WRITE UP PICOCTF 2021

REZKA NORHAFIZAH

Contents

[GENERAL SKILLS]

OBEDIENT CAT

Diberikan file berupa ascii text, saya langsung menggunakan perintah cat untuk melihat isinya :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# cat flag
picoCTF{s4n1ty_v3r1f13d_2fd6ed29}
```

Flag = picoCTF{s4n1ty_v3r1f13d_2fd6ed29}

PYTHON WRANGLING

Diberikan 3 file (2 file berupa ascii text, dan 1 file berupa script python), kemudian saya lihat isi kedua file ascii text tersebut :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# cat flag.txt.en                                                                                                  1 ×
gAAAAABgUAIWIvSiR0W23DAHIK5DX6Y4BvwES94M_XdDcNAquhp-A0D2z8n812YEXaSD9WhoweBh2cm5Wa0cqzuW0Kc7fOct0OJnpOmVF8A91j0Hx4dKt
vk3l5ghPT71Y7GxErPRyJUs

┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# cat pw.txt
ac9bd0ffac9bd0ffac9bd0ffac9bd0ff
```

Selanjutnya, setelah melihat isi file saya langsung eksekusi dengan python3 script tersebut :

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# python3 ./ende.py
Usage: ./ende.py (-e/-d) [file]
```

Namun, ternyata tidak berhasil dan memunculkan clue bahwa harus ada file yang di encode dan di decode. Lalu, saya coba kembali eksekusi dengan menyertakan file flag.txt.en sebagai text yang di encode dan ternyata diminta untuk memasukkan password :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# python3 ./ende.py -d flag.txt.en
Please enter the password:ac9bd0ffac9bd0ffac9bd0ffac9bd0ff
picoCTF{4p0110_1n_7h3_h0us3_ac9bd0ff}
```

Kemudian saya masukkan passwordnya dan memunculkan flag.

Flag = picoCTF{4p0110_1n_7h3_h0us3_ac9bd0ff}

WAVE A FLAG

Diberikan file maka saya langsung analisis dan ternyata itu adalah ELF file :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# file warm
warm: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64
.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=01b148cdedfc38125cac0d87e0537466d47927b1, with debug_info, not stripped
```

Kemudian saya strings file tersebut dan ternyata flag nya langsung muncul :

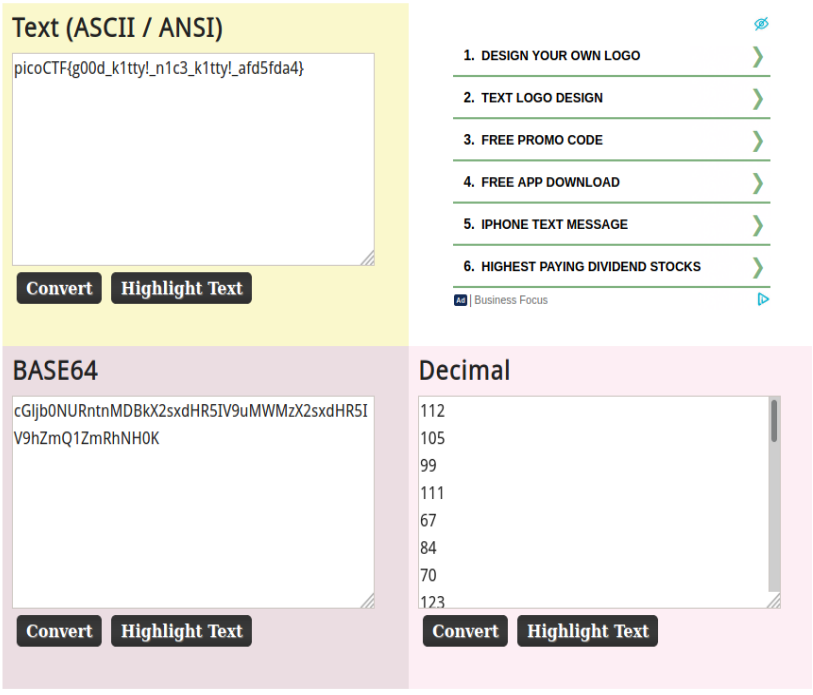
```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# strings warm
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
printf
__cxa_finalize
strcmp
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
=y
=W
=Z
AWAVI
AUATL
[]A\A]A^A_
Hello user! Pass me a -h to learn what I can do!
Oh, help? I actually don't do much, but I do have this flag here: picoCTF{b1scu1ts_4nd_gr4vy_f0668f62}
I don't know what '%s' means! I do know what -h means though!
;*3$"
```

Flag = picoCTF{b1scu1ts_4nd_gr4vy_f0668f62}

NICE NETCAT….

Diberikan sebuah netcat listener, lalu langsung saya jalankan dan muncul deretan angka decimal :

```
┌──(kali㉿kali)-[~]
└─$ nc mercury.picoctf.net 21135
112
105
99
111
67
84
70
123
103
48
48
100
95
107
49
116
116
121
33
95
110
49
99
51
95
107
49
116
116
121
33
95
97
102
100
53
102
100
97
52
125
10
```



Deretan decimal tersebut saya konversikan ke ascii dan menghasilkan flag.

Flag = picoCTF{g00d_k1tty!_n1c3_k1tty!_afd5fda4}

STATIC AIN'T ALWAYS NOISE....

Diberikan 2 file, yaitu ELF file dan bash script :

```
┌──(root💀 kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# file static
static: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-
64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=9f1762ab580608bef0d251f5fdfaad3d19ae0963, not stripped

┌──(root💀 kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# file ltdis.sh
ltdis.sh: Bourne-Again shell script, ASCII text executable
```

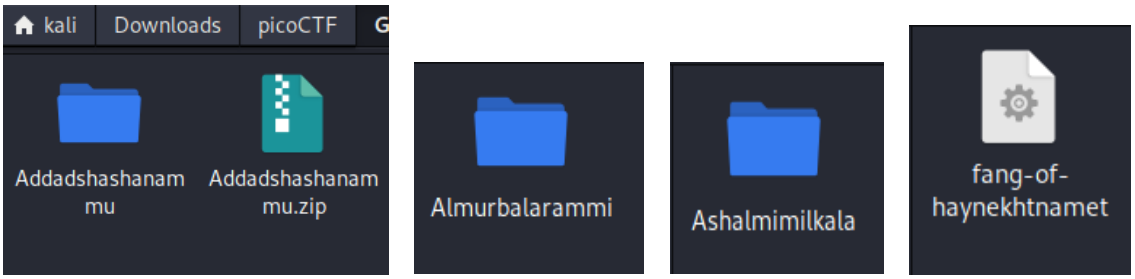
Kemudian saya strings file static dan langsung memunculkan flag :

```
┌──(root💀 kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# strings static
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
AWAVI
AUATL
[]A\A]A^A_
Oh hai! Wait what? A flag? Yes, it's around here somewhere!
;*3$"
picoCTF{d15a5m_t34s3r_6f8c8200}
GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
crtstuff.c
deregister_tm_clones
```

Flag = picoCTF{d15a5m_t34s3r_6f8c8200}

TAB, TAB, ATTACK

Diberikan sebuah file zip, lalu saya ekstrak dan ternyata di dalam nya berupa folder-folder. Kemudian saya buka folder-folder tersebut sampai mendapat file fang-of-haynekhtnamet :



Selanjutnya, saya analisis file tersebut dan yang ternyata adalah ELF file. Lalu, langsung saya jalankan :

```
┌──(kali㉿kali)-[~/…/Assurnabitashpi/Maelkashishi/Onnissiralis/Ularradallaku]
└─$ file fang-of-haynekhtnamet
fang-of-haynekhtnamet: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked,

┌──(kali㉿kali)-[~/…/Assurnabitashpi/Maelkashishi/Onnissiralis/Ularradallaku]
└─$ ./fang-of-haynekhtnamet
*ZAP!* picoCTF{l3v3l_up!_t4k3_4_r35t!_2bcfb2ab}
```

Flag = picoCTF{l3v3l_up!_t4k3_4_r35t!_2bcfb2ab}

MAGIKARP GROUND MISSION

DIberikan akses ssh dengan username beserta password, lalu langsung saya jalankan dan pertama saya menggunakan perintah ls sesuai yang diinstruksikan soal :

```
┌──(kali㉿kali)-[~]
└─$ ssh ctf-player@venus.picoctf.net -p 50711
The authenticity of host '[venus.picoctf.net]:50711 ([3.131.124.143]:50711)' can't be established.
ECDSA key fingerprint is SHA256:NrQkIxNEQQho/GA7jE0WlIa7Jh4VF9sAvC5awkbuj1Q.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[venus.picoctf.net]:50711,[3.131.124.143]:50711' (ECDSA) to the list of known hosts.
ctf-player@venus.picoctf.net's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1041-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@pico-chall$ ls
1of3.flag.txt  instructions-to-2of3.txt
ctf-player@pico-chall$ cat 1of3.flag.txt
picoCTF{xxsh_
```

Seperti dilihat pada gambar, terdapat dua file, dan setelah saya lihat terlihat satu file merupakan flag tapi ternyata masih berupa sebagian. Lalu di file satunya terdapat instruksi untuk masuk ke direktori root :

```
ctf-player@pico-chall$ cat instructions-to-2of3.txt
Next, go to the root of all things, more succinctly `/`
ctf-player@pico-chall$ cd /
ctf-player@pico-chall$ ls
2of3.flag.txt  boot  etc   instructions-to-3of3.txt  lib64  mnt  proc  run   srv  tmp  var
bin            dev   home  lib                       media  opt  root  sbin  sys  usr
ctf-player@pico-chall$ cat 2of3.flag.txt
0ut_0f_\/\/4t3r_
ctf-player@pico-chall$ cat instructions-to-3of3.txt
Lastly, ctf-player, go home... more succinctly `~`
```

Lalu, saya mengikuti instruksi yang ada dan ternyata terdapat file yang sama (1 file merupakan bagian flag, dan yang lainnya berupa instruksi). Selanjutnya, saya mengikuti instruksi lagi yaitu kembali ke home dengan perintah ~ .

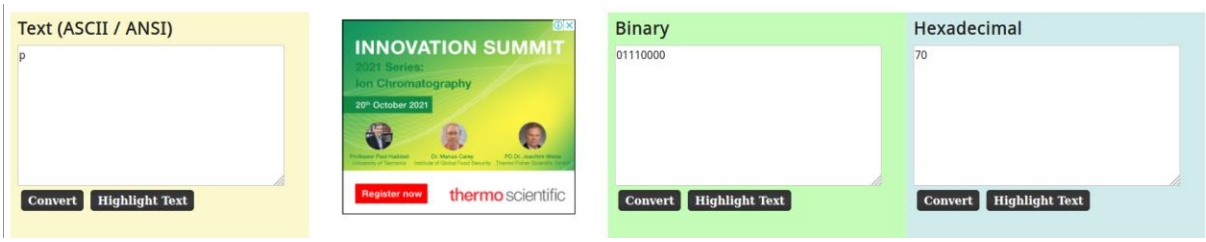
```
ctf-player@pico-chall$ cd ~
ctf-player@pico-chall$ ls
3of3.flag.txt  drop-in
ctf-player@pico-chall$ cat 3`
> ^C
ctf-player@pico-chall$ ls
3of3.flag.txt  drop-in
ctf-player@pico-chall$ cat 3of3.flag.txt
71be5264}
```

Setelah itu, saya mendapat file flag lagi dan gabungkan saja semua flag yang didapat maka didapat flagnya.

Flag = picoCTF{xxsh_0ut_0f_\/\/4t3r_71be5264}

LETS WARM UP

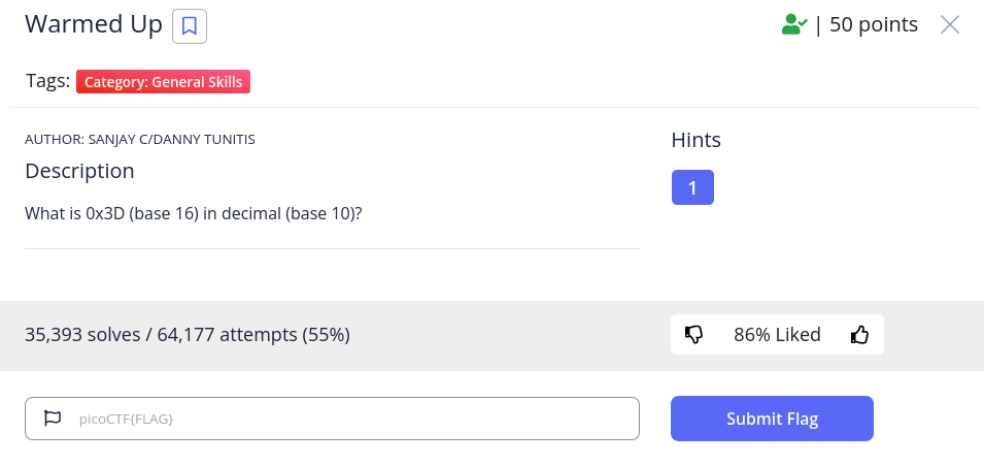
Saya melakukan konversi dari hexa 0x70 ke ascii :



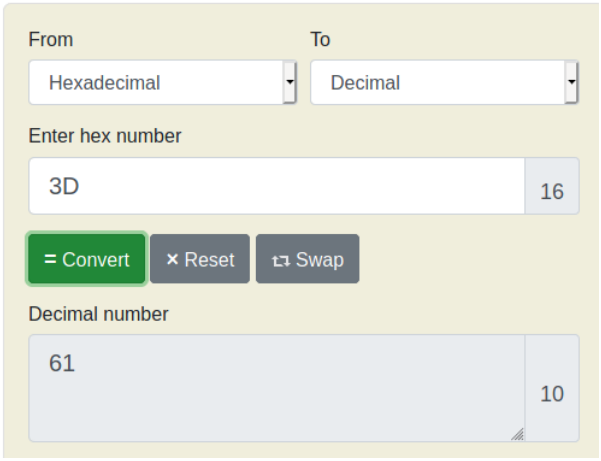
Flag = picoCTF{p}

WARMED UP

Saya melakukan konversi 0x3D yang merupakan base16 (hexadecimal) ke base10(decimal) :



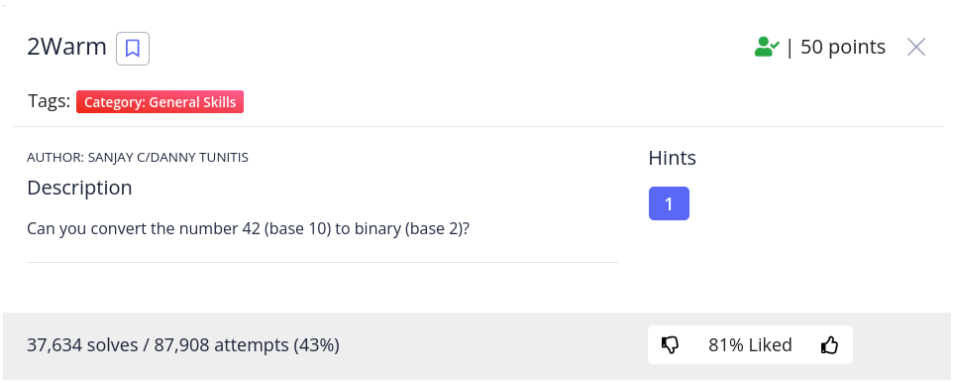
Hexadecimal to Decimal converter



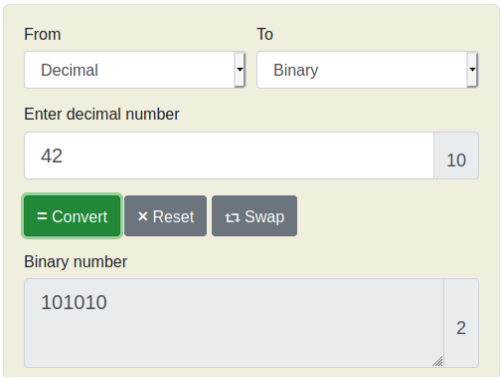
Flag = picoCTF{61}

2WARM

Saya melakukan konversi dari 42 yang merupakan base10 (decimal) ke base2 (binary) :



Decimal to Binary converter



Flag = picoCTF{101010}

WHAT'S A NET CAT?

Diberikan netcat listener, saya jalankan dan muncul flag :

```
┌──(kali㉿kali)-[~]
└─$ nc jupiter.challenges.picoctf.org 64287
You're on your way to becoming the net cat master
picoCTF{nEtCat_Mast3ry_284be8f7}
```

Flag = picoCTF{nEtCat_Mast3ry_284be8f7}

STRINGS IT

Diberikan file, lalu langsung saya strings sesuai clue soal dan didapat :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# strings strings
/lib64/ld-linux-x86-64.so.2
libc.so.6
puts
stdout
__cxa_finalize
setvbuf
__libc_start_main
GLIBC_2.2.5
__gmon_start__
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
=11'
=g
=j
AWAVI
AUATL
[]A\A]A^A_
Maybe try the 'strings' function? Take a look at the man page
;*3$"
XMdasaWpAXqIHqvFBYTt
32VO1kKGW7st50mkv
B2WqFg3mFhCfUyvG3sNEs9Ep3FYP2gEkUePqFgUVN30MAZtV
zc2qhtc8wESHxGya1S9WpEXLgKo4D8ZrKODtQ4
YkHTxIzcJIJkLASXl1wr3EJ2IfoAKQskB7CYl54MqTnFhKd4
8s0TgxgiH9CL890ND9xWiSa2Y3iH8UeyjDfcLjougfE8
Ltd2AGC5UT2165K4WnvoNBbi0oauIrvQDTzBTWPQqp3PVlj
pp6DsuJVD5M8STb8BSUZD2WSVewAXSjZYjyumV
NJhIzdIKzLkeSDE5xAGul7rbxbsgThIyLL8sMDfFxcc
7uZYDkCYS7X7PCpioV
Z3IqcPbmAvXaUHJ0k8gdMvS7oWUds0qXme3sT
n6DXS8V5ckI69aW1HWpBQJqWtLP7
xOXVJSJLwQSEM9czbp65M
BUra2vYXnSzSCI8t2wfpIS5yW
CegIU1EsiFMtMzfWecPdeDIpCjC
F0ErUDevXbvJ4h7TzN8ZpZr7Zihy3PxF4p8OSJgoOMBM5
zI0IaBkin3yYMLH1B2PQ30
CD2PgwllLChjvU1WeqPc7CvZXPMS5TEYFrwil
kRVKmtGsZ50Vur9B6fbwaVrK
HcN5aYkkLBqCyuXX7r4W0OLVpGIy
4mjSVGAe8mrk3ZsH0jE
tof3pKjWoapV6JTk133
l7ztg2JgNU3g4Yl7psB8bqPYOe7v44ThiI0xtQOLSmkRWs8
ylGhSUkX5GQrA8Hr2bL6CqFulmPMKuUY6asDxCRexyxdB2QCpclU7
YXBmvJArPSie44FnIuFjL0Wtkc0bQmR
```

