for natas20:
p5mCvP7GS2K6Bmt3gqhM2Fc1A5T8MVyw

Level 20-21:

i. The code checked if the user was an admin from the session data.
ii. By using Burp Suite and modifying the input to try%0Aadmin 1 (where %0A is newline), I tricked the system into setting admin to true.
iii. Did it twice, and the second request revealed the password.
iv. So the password for natas21: BPhv63cKE1lkQl04cE5CuFTzXe15NfiH.

These last levels are done in hurry so I did not paste images so this is upto me a long learning journey...