Karena terlalu banyak memunculkan string, saya menggunakan perintah grep untuk memunculkan flag :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/GeneralSkills]
└─# strings strings | grep pico
picoCTF{5tRIng5_1T_827aee91}
```

Flag = picoCTF{5tRIng5_1T_827aee91}

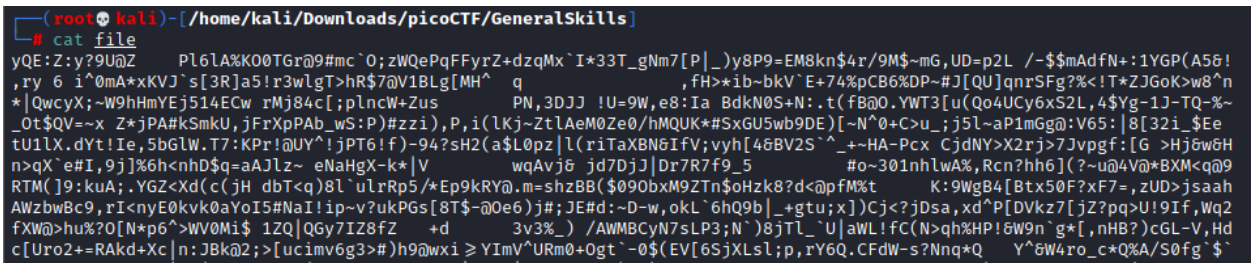
BASES

Diberikan base64, saya konversi ke ascii dan didapat flag :

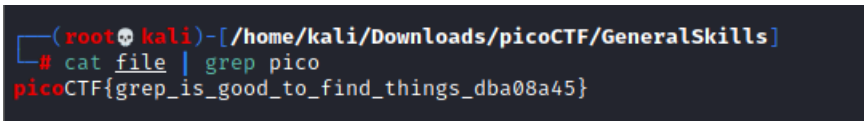
Flag = picoCTF{l3arn_th3_r0p35}

FIRST GREP

Diberikan sebuah file, lalu saya coba lihat ternyata muncul banyak string seperti berikut :



Lalu, saya menggunakan perintah grep sesuai clue soal dan didapat flag :



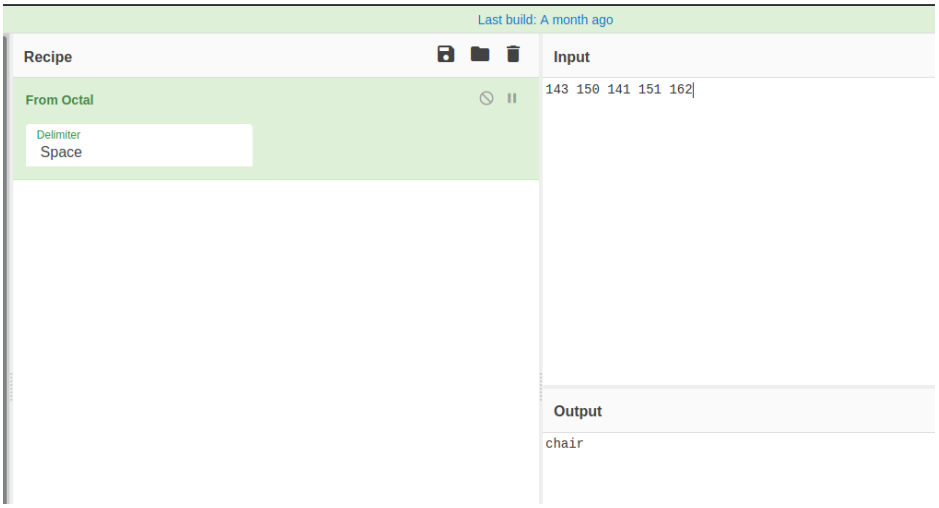
Flag = picoCTF{grep_is_good_to_find_things_dba08a45}

BASED

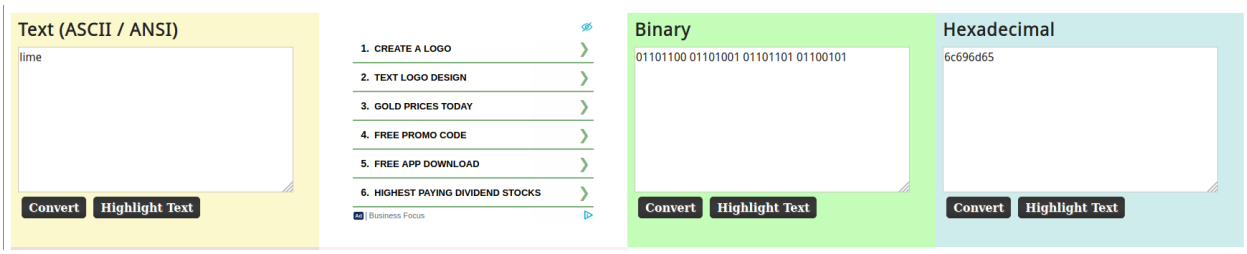
Diberikan sebuah netcat listener, kemudian saya jalankan dan muncul sebuah inputan. Setelah saya cermati, ternyata perintahnya ialah menerjemahkan dari deretan bilangan yang dimunculkan dan diinput sebagai ascii (word). Setelah itu, saya konversi binary yang keluar menjadi ascii yaitu “table” :

```
┌──(root💀kali)-[/home/kali/mnt/LKSN 2021]
└─# nc jupiter.challenges.picoctf.org 29221
Let us see how data is stored
table
Please give the 01110100 01100001 01100010 01101100 01100101 as a word.
...
you have 45 seconds.....

Input:
table
Please give me the  143 150 141 151 162 as a word.
Input:
chair
Please give me the 6c696d65 as a word.
Input:
lime
You've beaten the challenge
Flag: picoCTF{learning_about_converting_values_00a975ff}
```

Selanjutnya, muncul deretan bilangan yang ternyata ialah bilangan octal yang jika dikonversi menjadi chair :



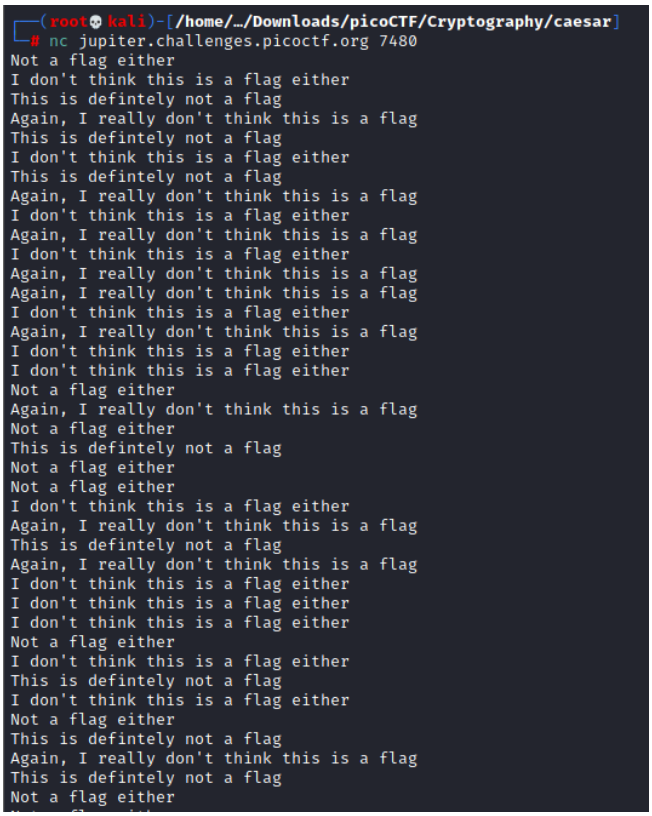
Lalu, muncul lagi deretan bilangan yang berupa hexadecimal dan saya konversi lagi ke ascii :



Flag = picoCTF{learning_about_converting_values_00a975ff}

PLUMBING

Diberikan sebuah nama website beserta portnya, lalu saya coba jalankan menggunakan netcat dan didapat :



Karena terlalu banyak menghasilkan output, saya menggunakan perintah grep untuk memunculkan flag :

Flag = picoCTF{digital_plumb3r_06e9d954}

[WEB EXPLOITATION]

INSP3CT0R

Diberikan sebuah website dengan tampilan berikut :

Lalu, saya coba inspect dengan melihat page source (Ctrl+U) web tersebut dan ternyata muncul hint berupa 1/3 bagian dari flag :

```
1  <!doctype html>
2  <html>
3    <head>
4      <title>My First Website :)</title>
5      <link href="https://fonts.googleapis.com/css?family=Open+Sans|Roboto" rel="stylesheet">
6      <link rel="stylesheet" type="text/css" href="mycss.css">
7      <script type="application/javascript" src="myjs.js"></script>
8    </head>
9
10   <body>
11     <div class="container">
12       <header>
13     <h1>Inspect Me</h1>
14       </header>
15
16       <button class="tablink" onclick="openTab('tabintro', this, '#222')" id="defaultOpen">What</button>
17       <button class="tablink" onclick="openTab('tababout', this, '#222')">How</button>
18
19       <div id="tabintro" class="tabcontent">
20     <h3>What</h3>
21     <p>I made a website</p>
22       </div>
23
24       <div id="tababout" class="tabcontent">
25     <h3>How</h3>
26     <p>I used these to make this site: <br/>
27       HTML <br/>
28       CSS <br/>
29       JS (JavaScript)
30     </p>
31     <!-- Html is neat. Anyways have 1/3 of the flag: picoCTF{tru3_d3 -->
32       </div>
33
34     </div>
35
36   </body>
37 </html>
38
```

Setelah itu, saya klik link mycss.css dan myjs.js dan ternyata di bagian akhir halamannya terdapat sisa flag nya :

```
/* You need CSS to make pretty pages. Here's part 2/3 of the flag: t3ct1ve_0r_ju5t */
```

```
/* Javascript sure is neat. Anyways part 3/3 of the flag: _lucky?832b0699} */
```

Flag = picoCTF{tru3_d3t3ct1ve_0r_ju5st_lucky?832b0699}

GET AHEAD

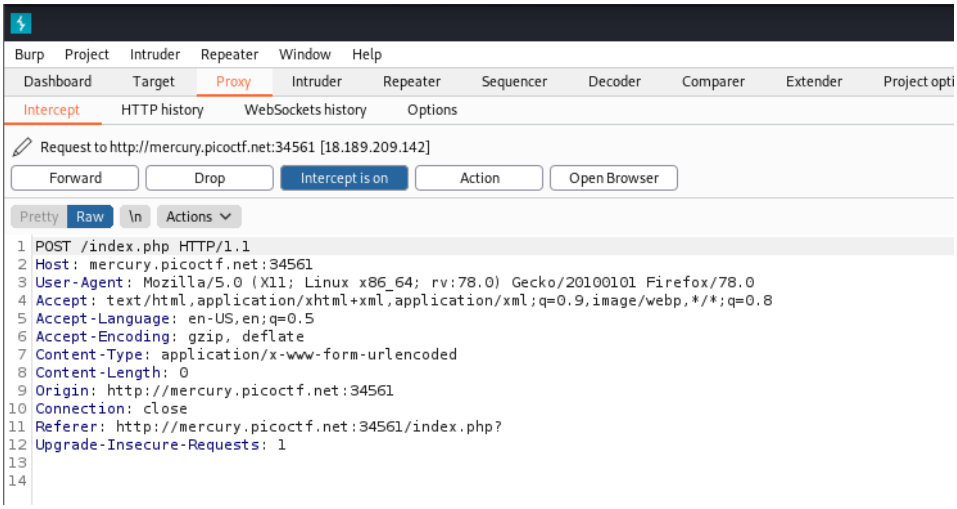
Diberikan website dengan tampilan berikut :

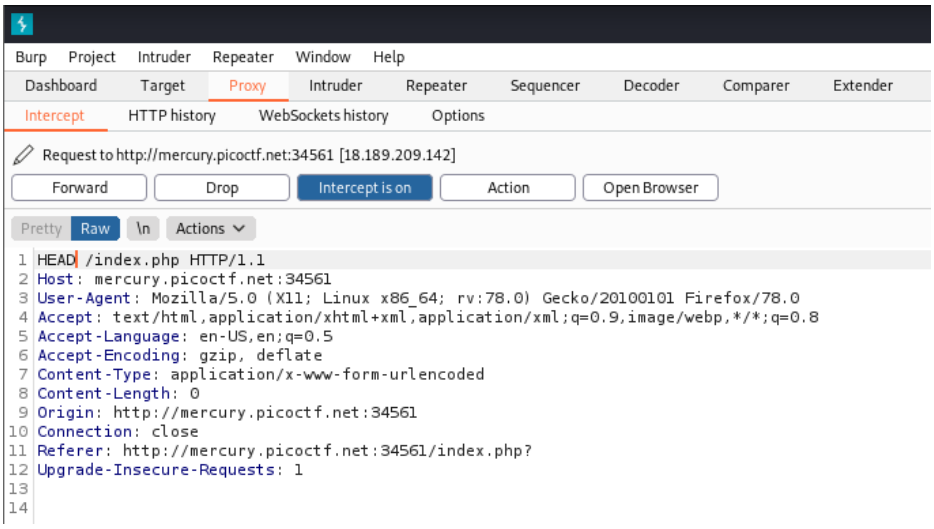
Kemudian sesuai hint soal memberitahu bahwa mungkin perlu 2 pilihan untuk penyelesaiannya, setelah itu saya melihat di page source bahwa metode request untuk Red adalah GET dan untuk Blue adalah POST :

```
<!doctype html>
<html>
<head>
    <title>Red</title>
    <link rel="stylesheet" type="text/css" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
    <style>body {background-color: red;}</style>
</head>
    <body>
        <div class="container">
            <div class="row">
                <div class="col-md-6">
                    <div class="panel panel-primary" style="margin-top:50px">
                        <div class="panel-heading">
                            <h3 class="panel-title" style="color:red">Red</h3>
                        </div>
                        <div class="panel-body">
                            <form action="index.php" method="GET">
                                <input type="submit" value="Choose Red"/>
                            </form>
                        </div>
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="panel panel-primary" style="margin-top:50px">
                        <div class="panel-heading">
                            <h3 class="panel-title" style="color:blue">Blue</h3>
                        </div>
                        <div class="panel-body">
                            <form action="index.php" method="POST">
                                <input type="submit" value="Choose Blue"/>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </body>
</html>
```

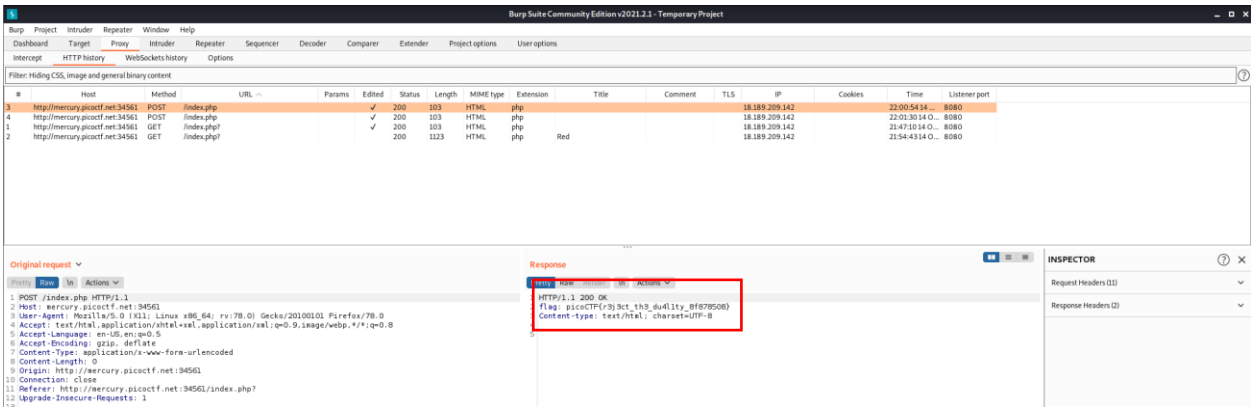
Selanjutnya, hint soal mengatakan bahwa tools yang digunakan adalah burp suite, lalu saya coba buka burp suite dan menghidupkan intercept nya, kemudian data request dari client akan disampaikan terlebih dahulu lewat burp suite dan kemudian nanti kita bisa memutuskan data tersebut akan di forward ke server atau tidak.

Kemudian, saya coba jalankan dengan memilih warna Blue dan sesuai page source request method yang digunakan adalah POST. Lalu, saya mengubah metode requestnya menjadi HEAD (sesuai hint pada nama soal yaitu GET aHEAD) dan saya forward ke server :





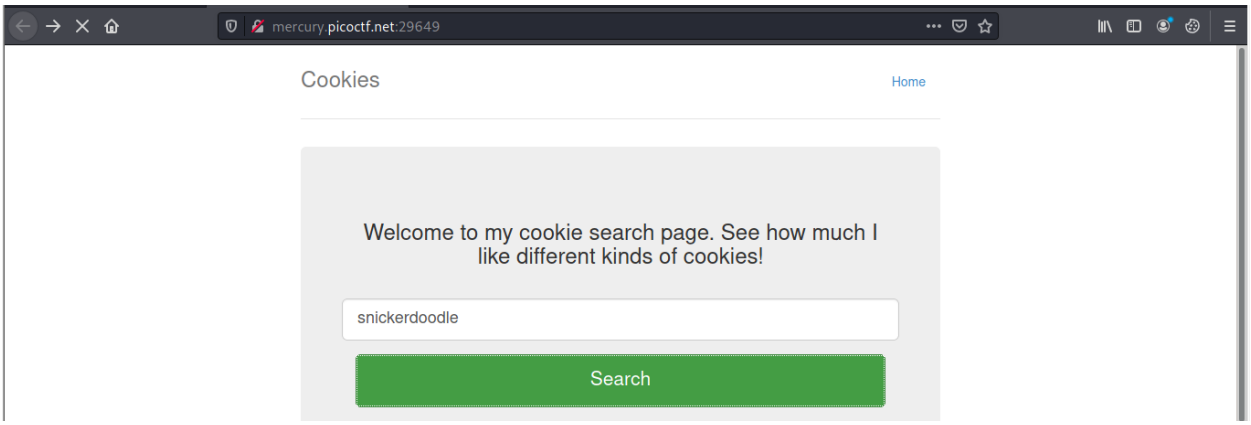
Setelah itu, saya coba lihat bagian header response nya, dan ternyata menghasilkan flag :



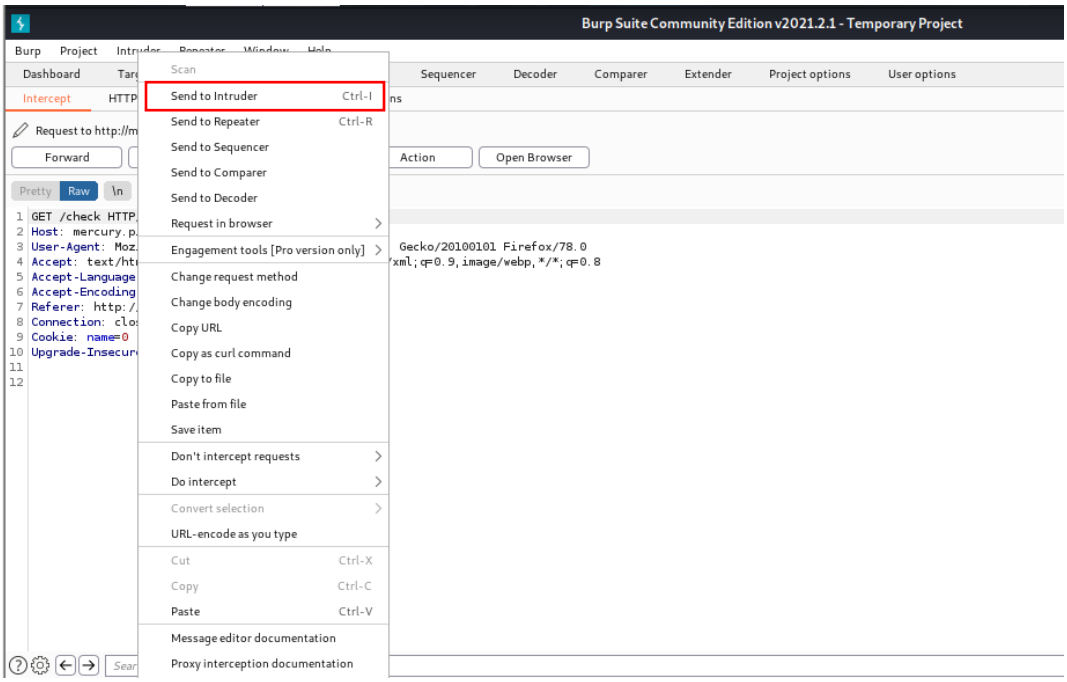
Flag = picoCTF{r3j3ct_th3_du4l1ty_8f878508}

COOKIES

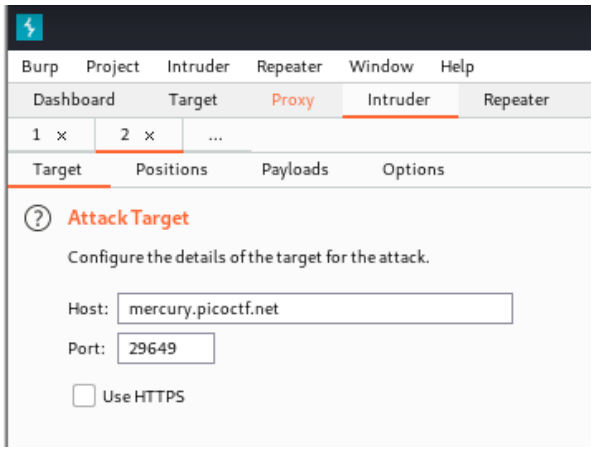
Diberikan sebuah website dengan tampilan berikut, di mana ini merupakan website yang dibuat untuk pencarian cookie :



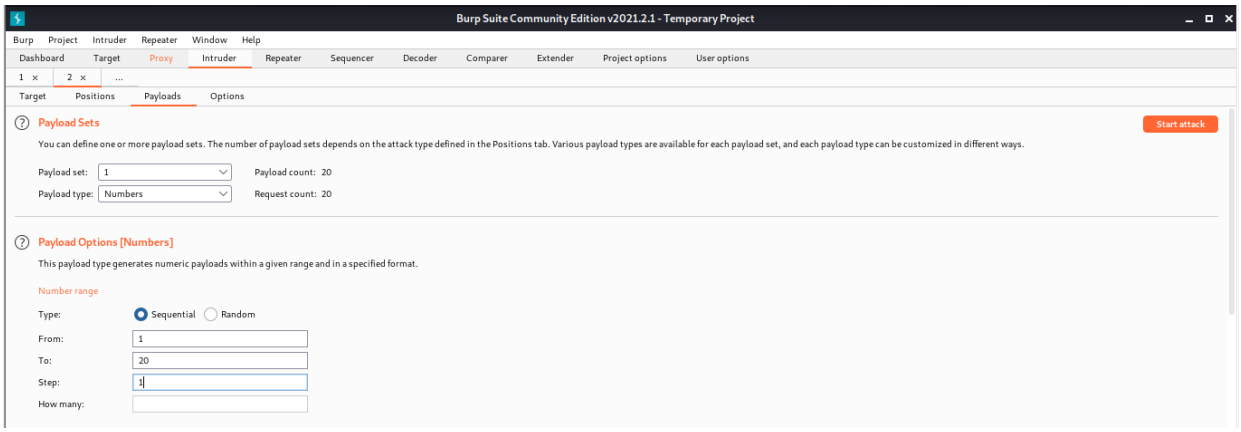
Setelah itu, saya coba input “snickerdoodle” sesuai dengan kata yang ditampilkan pertama kali di website, dan setelah saya lihat nilai cookienya ternyata bernilai 0. Kemudian saya berpikir untuk melihat output yang keluar nantinya apabila saya merubah nilai cookienya, untuk itu saya menggunakan burp suite dan kemudian request data yang saya lakukan saya send to intruder supaya saya bisa memasang payloads untuk cookie secara sekaligus nantinya :



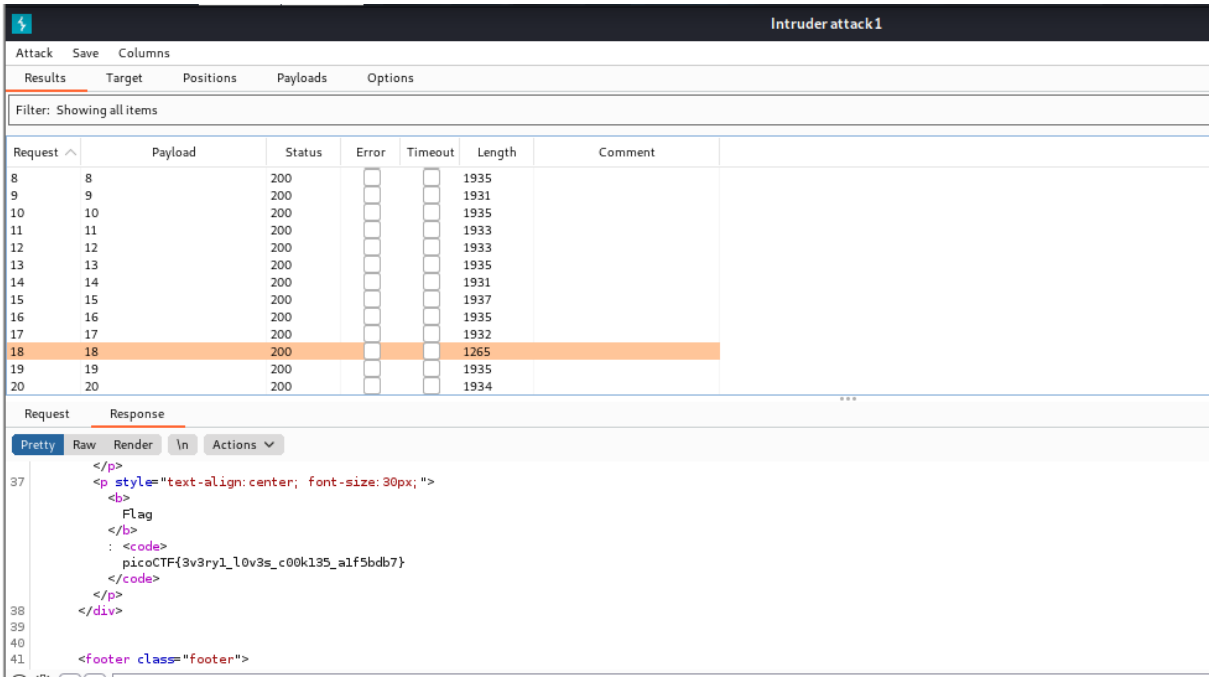
Setelah itu, di bagian intruder saya set target attack nya http://mercury.picoctf.net:29649 :



Dan untuk payload nya saya set 1 dengan type numbers dan range dari 1-20 sehingga jumlah requestnya sebanyak 20 request :



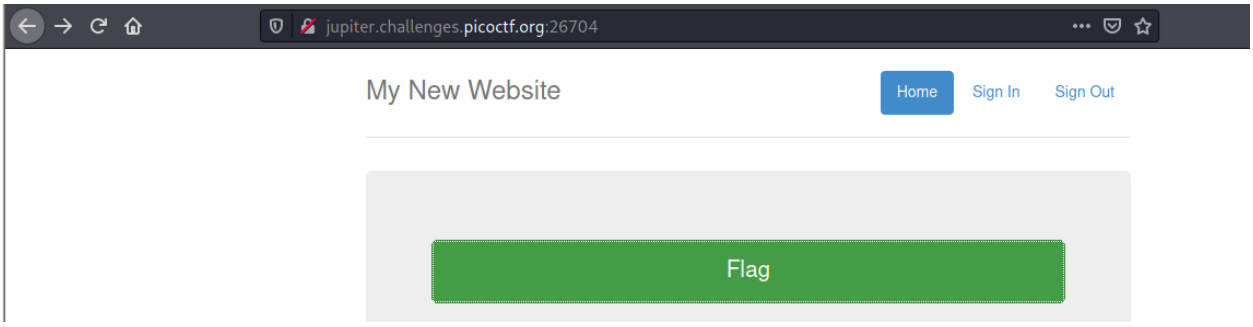
Setelah itu, saya jalankan dan saya cek header response nya satu persatu ternyata di request no 18 terdapat flag :



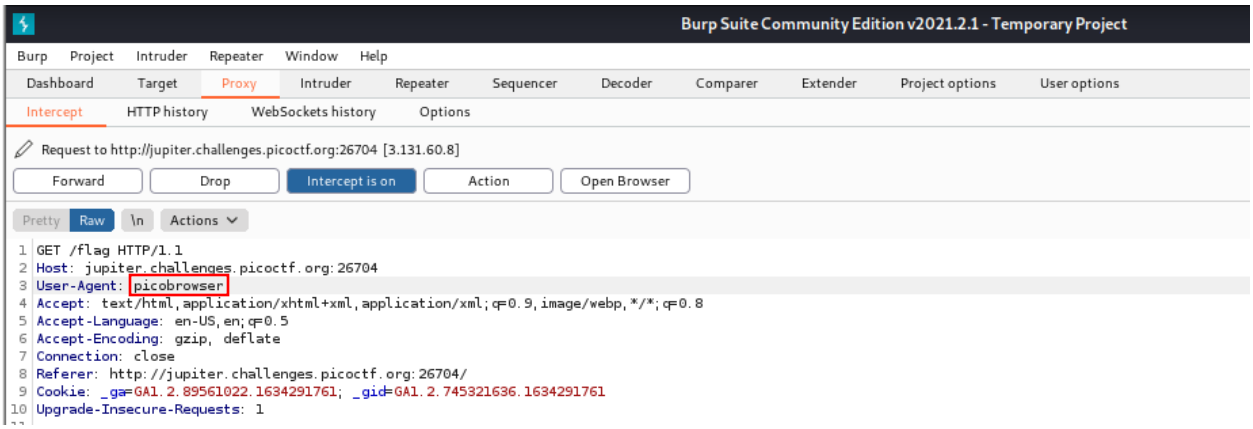
Flag = picoCTF{3v3ry1_l0v3s_c00k135_alf5bdb7}

PICOBROWSER

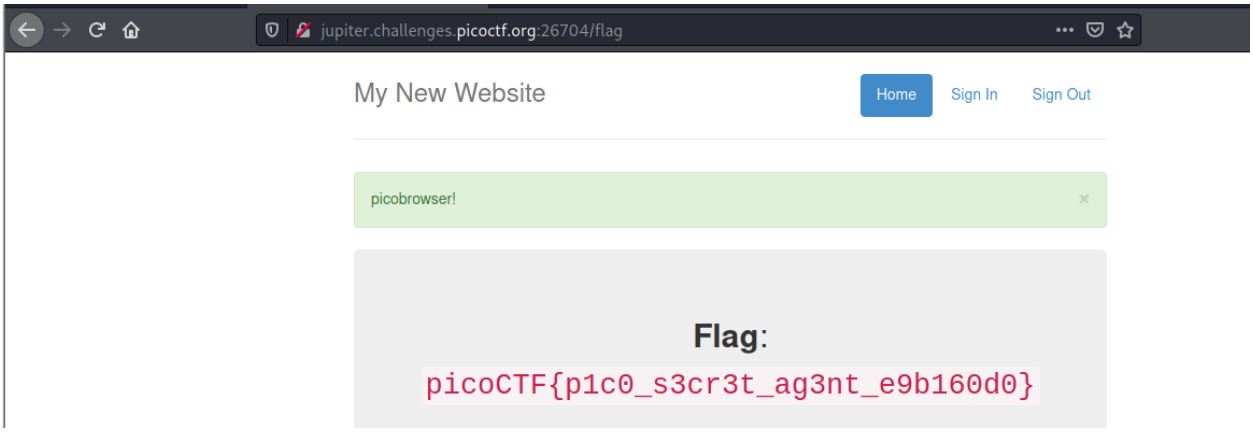
Diberikan sebuah website dengan tampilan berikut :



Kemudian, sesuai hint soal bahwa web tersebut hanya bisa diakses dengan user agent picobrowser, maka dari itu saya menggunakan burp suite untuk memodifikasi user agent-nya :



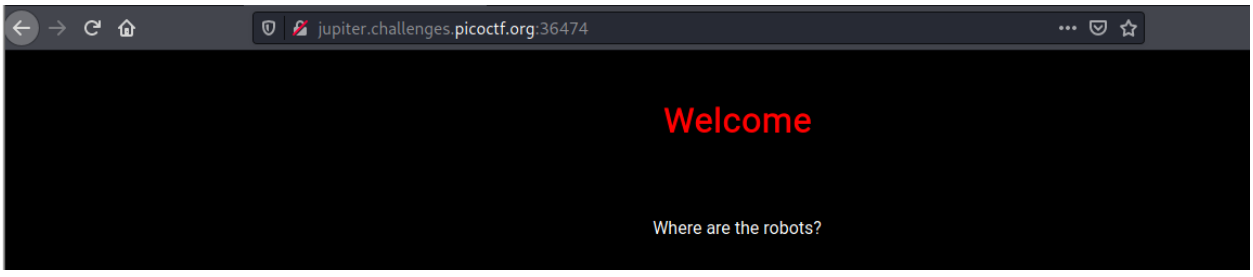
Selanjutnya, saya forward ke server maka tampilan web akan berubah menjadi flag yang direquest :



Flag = picoCTF{p1c0_s3cr3t_ag3nt_e9b160d0}

WHERE ARE THE ROBOTS

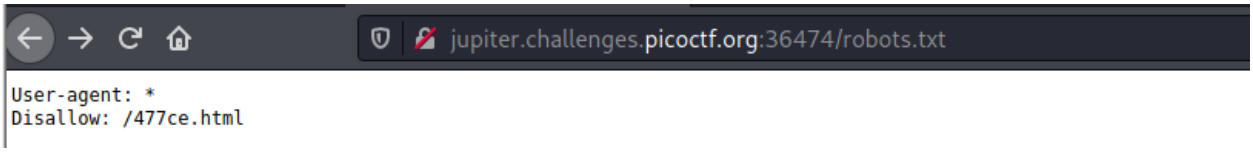
Diberikan website dengan tampilan berikut :



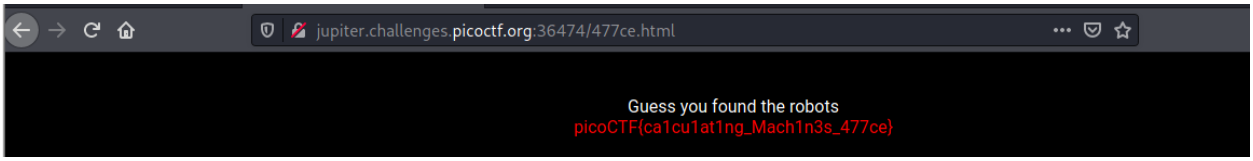
Setelah itu, terdapat hint bahwa “What part of the website could tell you where the creator doesn't want you to look?” di sini saya menyimpulkan kemungkinan 2 hal yang dimaksud, yaitu pertama ialah page source nya dan yang kedua dir listing yang disembunyikan. Namun, setelah saya cek page source nya tidak ada apa-apa, dan ketika saya scan dir listing nya menggunakan gobuster, saya menemukan directory robots.txt :

```
┌──(kali㉿kali)-[~/Downloads]
└─$ gobuster dir -u http://jupiter.challenges.picoctf.org:36474 -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://jupiter.challenges.picoctf.org:36474
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2021/10/15 08:32:17 Starting gobuster in directory enumeration mode
===============================================================
/index.html           (Status: 200) [Size: 431]
/robots.txt           (Status: 200) [Size: 36]

===============================================================
2021/10/15 08:39:31 Finished
===============================================================
```

Kemudian, saya run di website tersebut dan ternyata terdapat hint lagi untuk ke directory 477ce.html :



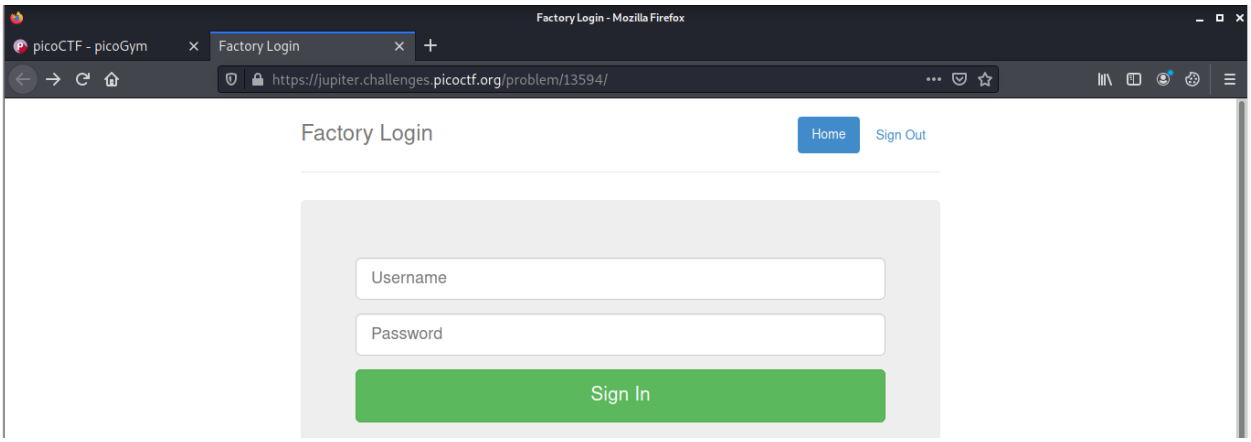
Setelah itu, saya beralih ke directory tersebut dan ternyata terdapat flag :



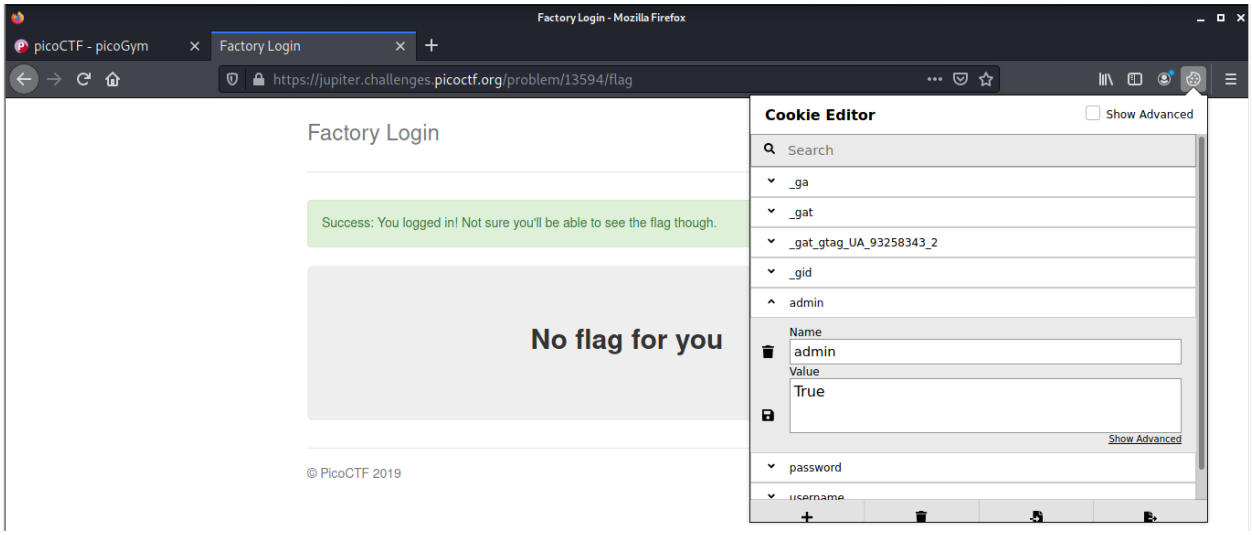
Flag = picoCTF{ca1cu1at1ng_Mach1n3s_477ce}

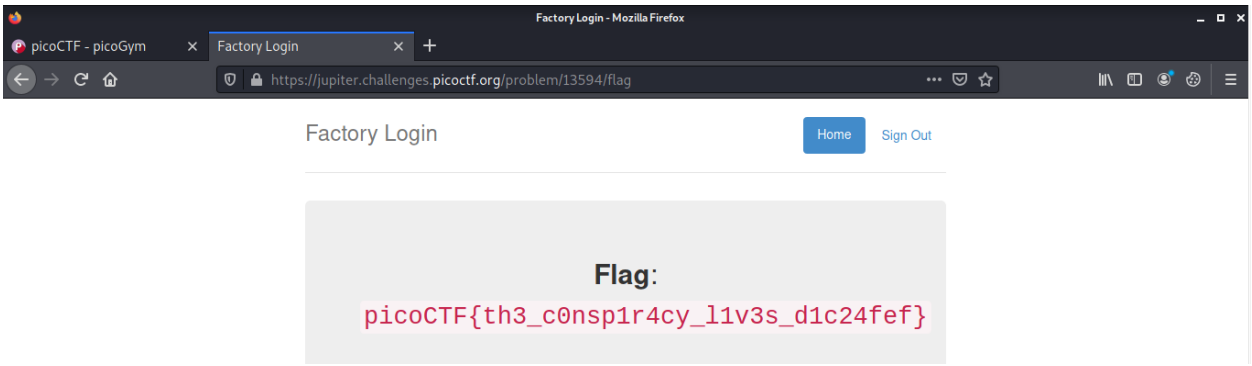
LOGON

Diberikan sebuah website dengan tampilan login seperti berikut :



Kemudian, saya coba input sembarang karakter dan ternyata tidak menghasilkan apa-apa. Lalu, saya cek cookie yang ada pada website dan menemukan ada cookie dengan nama admin dengan value False, langsung saja saya ubah menjadi True dan ternyata menampilkan flag :

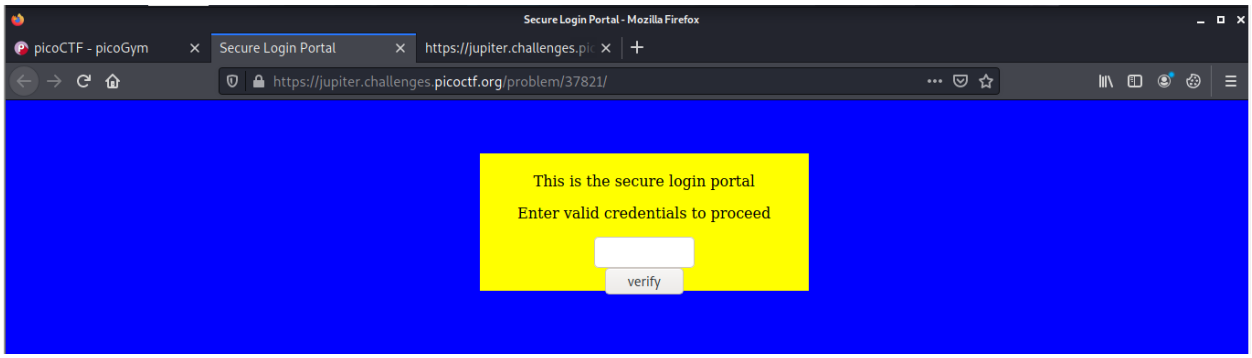




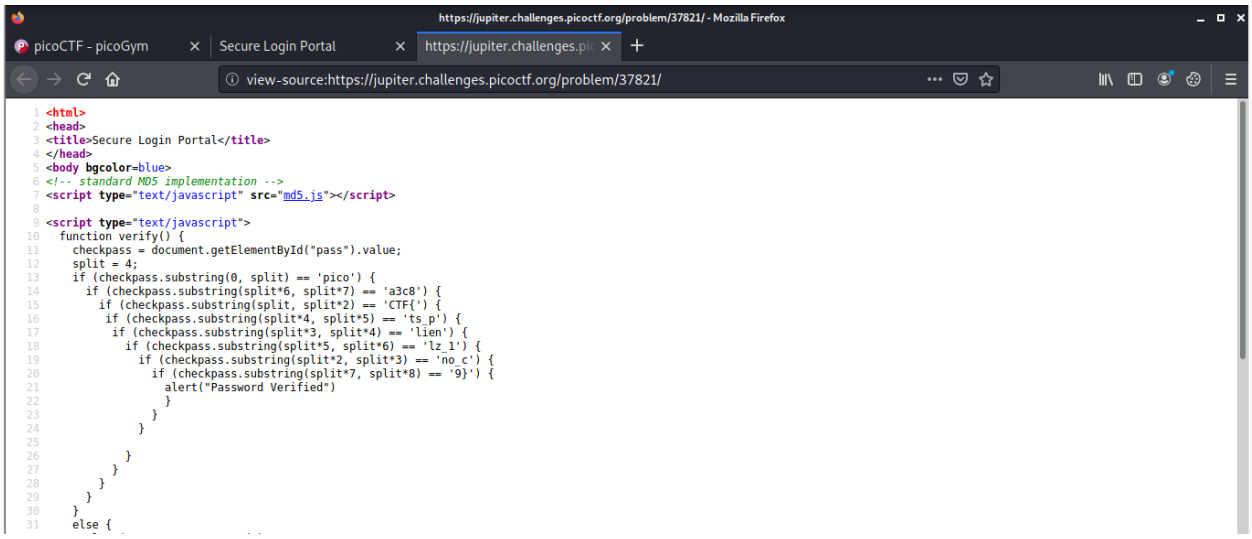
Flag = picoCTF{th3_c0nsp1r4cy_l1v3s_d1c24fef}

DON'T-USE-CLIENT-SIDE

Diberikan sebuah website dengan tampilan seperti berikut, di mana kita diminta untuk menginput kredensial untuk verifikasi :



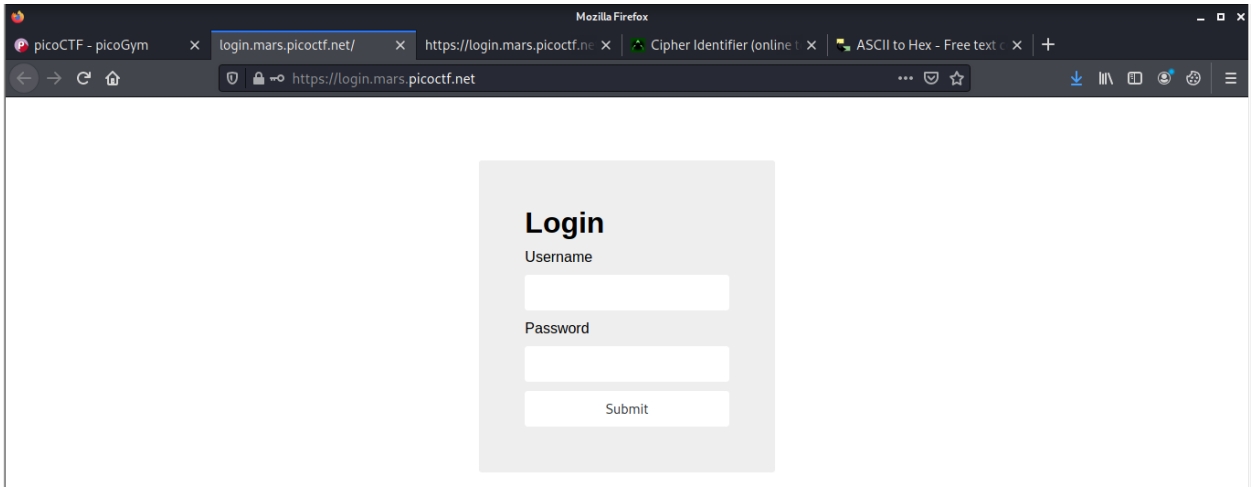
Kemudian, saya coba lihat page source nya ternyata di sana terdapat java script yang kalau diurutkan merupakan flag :



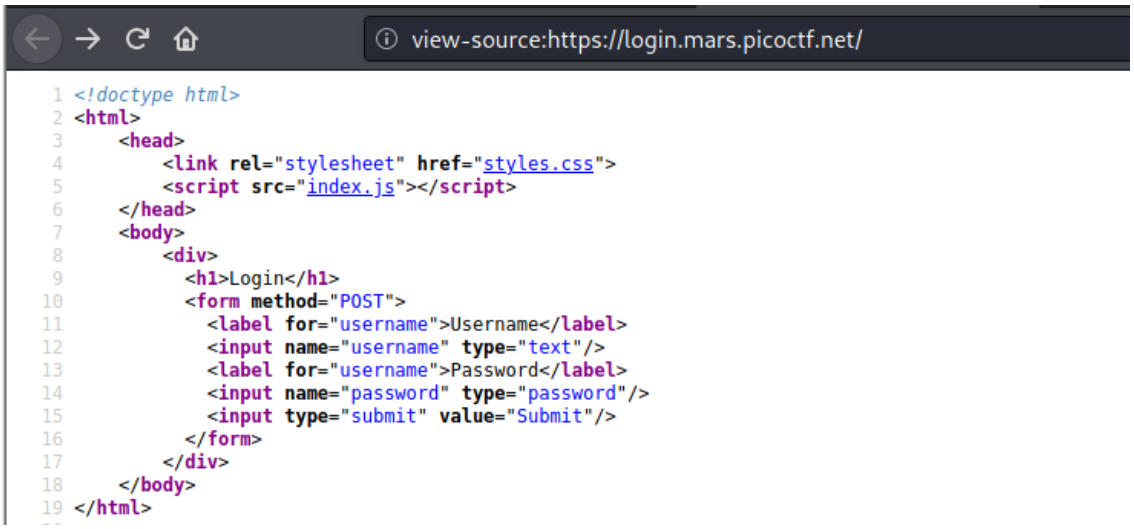
Flag = picoCTF{no_clients_plz_1a3c89}

LOGIN

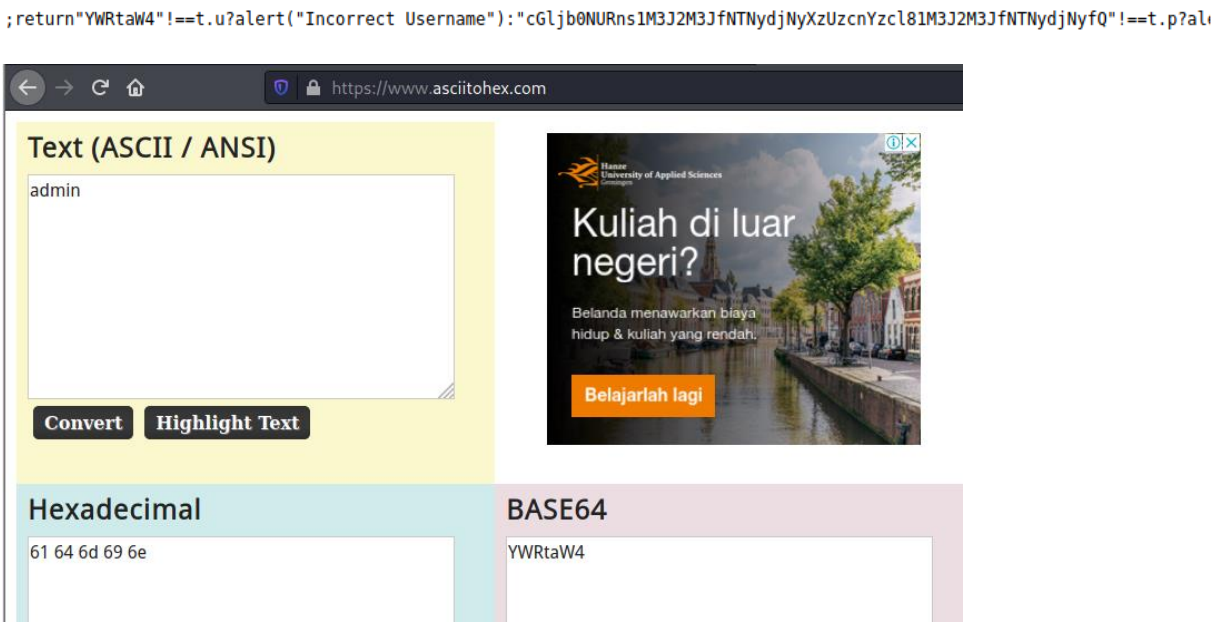
Diberikan sebuah web dengan tampilan login seperti berikut :

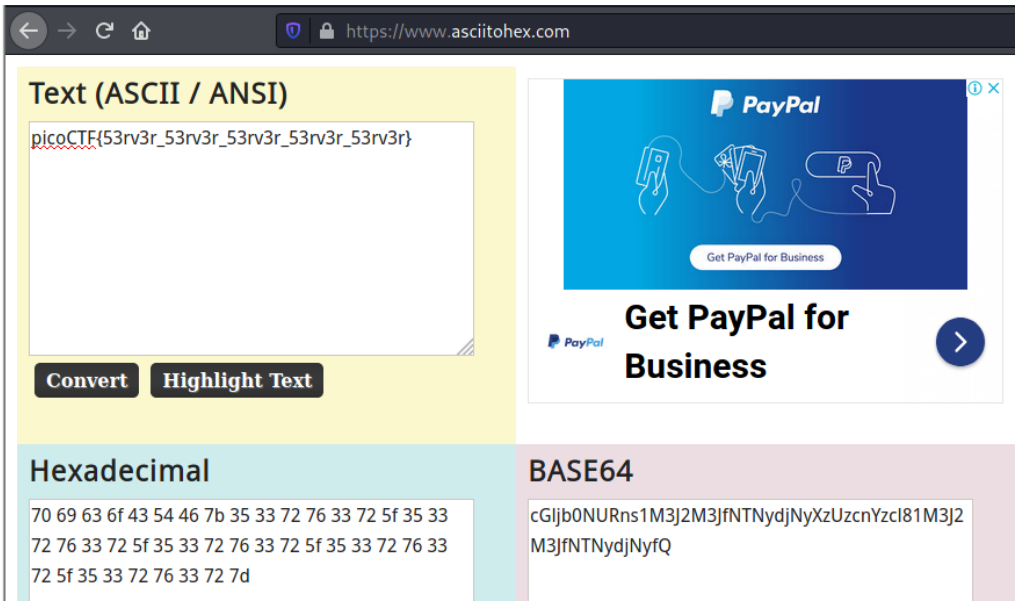


Selanjutnya, saya coba input sembarang karakter dan ternyata tidak berhasil. Lalu, saya coba lihat page source-nya dan di sana terdapat link index.js :

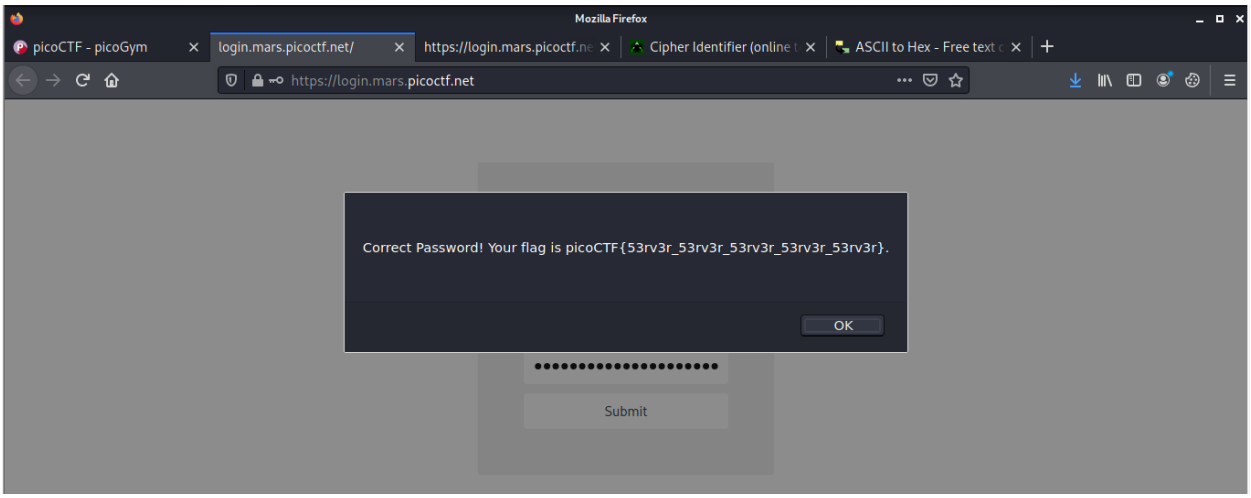


Selanjutnya, saya coba amati halaman index.js tersebut dan saya menemukan format base64 yang saya asumsikan adalah username dan password, lalu langsung saya konversi ke ascii :





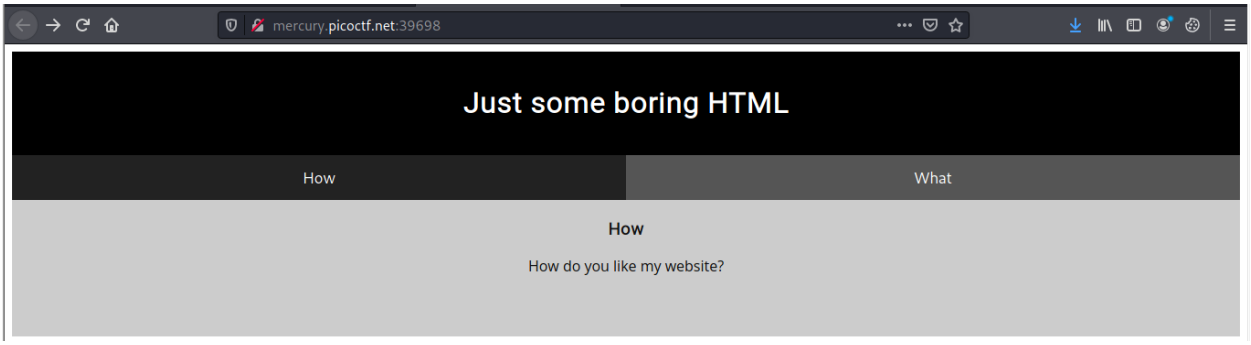
Didapat user admin dengan password picoCTF{53v3r_53v3r_53v3r_53v3r_53v3r}. Sebenarnya, di sini flagnya sudah ketemu, namun untuk memastikan saya coba input ke form login tadi dan ternyata benar itu adalah flag :



Flag = picoCTF{53v3r_53v3r_53v3r_53v3r_53v3r}

SCAVENGER HUNT

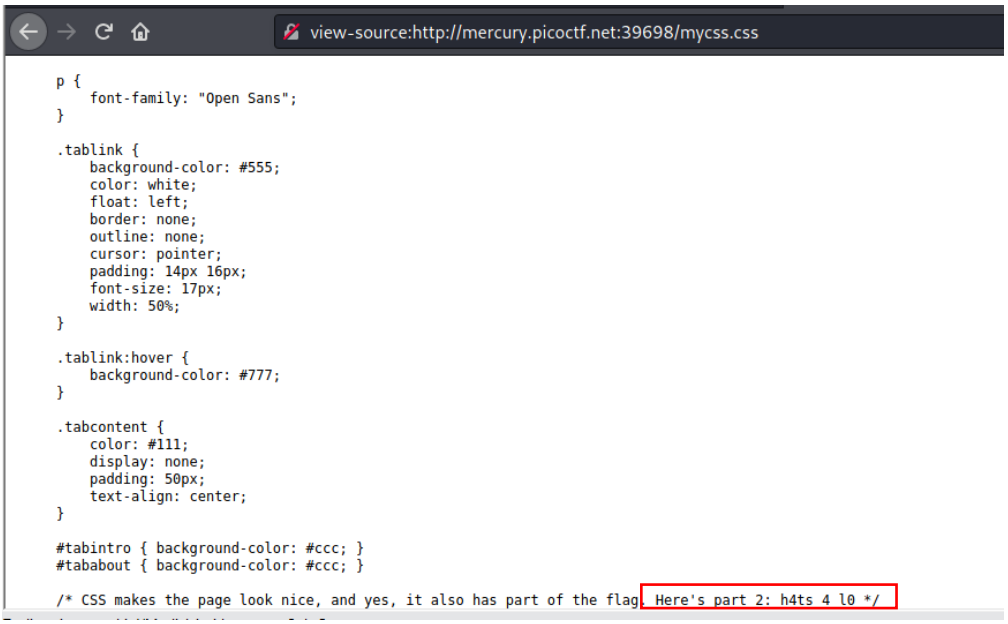
Diberikan website dengan tampilan berikut :



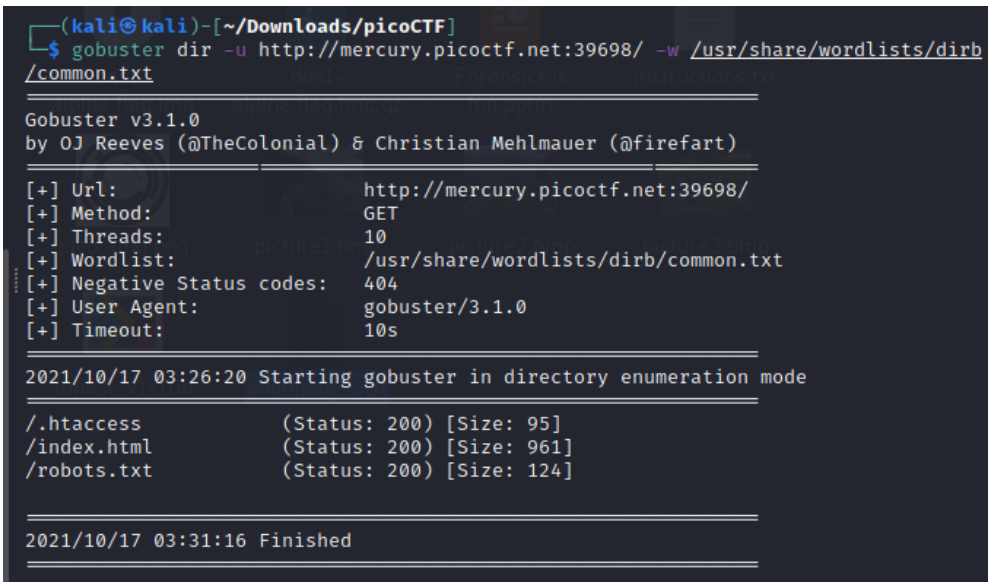
Kemudian, saya coba cek page source halaman tersebut (Ctrl+U) dan terdapat bagian pertama dari flag :

```
<!doctype html>
<html>
  <head>
    <title>Scavenger Hunt</title>
    <link href="https://fonts.googleapis.com/css?family=Open+Sans|Roboto" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="mycss.css">
    <script type="application/javascript" src="myjs.js"></script>
  </head>

  <body>
    <div class="container">
      <header>
        <h1>Just some boring HTML</h1>
      </header>

      <button class="tablink" onclick="openTab('tabintro', this, '#222')" id="defaultOpen">How</button>
      <button class="tablink" onclick="openTab('tababout', this, '#222')">What</button>

      <div id="tabintro" class="tabcontent">
        <h3>How</h3>
        <p>How do you like my website?</p>
      </div>

      <div id="tababout" class="tabcontent">
        <h3>What</h3>
        <p>I used these to make this site: <br/>
          HTML <br/>
          CSS <br/>
          JS (JavaScript)
        </p>
      <!-- Here's the first part of the flag: picoCTF{t -->
      </div>

    </div>

  </body>
</html>
```

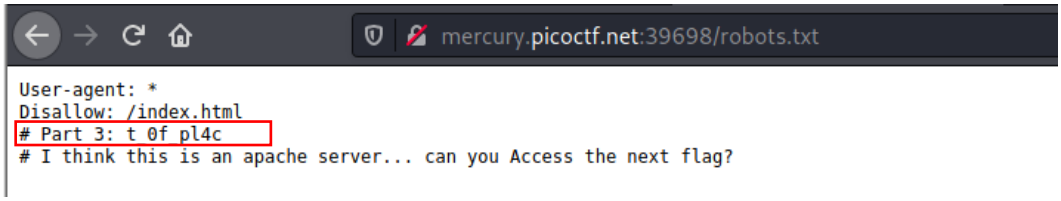
Lalu, saya klik link mycss.css nya dan terdapat bagian kedua dari flag :

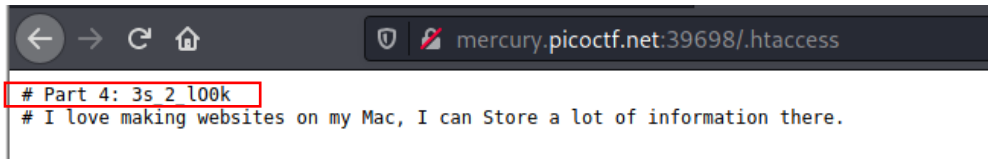


Selanjutnya. saya coba scan menggunakan gobuster untuk melihat directory apa saja yang terdapat di website tersebut, dan saya menemukan :

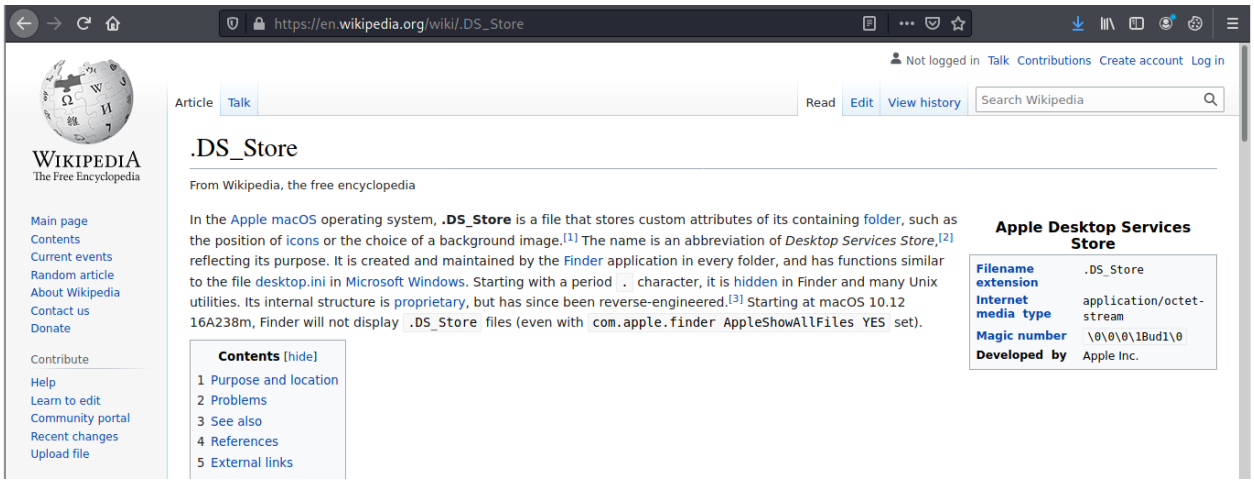


Kemudian, langsung saja saya coba cek satu persatu di website dan ternyata terdapat bagian ketiga dan keempat dari flag :

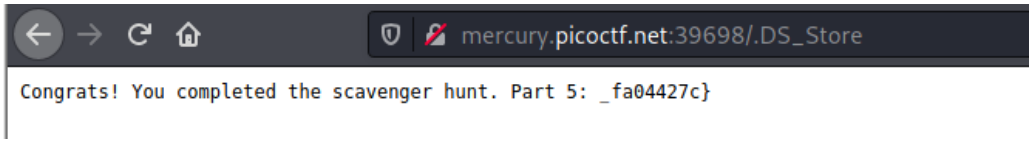




Lalu, di sini saya agak kebingungan karena ini merupakan directory terakhir yang ada, dan flag yang saya dapatkan ternyata masih belum lengkap. Setelah saya baca kembali, di bagian halaman .htaccess dikatakan bahwa si pembuat web membuat web tersebut menggunakan OS Macintosh dan kata "store" menggunakan huruf besar di awal kata. Lalu, saya coba search di google dan saya menemukan informasi dari Wikipedia yang isinya :



Kemudian, saya coba ketikkan .DS_Store di website dan saya mendapat bagian terakhir flag :

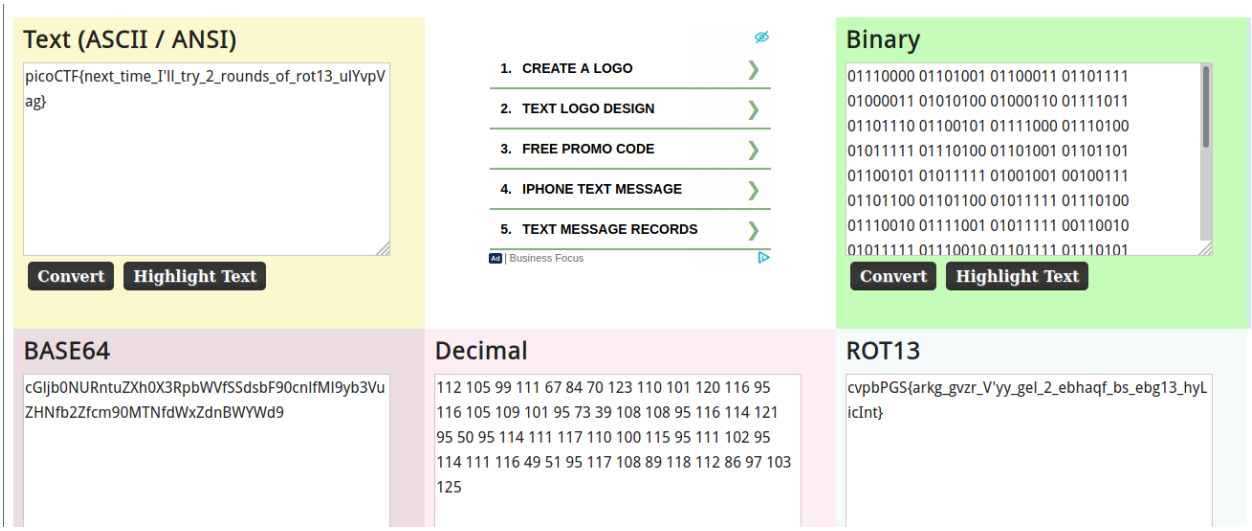


Flag = picoCTF{t4hts_4_l0t_0f_pl4c3s_2_l00k_fa04427c}

[CRYPTOGRAPHY]

MOD 26

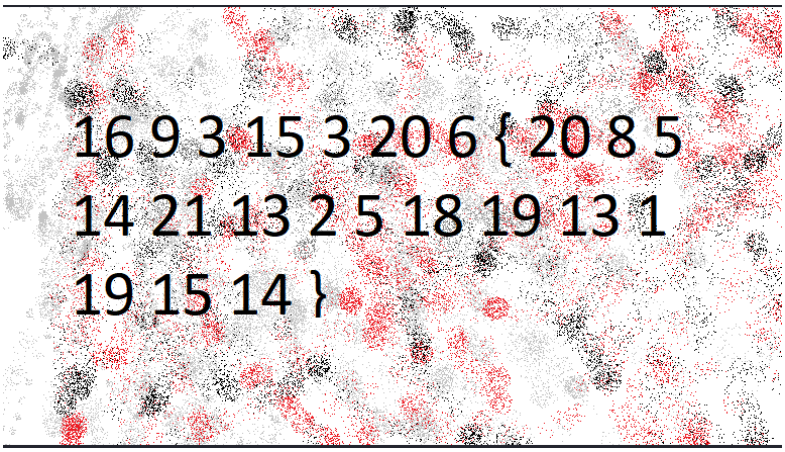
Diberikan kode berupa ROT13, lalu saya konversi ke ascii dan didapat flag :



Flag = picoCTF{next_time_I'll_try_2_rounds_of_rot13_ulYvpVag}

THE NUMBERS

Diberikan file berupa gambar berikut :



Awalnya saya kira nomor tersebut adalah angka decimal, namun ternyata bukan. Lalu saya membuat script python seperti berikut :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/Cryptography]
└─# cat script.py
#!/usr/bin/env python

from string import ascii_uppercase as uppercase

num = [16, 9, 3, 15, 3, 20, 6, "{", 20, 8, 5, 14, 21, 13, 2, 5, 18, 19, 13, 1, 19, 15, 14, "}"]

flag = []

for i in num:
        if type(i) == str:
                flag.append(i)
        else:
                i = i - 1
                flag.append(uppercase[i])

print ''.join(flag)
```

Script diatas bertujuan mengkonversi nomor-nomor yang diberikan tadi supaya menjadi flag (karena hint soal format berupa picoCTF{}) maka untuk mengubah angka 16 menjadi huruf p adalah mengurangi nya dengan angka 1 (karena huruf p merupakan urutan ke 15 dalam alfabet). Dan langsung saya jalankan seperti berikut :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/Cryptography]
└─# ./script.py
PICOCTF{THENUMBERSMASON}
```

Flag = PICOCTF{THENUMBERMASON}

CAESAR

Diberikan sebuah file dengan nama ciphertext dengan isi sebagai berikut :

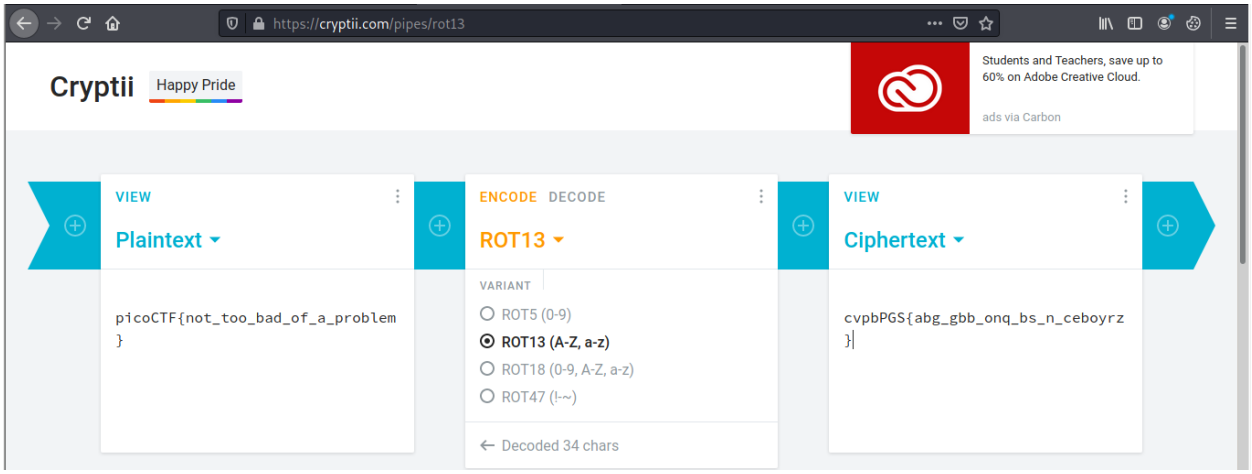
```
┌──(root💀kali)-[/home/…/Downloads/picoCTF/Cryptography/caesar]
└─# cat ciphertext
picoCTF{dspttjohuifsvcjdpoabrkttds}
```

Kemudian, cipher text tersebut saya decode dan ternyata setelah digeser 25 kali muncul kata yang merupakan flag :

Flag = picoCTF{crossingtherubiconzaqjsscr}

13

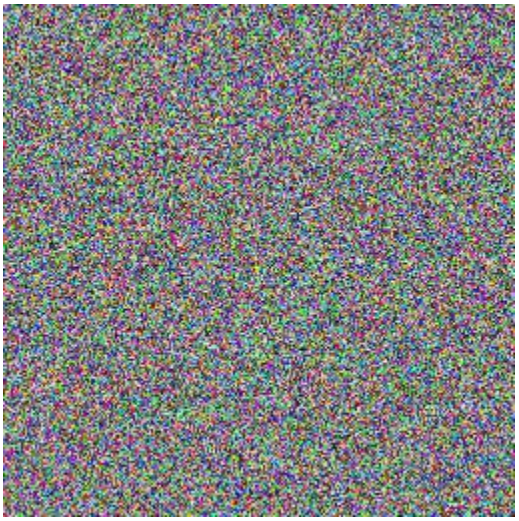
Diberikan sebuah ciphertext dalam bentuk ROT13 langsung saja saya decrypt :

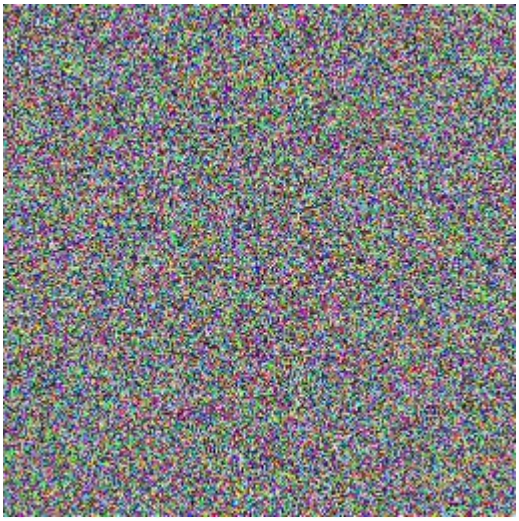


Flag = picoCTF{not_too_bad_of_a_problem}

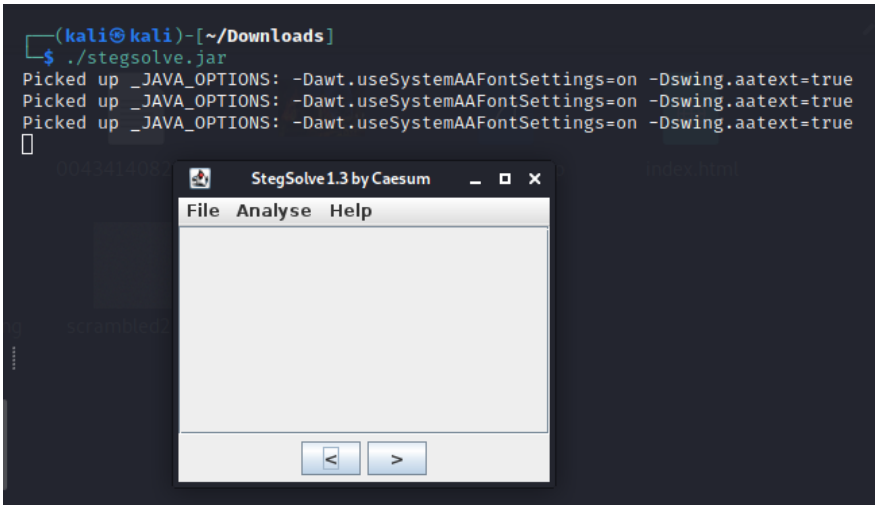
PIXELATED

Diberikan dua gambar dengan tampilan berikut :

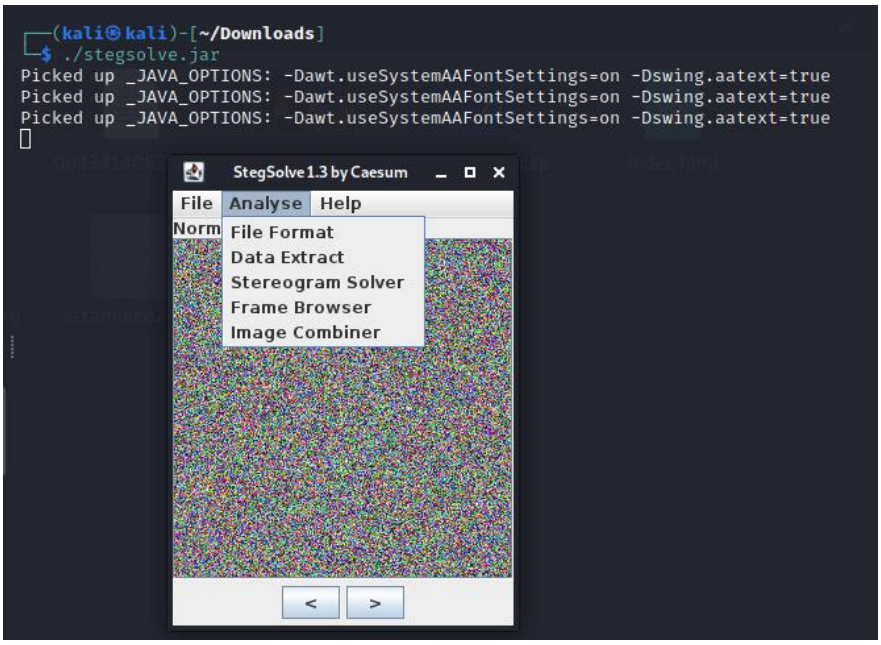




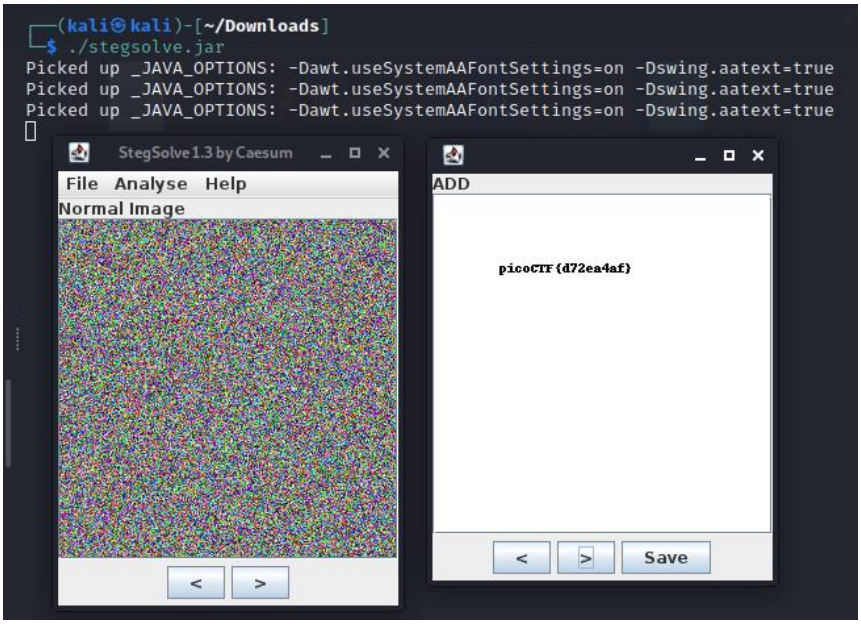
Kemudian sesuai hint soal yaitu visual cryptography saya mencoba menggunakan tools stegsolve untuk menggabungkan kedua gambar tersebut :



Pertama-tama, saya buka terlebih dahulu gambar 1 dan selanjutnya saya menggunakan tools analyze yaitu image combiner dan saya masukkan gambar 2 :



Setelah saya cek satu persatu algoritma nya, saya menemukan flag pada algoritma ADD :



Flag = picoCTF{d72ea4af}

EASY1

Diberikan sebuah ciphertext dengan key-nya, lalu saya coba decrypt menggunakan vigenere cipher online:

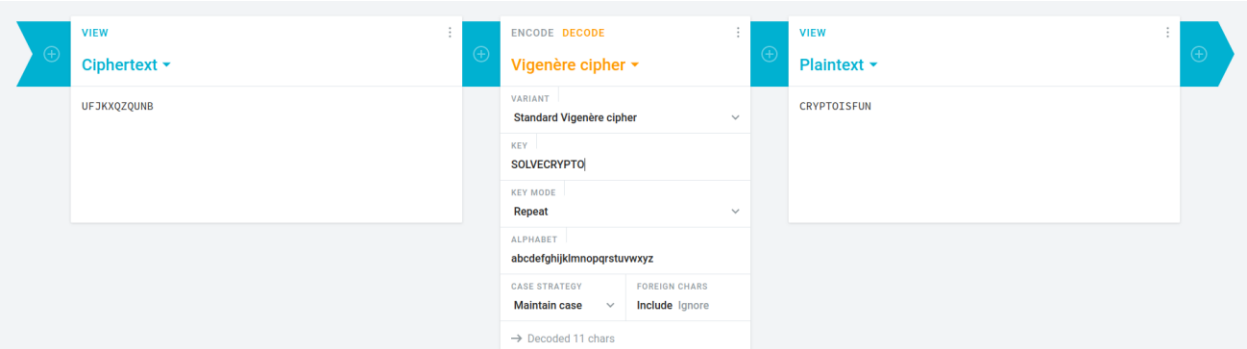
Easy1

Tags: Category: Cryptography

AUTHOR: ALEX FULTON/DANNY

Description

The one time pad can be cryptographically secure, but not when you know the key. Can you solve this? We've given you the encrypted flag, key, and a table to help UFJKXQZQUNB with the key of SOLVECRYPTO. Can you use this table to solve it?.



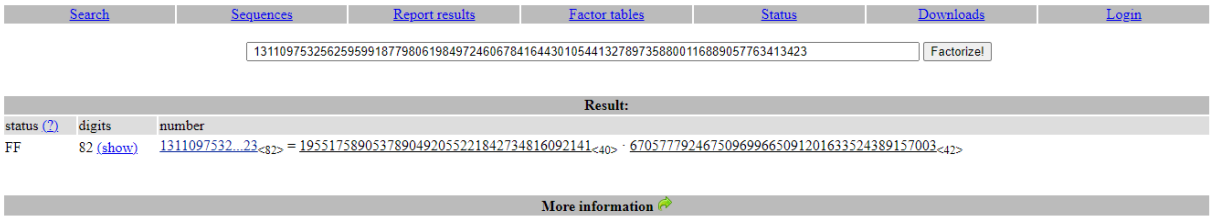
Flag = picoCTF{CRYPTOISFUN}

MIND YOUR Ps & Qs

Diberikan sebuah ciphertext RSA beserta key nya seperti berikut :

```
Decrypt my super sick RSA:
c: 861270243527190895777142537838333832920579264010533029282104230006461420086153423
n: 131109753256259599187798061984972460678416443010544132789735880011688905776341 3423
e: 65537
```

Kemudian nilai n saya faktorkan di http://factordb.com/ untuk mendapat nilai p dan q :



Setelah didapat nilai p dan q nya saya membuat script seperti di bawah untuk melakukan decrypt terhadap ciphertext :

```
┌──(kali㉿kali)-[~/Belajar/exercise/PICO/psqs]
└─$ cat RSA.py
#!/usr/bin/env python3

from Crypto.Util.number import inverse, long_to_bytes

c = 861270243527190895777142537838333832920579264010533029282104230006461420086153423
n = 131109753256259599187798061984972460678416443010544132789735880011688905776341 3423
e = 65537
p = 195517589053789049205522184273481 6092141
q = 670577792467509699665091201633524389157003

phi = (p-1)*(q-1)

d = inverse(e, phi)

m = pow(c,d,n)

print(long_to_bytes(m))
```

Selanjutnya saya jalankan script tersebut dan didapat flag nya :

```
┌──(kali㉿kali)-[~/Belajar/exercise/PICO/psqs]
└─$ ./RSA.py
b'picoCTF{sma11_N_n0_g0od_13686679}'
```

Flag = picoCTF{sma11_N_n0_g00d_13686679}

[FORENSIC]

INFORMATION

Diberikan file berupa gambar, lalu saya coba strings dan ternyata muncul base64 :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/Forensic]
└─# strings cat.jpg
JFIF
0Photoshop 3.0
8BIM
PicoCTF
http://ns.adobe.com/xap/1.0/
<?xpacket begin='
' id='W5M0MpCehiHzreSzNTczkc9d'?>
<x:xmpmeta xmlns:x='adobe:ns:meta/' x:xmptk='Image::ExifTool 10.80'>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <rdf:Description rdf:about=''
  xmlns:cc='http://creativecommons.org/ns#'>
  <cc:license rdf:resource='cGljb0NURnt0aGVfbTN0YWRhdGFfMXNfbW9kaWZpZWR9'/>
 </rdf:Description>
 <rdf:Description rdf:about=''
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:rights>
   <rdf:Alt>
    <rdf:li xml:lang='x-default'>PicoCTF</rdf:li>
   </rdf:Alt>
  </dc:rights>
 </rdf:Description>
</rdf:RDF>
</x:xmpmeta>
```

Lalu, saya decode base64 tersebut di terminal dan didapat flag :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/Forensic]
└─# echo "cGljb0NURnt0aGVfbTN0YWRhdGFfMXNfbW9kaWZpZWR9" | base64 -d
picoCTF{the_m3tadata_1s_modified}
```

Flag = picoCTF{the_m3tadata_1s_modified}

MATRYOSHKA DOLL

Diberikan file berupa gambar, lalu sesuai hint yaitu terdapat file yang disembunyikan di dalam image tersebut. Selanjutnya, saya menggunakan tools binwalk untuk mengesktrak file tersembunyi di dalam image tersebut :

```
┌──(root💀kali)-[/home/kali/Downloads/picoCTF/Forensic]
└─# binwalk -e dolls.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 594 x 1104, 8-bit/color RGBA, non-interlaced
3226          0xC9A           TIFF image data, big-endian, offset of first image directory: 8
272492        0x4286C         Zip archive data, at least v2.0 to extract, compressed size: 378950, uncompressed size:
 383938, name: base_images/2_c.jpg
651608        0x9F158         End of Zip archive, footer length: 22
```

Setelah itu, didapat folder yang didalamnya image lagi kemudian saya menggunakan binwalk kembali dan begitu seterusnya sampai saya mendapat file flag.txt :

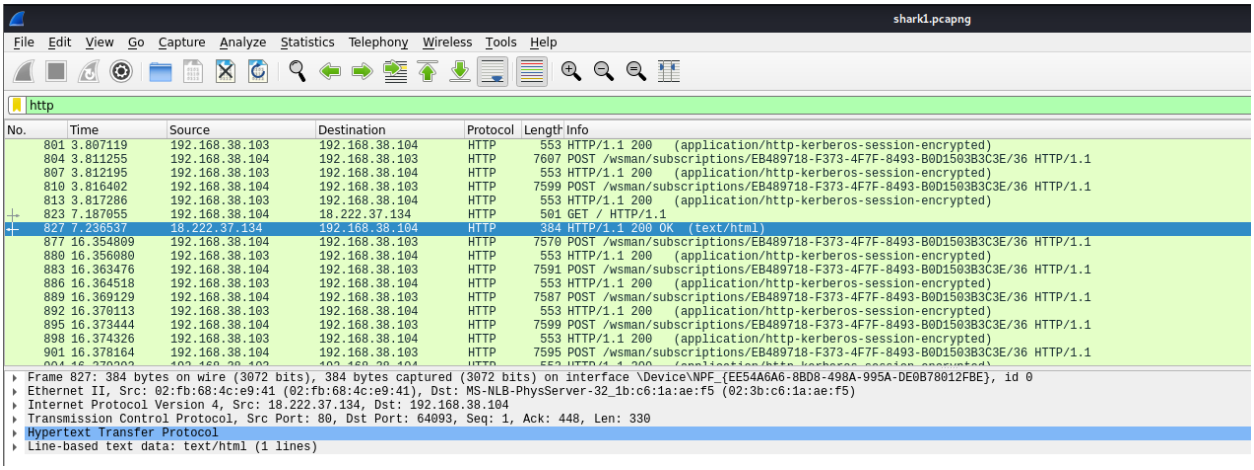
```
┌──(root💀kali)-[/home/…/Downloads/picoCTF/Forensic/_dolls.jpg.extracted]
└─# ls -l
total 376
-rw-r--r-- 1 root root 379138 Oct 14 02:01 4286C.zip
drwxr-xr-x 2 root root   4096 Oct 14 02:01 base_images

┌──(root💀kali)-[/home/…/Downloads/picoCTF/Forensic/_dolls.jpg.extracted]
└─# cd base_images

┌──(root💀kali)-[/home/…/picoCTF/Forensic/_dolls.jpg.extracted/base_images]
└─# ls -l
total 376
-rw-r--r-- 1 root root 383938 Mar 15  2021 2_c.jpg

┌──(root💀kali)-[/home/…/picoCTF/Forensic/_dolls.jpg.extracted/base_images]
└─# binwalk -e 2_c.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             PNG image, 526 x 1106, 8-bit/color RGBA, non-interlaced
3226          0×C9A           TIFF image data, big-endian, offset of first image directory: 8
187707        0×2DD3B         Zip archive data, at least v2.0 to extract, compressed size: 196043, uncompressed size:
 201445, name: base_images/3_c.jpg
383805        0×5DB3D         End of Zip archive, footer length: 22
383916        0×5DBAC         End of Zip archive, footer length: 22


┌──(root💀kali)-[/home/…/picoCTF/Forensic/_dolls.jpg.extracted/base_images]
└─# ls
2_c.jpg  _2_c.jpg.extracted
```

```
┌──(kali㉿kali)-[~/…/base_images/_3_c.jpg.extracted/base_images/_4_c.jpg.extracted]
└─$ ls
136DA.zip  flag.txt

┌──(kali㉿kali)-[~/…/base_images/_3_c.jpg.extracted/base_images/_4_c.jpg.extracted]
└─$ cat flag.txt
picoCTF{4f11048e83ffc7d342a15bd2309b47de}
```

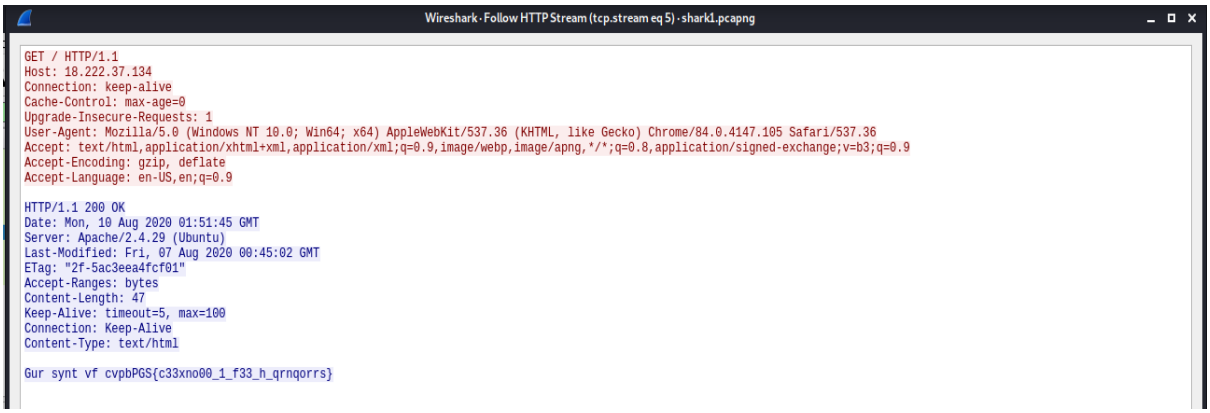
Flag = picoCTF{4f11048e83ffc7d342a15bd2309b47de}

WIRESHARK DOO….. DOO… DOOO….

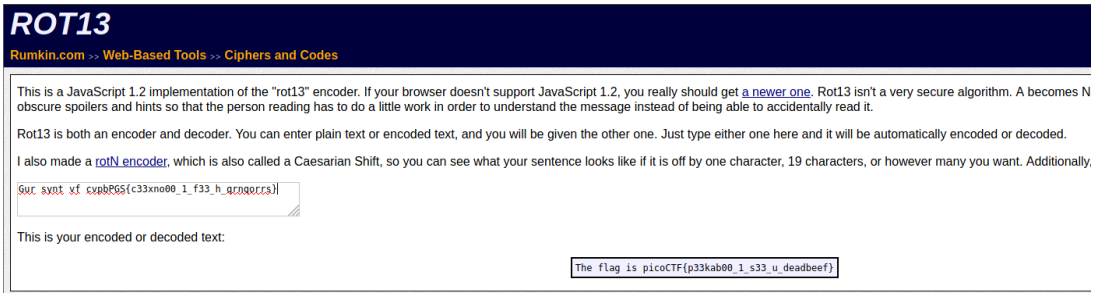
Diberikan sebuah file hasil capture packet, saya buka dan saya filter berdasarkan protocol HTTP :



Kemudian, di sini saya mendapat packet yang cukup menarik perhatian saya karena berisi text html. Selanjutnya saya menggunakan fiture follow HTTP stream dan muncul seperti berikut :



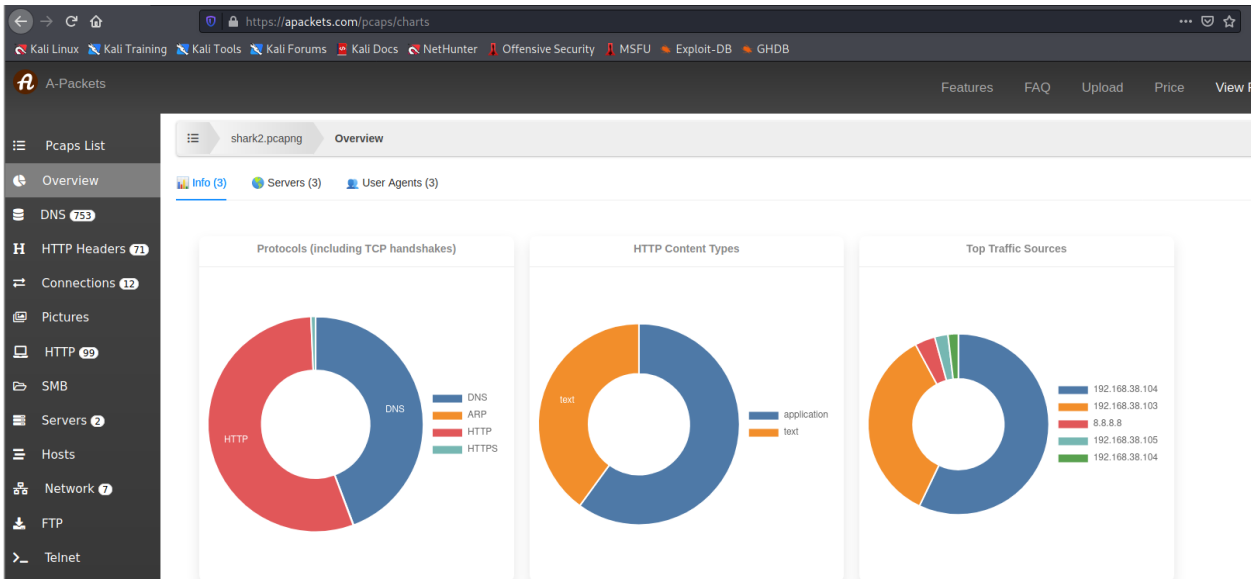
Di bagian akhir packet response terdapat kode yang saya asumsikan adalah ROT13. Lalu saya langsung decode dan didapat flagnya :



Flag = picoCTF{p33kab00_1_s33_u_deadbeef}

WIRESHARK TWOO… TWOO… TWOOO…

Diberikan sebuah file capture packet, pertama saya buka menggunakan wireshark dan ternyata terdapat banyak sekali flag palsu. Di sini saya kebingungan cukup lama sampai saya menemukan tool analisis pcap online di https://apackets.com/ :



Terlihat bahwa terdapat dua protocol yang dominan, yaitu HTTP dan DNS. Karena protocol HTTP sudah saya cek di wireshark dan ternyata tidak ada apa-apa, saya langsung melihat ke bagian protocol DNS dan saya menemukan trafik data ke sebuah alamat domain :

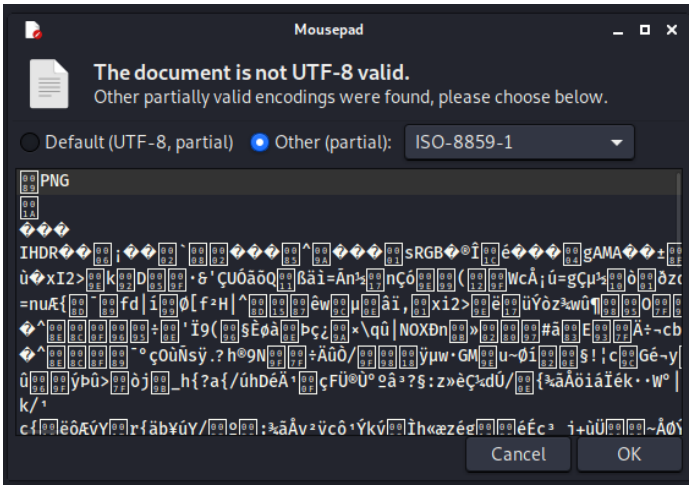
192.168.38.104	18.217.1.57	cGljb0NU.reddshrimpandherring.com cGljb0NU.reddshrimpandherring.com.us-west-1.ec2-utilities.amazonaws.com cGljb0NU.reddshrimpandherring.com.windomain.local RntkbnNf.reddshrimpandherring.com RntkbnNf.reddshrimpandherring.com.us-west-1.ec2-utilities.amazonaws.com RntkbnNf.reddshrimpandherring.com.windomain.local M3hmMWxf.reddshrimpandherring.com M3hmMWxf.reddshrimpandherring.com.us-west-1.ec2-utilities.amazonaws.com M3hmMWxf.reddshrimpandherring.com.windomain.local ZnR3X2Rl.reddshrimpandherring.com ZnR3X2Rl.reddshrimpandherring.com.us-west-1.ec2-utilities.amazonaws.com ZnR3X2Rl.reddshrimpandherring.com.windomain.local YWRiZWVm.reddshrimpandherring.com YWRiZWVm.reddshrimpandherring.com.us-west-1.ec2-utilities.amazonaws.com YWRiZWVm.reddshrimpandherring.com.windomain.local fQ==.reddshrimpandherring.com fQ==.reddshrimpandherring.com.us-west-1.ec2-utilities.amazonaws.com fQ==.reddshrimpandherring.com.windomain.local

Lalu, saya tertarik pada setiap kata awal subdomain yang terlihat seperti base64 dan saya coba menggabungkannya dan dikonversi ternyata itu adalah flag :

Flag = picoCTF{dns_3xf1l_ftw_deadbeef}

EXTENSIONS

Diberikan sebuah file tanpa ekstensi, kemudian setelah saya buka di bagian header terdapat kata PNG yang mengindikasikan file tersebut adalah file PNG :



Setelah itu, saya rename file tersebut dengan ekstensi png dan saya buka kembali dan muncul flag nya :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic]
└─$ mv flag.txt flag.png

┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic]
└─$ open flag.png
```

picoCTF{now_you_know_about_extensions}

Flag = picoCTF{now_you_know_about_extensions}

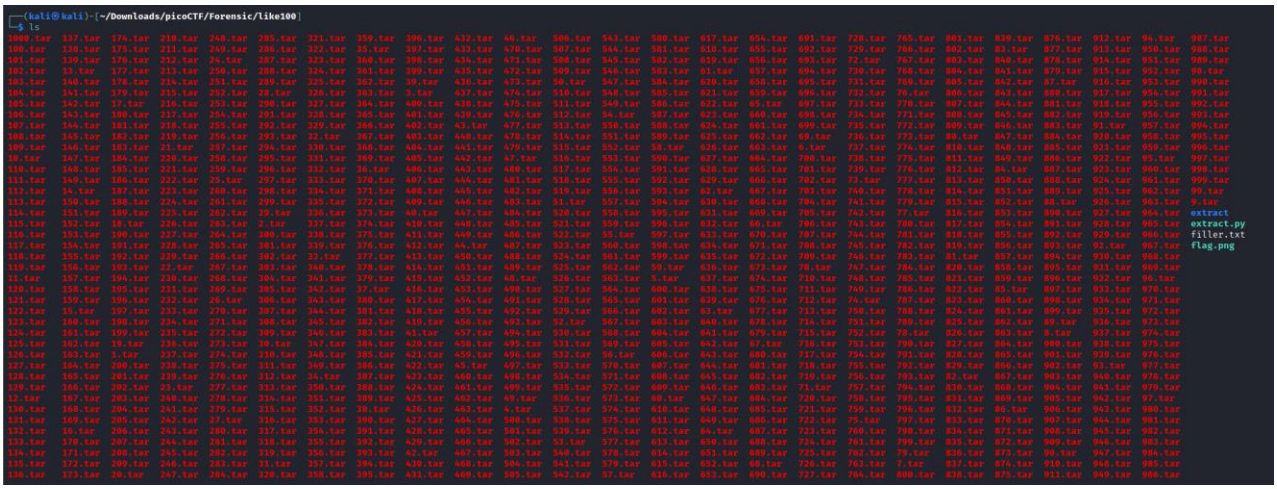
LIKE1000

Diberikan sebuah file berekstensi tar, lalu saya membuat script python seperti berikut :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/like100]
└─$ cat extract.py
#!/usr/bin/env python3

import tarfile

for i in range(1000, 0, -1):
        my_tar = str(i) + '.tar'
        print ('[*]', my_tar)

        my_tar = tarfile.open(my_tar)
        my_tar.extractall('./')
        my_tar.close()
```

Script tersebut bertujuan untuk mengekstrak file tar sebanyak 1000 kali (sesuai hint pada nama soal) lalu saya jalankan dan didapat :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/like100]
└─$ ./extract.py
[*] 1000.tar
[*] 999.tar
[*] 998.tar
[*] 997.tar
[*] 996.tar
[*] 995.tar
[*] 994.tar
[*] 993.tar
[*] 992.tar
[*] 991.tar
[*] 990.tar
[*] 989.tar
[*] 988.tar
[*] 987.tar
[*] 986.tar
[*] 985.tar
[*] 984.tar
[*] 983.tar
[*] 982.tar
[*] 981.tar
[*] 980.tar
[*] 979.tar
[*] 978.tar
[*] 977.tar
[*] 976.tar
[*] 975.tar
```



Kemudian, untuk mencari yang mana merupakan file berisi flag saya coba hapus file yang berekstensi tar dengan menggunakan command rm :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/like100]
└─$ rm -r *".tar"
```

Lalu, saya coba lihat lagi dan ternyata terdapat file flag yang berupa gambar :

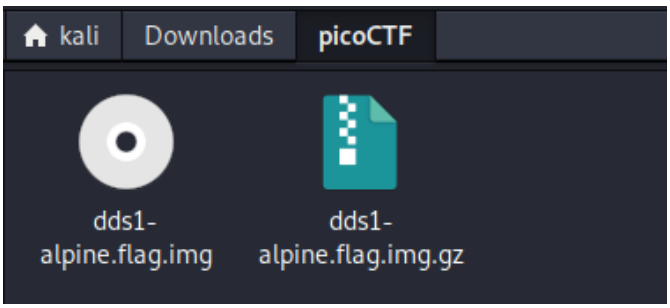
```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/like100]
└─$ ls -lah
total 60K
drwxr-xr-x 3 kali kali  28K Oct 14 23:05 .
drwxr-xr-x 4 kali kali 4.0K Oct 14 23:01 ..
drwxr-xr-x 2 kali kali 4.0K Oct 14 23:00 extract
-rwxrwxrwx 1 kali kali  192 Oct 14 23:00 extract.py
-rw-r--r-- 1 kali kali   27 Aug  4  2019 filler.txt
-rwxrw-rw- 1 kali kali  13K Aug  4  2019 flag.png
```

picoCTF{l0t5_0f_TAR5}

Flag = picoCTF{l0t5_0f_TAR5}

DISK, DISK, SLEUTH!

Diberikan sebuah file zip saya coba ekstrak dan file tersebut adalah sebuah file iso/image disk :



Kemudian, saya melakukan strings terhadap file tersebut dan saya grep sesuai format flag :

```
┌──(root💀 kali)-[/home/kali/Downloads/picoCTF]
└─# strings dds1-alpine.flag.img | grep pico
ffffffff81399ccf t pirq_pico_get
ffffffff81399cee t pirq_pico_set
ffffffff820adb46 t pico_router_probe
  SAY picoCTF{f0r3ns1c4t0r_n30phyt3_a6f4cab5}
```

Flag = picoCTF{f0r3ns1c4t0r_n30phyt3_a6f4cab5}

DISK, DISK, SLEUTH! II

Diberikan sebuah file disk image, lalu saya download dan soal memberikan hint berupa link berikut http://wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview . Setelah saya pahami saya menemukan command mmls yang berfungsi untuk menampilkan layout/partisi dari sebuah file disk images :

Volume System Tools

These tools take a disk (or other media) image as input and analyze its partition structures. Examples include DOS partitions, BSD disk labels, and the Sun Volume Table of Contents (VTOC). These can be used find hidden data between partitions and to identify the file system offset for The Sleuth Kit tools. The media management tools support DOS partitions, BSD disk labels, Sun VTOC, and Mac partitions.

- mmls: Displays the layout of a disk, including the unallocated spaces.
- mmstat: Display details about a volume system (typically only the type).
- mmcat: Extracts the contents of a specific volume to STDOUT.

Setelah saya jalankan saya mendapat sebuah partisi yang dialokasikan untuk Linux :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/Disk 2]
└─$ mmls dds2-alpine.flag.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  ------    0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000262143   0000260096   Linux (0x83)
```

Kemudian untuk melihat apa saja yang terdapat di dalam partisi tersebut saya menggunakan command fls untuk melihat list file :

File Name Layer Tools

These file system tools process the file name structures, which are typically located in the parent directory.

- ffind: Finds allocated and unallocated file names that point to a given meta data structure.
- fls: Lists allocated and deleted file names in a directory.

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/Disk 2]
└─$ fls -o 2048 dds2-alpine.flag.img
d/d 26417:      home
d/d 11: lost+found
r/r 12: .dockerenv
d/d 20321:      bin
d/d 4065:       boot
d/d 6097:       dev
d/d 2033:       etc
d/d 8129:       lib
d/d 14225:      media
d/d 16257:      mnt
d/d 18289:      opt
d/d 16258:      proc
d/d 18290:      root
d/d 16259:      run
d/d 18292:      sbin
d/d 12222:      srv
d/d 16260:      sys
d/d 18369:      tmp
d/d 12223:      usr
d/d 14229:      var
V/V 32513:      $OrphanFiles
```

Setelah itu, terdapat beberapa direktori dan di sini saya tertarik pada direktoi root karena berpotensi mengandung flag. Selanjutnya untuk melihat apa saja isi dari direktori root tersebut saya menggunakan command fls lagi seperti gambar di bawah :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/Disk 2]
└─$ fls -o 2048 dds2-alpine.flag.img 18290
r/r 18291:      down-at-the-bottom.txt
```

Lalu saya menemukan file txt dan saya menggunakan command icat untuk membaca isi file tersebut :

Meta Data Layer Tools

These file system tools process the meta data structures, which store the details about a file. Examples of this structure include directory entries in FAT, MFT entries in NTFS, and inodes in ExtX and UFS.

- icat: Extracts the data units of a file, which is specified by its meta data address (instead of the file name).
- ifind: Finds the meta data structure that has a given file name pointing to it or the meta data structure that points to a given data unit.
- ils: Lists the meta data structures and their contents in a pipe delimited format.
- istat: Displays the statistics and details about a given meta data structure in an easy to read format.

```
┌──(kali㉿kali)-[~/Downloads/picoCTF/Forensic/Disk 2]
└─$ icat -o 2048 dds2-alpine.flag.img 18291
 _     _     _     _     _     _     _     _     _     _     _     _     _
/ \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \
( p ) ( i ) ( c ) ( o ) ( C ) ( T ) ( F ) ( { ) ( f ) ( 0 ) ( r ) ( 3 ) ( n )
 \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/

 _     _     _     _     _     _     _     _     _     _     _     _     _
/ \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \
( s ) ( 1 ) ( c ) ( 4 ) ( t ) ( 0 ) ( r ) ( _ ) ( n ) ( 0 ) ( v ) ( 1 ) ( c )
 \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/

 _     _     _     _     _     _     _     _     _     _     _
/ \   / \   / \   / \   / \   / \   / \   / \   / \   / \   / \
( 3 ) ( _ ) ( f ) ( 5 ) ( 5 ) ( 6 ) ( 5 ) ( e ) ( 7 ) ( b ) ( } )
 \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/   \_/
```

Flag = picoCTF{f0r3ns1c4t0r_n0v1c3_f5565e7b}

WHAT LIES WITHIN

Diberikan sebuah file berupa gambar berikut :

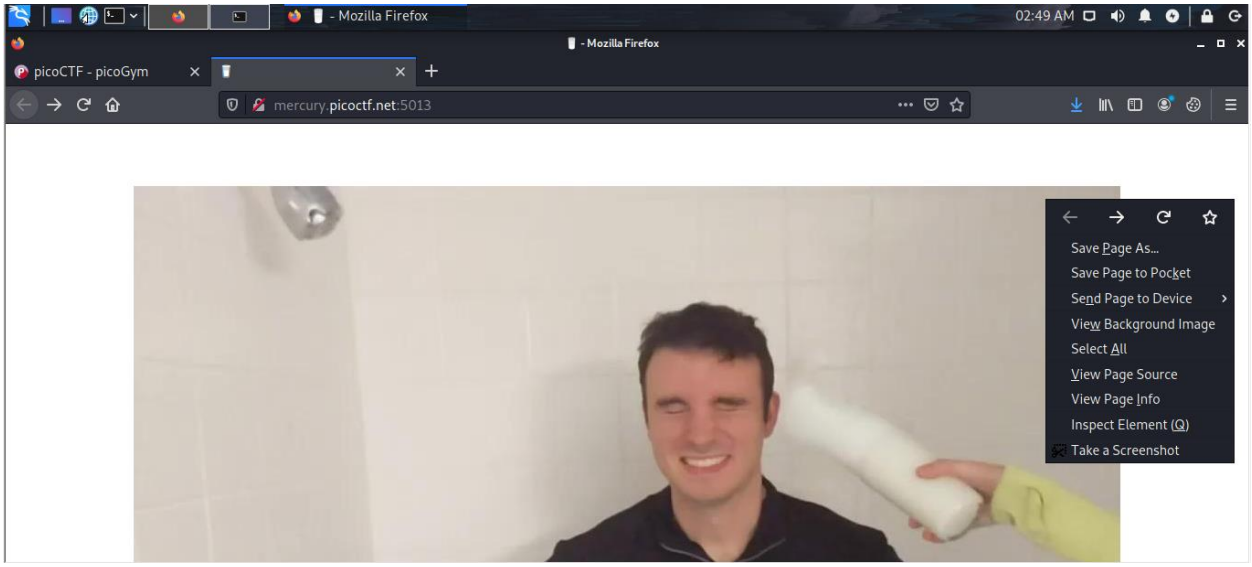
Lalu, sesuai hint nama soal terdapat flag yang tersembunyi di dalam gambar, dan saya coba menggunakan tool steghide namun perlu passphrase untuk mengekstraknya. Kemudian saya mencoba lagi menggunakan tools zsteg dan saya menemukan flag :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF]
└─$ zsteg buildings.png
b1,r,lsb,xy         .. text: "^5>R5YZrG"
b1,rgb,lsb,xy       .. text: "picoCTF{h1d1ng_1n_th3_b1t5}"
b1,abgr,msb,xy      .. file: PGP Secret Sub-key -
b2,b,lsb,xy         .. text: "XuH}p#8Iy="
b3,abgr,msb,xy      .. text: "t@Wp-_tH_v\r"
b4,r,lsb,xy         .. text: "fdD\"\"\"\" "
b4,r,msb,xy         .. text: "%Q#gpSv0c05"
b4,g,lsb,xy         .. text: "fDfffDD\"\""
b4,g,msb,xy         .. text: "f\"fff\"\"DD"
b4,b,lsb,xy         .. text: "\"$BDDDDf"
b4,b,msb,xy         .. text: "wwBDDDfUU53w"
b4,rgb,msb,xy       .. text: "dUcv%F#A`"
b4,bgr,msb,xy       .. text: " V\"c7Ga4"
b4,abgr,msb,xy      .. text: "gOC_$_@o"
```

Flag = picoCTF{h1d1ng_1n_th3_b1t5}

MILKSLAP

Diberikan sebuah website dengan sebuah gif sebagai tampilannya :



Lalu, saya coba simpan file gif tersebut ke local dan saya menggunakan zsteg untuk menemukan flagnya :

```
┌──(kali㉿kali)-[~/Downloads/picoCTF]
└─$ zsteg concat_v.png
imagedata           .. text: "\n\n\n\n\n\n\t\t"
b1,b,lsb,xy         .. text: "picoCTF{imag3_m4n1pul4t10n_sl4p5}\n"
b1,bgr,lsb,xy       .. <wbStego size=9706075, data="\xB6\xAD\xB6}\xDB\xB2lR\x7F\xD
F\x86\xB7c\xFC\xFF\xBF\x02Zr\x8E\xE2Z\x12\xD8q\xE56MJ-X:\xB5\xBF\xF7\x7F\xDB\xDFI\
bm\xDB\xDB\x80m\x00\x00\x00\xB6m\xDB\xDB\xB6\x00\x00\x00\xB6\xB6\x00m\xDB\x12\x12m
\xDB\xDB\x00\x00\x00\x00\x00\xB6m\xDB\x00\xB6\x00\x00\x00\xDB\xB6mm\xDB\xB6\xB6\x0
0\x00\x00\x00\x00m\xDB", even=true, mix=true, controlbyte="[">
b2,r,lsb,xy         .. file: SoftQuad DESC or font file binary
b2,r,msb,xy         .. file: VISX image file
b2,g,lsb,xy         .. file: VISX image file
b2,g,msb,xy         .. file: SoftQuad DESC or font file binary - version 15722
b2,b,msb,xy         .. text: "UfUUUU@UUU"
b4,r,lsb,xy         .. text: "\"\"\"\"\"#4D"
b4,r,msb,xy         .. text: "wwww3333"
b4,g,lsb,xy         .. text: "wewwwwvUS"
b4,g,msb,xy         .. text: "\"\"\"\"DDDD"
b4,b,lsb,xy         .. text: "vdUeVwweDFw"
b4,b,msb,xy         .. text: "UUYYUUUUUUUU"
```

Flag = picoCTF{imag3_m4n1pul4t10n_sl4p5}

SO META

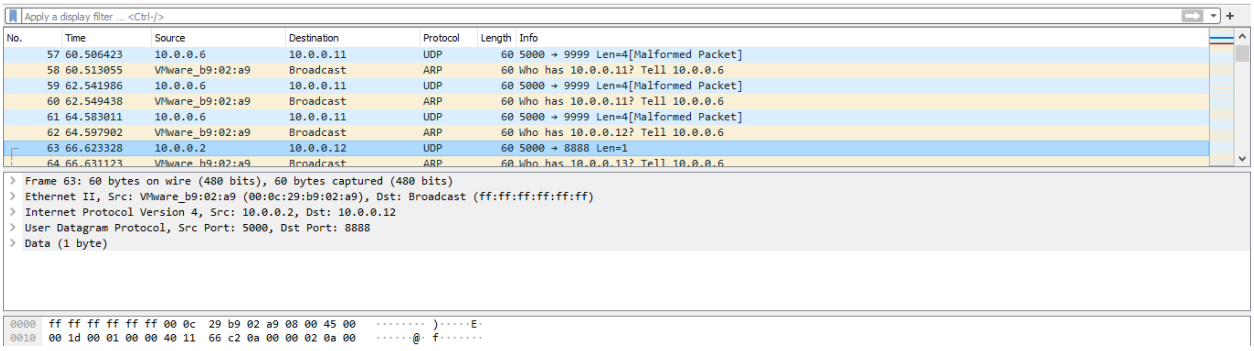
Diberikan sebuah file gambar, saya menggunakan exiftool untuk melihat informasi gambar :

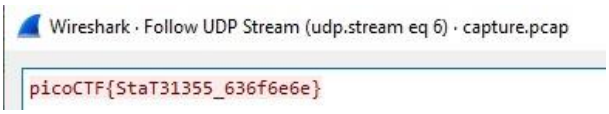
```
┌──(kali㉿kali)-[~/Downloads/picoCTF]
└─$ exiftool pico_img.png
ExifTool Version Number         : 12.30
File Name                       : pico_img.png
Directory                       : .
File Size                       : 106 KiB
File Modification Date/Time     : 2021:10:17 00:21:21-04:00
File Access Date/Time           : 2021:10:17 00:21:55-04:00
File Inode Change Date/Time     : 2021:10:17 00:21:55-04:00
File Permissions                : -rw-r--r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 600
Image Height                    : 600
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Software                        : Adobe ImageReady
XMP Toolkit                     : Adobe XMP Core 5.3-c011 66.145661, 2012/02/06-14:56:27
Creator Tool                    : Adobe Photoshop CS6 (Windows)
Instance ID                     : xmp.iid:A5566E73B2B811E8BC7F9A4303DF1F9B
Document ID                     : xmp.did:A5566E74B2B811E8BC7F9A4303DF1F9B
Derived From Instance ID        : xmp.iid:A5566E71B2B811E8BC7F9A4303DF1F9B
Derived From Document ID        : xmp.did:A5566E72B2B811E8BC7F9A4303DF1F9B
Warning                         : [minor] Text/EXIF chunk(s) found after PNG IDAT (may be ignored by some readers)
Artist                          : picoCTF{s0_m3ta_fec06741}
Image Size                      : 600×600
Megapixels                      : 0.360
```

Flag = picoCTF{s0_m3ta_fec06741}

SHARK ON WIRE 1

Diberikan sebuah file capture packet, lalu saya buka menggunakan wireshark, Kemudian sesuai hint soal yaitu “what are streams?” di sini saya menyimpulkan bahwa kita harus menggunakan fitur follow stream untuk melihat flag nya. Lalu, saya coba cek satu persatu protocol yang bisa di follow stream nyadan saya menemukan protocol UDP pada packet no 63 yang berisi flag :





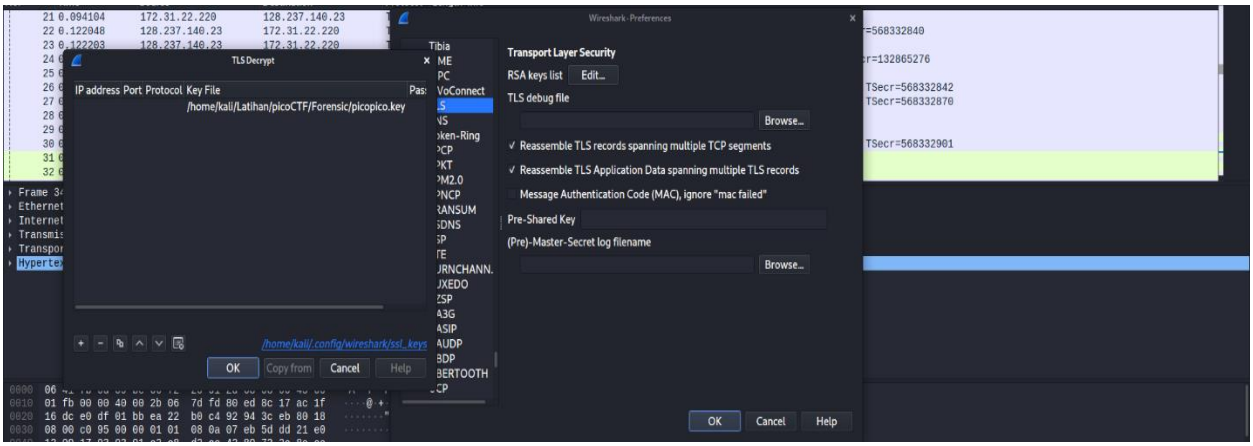
Flag = picoCTF{StaT31355_636f6e6e}

WEBNET0

Diberikan sebuah file packet capture dan key seperti berikut :

```
┌──(kali㉿kali)-[~/Latihan/picoCTF/Forensic]
└─$ cat picopico.key
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCwKlFPNKjseJF5
puCJU5×38XcT1eQge5zOKNahAlYudvGVOEs61TnIgvcER4ko8i3OCwak2/atcGk3
oz9jFKep7XFEYNP31IwwD9j/YazlKy4DRLGObOyIZUU1f2WRA7Uhf0POQXsDT1oU
X32jMKZkQSSDW4MRZd9trJYdO2TrcEPMsBiZQlFlvgnNwl3QlawozTHLAJKI36j1
cPwSMMeNca1e0Zi1s7R5IxfhpNXOBF0FmxiWvmeOHbaspyHg8UEmGBrkd4k4wXSK
GQvrc8QjycP4ScEdquxJiYnDT8iEbAq70/7f/5NIN1DE9YoGJqKYjTS9nRPB4Yvj
JN/SJnhvAgMBAAECggEACCnd3LrG/TZVH3sROqvqO1CwQPYPfUXdLVyNHab7EWon
pc+XBOHurJENG2CpRYF7h+nQ5ADhfIYSCicBf/jsEB7VueJ20CxEVtHVL3h6R6Bp
oHMle0Em8OcofuMpdL/kO+om3T8BkVSzCvCl5NMTUuAF7iRmfX7oDLALwM0IzzQv
2un+2UmT15rgAZfl3IL1PGvJhbhLxfeeyPE9MBy1SqBjQ9rNFn8sQv959J6BHz4b
EpK//ErtNP2yh7oiVBBgKEQ1gEuOjQC/4oxoqCFfZaf9XNRCxB/zY1nUprvJyz09
NMQWNF2EmvmBVGfoTxmuut5N0GbVr2UyHxWMKm2sOQKBgQDpb2+AWgWlGtetuLKJ
fJs8dnd6LhnafbKCOXMOT68qMBRoTpBtVTLRVSNvWCm8m4TTEazX4+ZA+bJFwUFw
aATDmHcr6lMI3tNKrcsnY2F7o5I4z6mwuRuSeszq/ndxZqCzwCu4nKixh3cznp7j
JiElNG0d8Lu5eQgmVAK1AhWXfQKBgQDBMa9ga7VJUP4pzcHnWAoi34OpfjvQYeGl
IKL3AKO4OedaHdH9qid41PQHnL7O3xzN669SkLZ5s0d88A/LFLk4oZNMKdkSTQIQ
+AMbXH01HGFvnCOuPg/FbNp1wS7zJEg5u5HFQWyMPNJLr/hZ6g2Yp+UGpAcGTwM/
RCPVAPhLWwKBgQDAB0OaOnPaVjKGXiHAqBirrGiswa/S5QQrzEaxxys5cUPYaoi0
6BldysPTnJr45JZna2rcTkXjvYTBjTDf3zHMFWgzYBfefC8kh8NPK5nNs8ldorbd
AemEnjBkP+DSELKyK6vLulOrdtzAQgRCp+MsT+xTbO2ArefeX826SXSpoQKBgC2v
nDOHBQXje1dTawlUToFUrgQE8AwlOYEdKKyUoCLOvqEW8DO2a0MtyM+MB6tQI7Wm
iH1T73L0LHGlK3bw3aRAwV5/fu/O+jAdFk8AHjPTFE+acu2fi4c6aKb0GjAxYksU
yjIFeK/pKinv4SESMkjpW0WowGiDgtcRPBAA/LaFAoGAfEM1rfM0v3UmB7PS6u0m
P3ckP2CFCdaryXPfC52GBcJ3Q46YpsQvLTVotM+teHvTjNw2jwwZxIl4NenGSEj3
KDhQoOiQC9BrDD+DB4I9+T9nxT3g7R6MrgITghB4We7TVhL/PljnJTyDqpjNA4kY
TveAJPv6Xq1ERt5PUtX3BqQ=
-----END PRIVATE KEY-----
```

Kemudian, hint soal mengatakan untuk decrypt TLS stream, lalu saya coba mendecrypt-nya menggunakan key tadi, dengan cara membukanya lewat wireshark dan pergi ke tab Edit > Preference > Protocols > TLS > Edit RSA key list, dan masukkan lokasi file key berada :



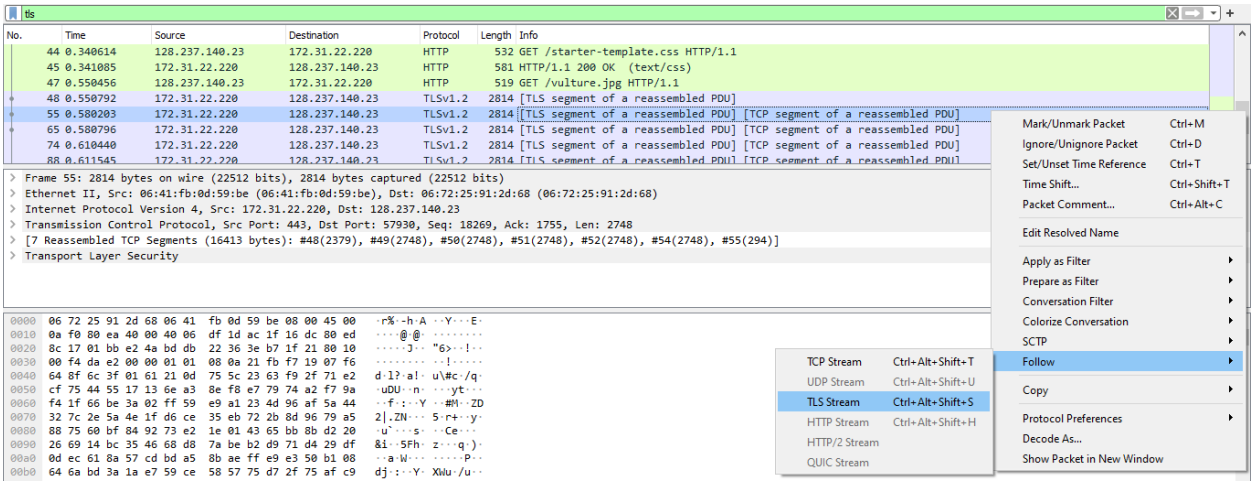
Setelah itu, saya lihat lagi ke packet protocol TLS dan ternyata di bagian header HTTP-nya terdapat flag yang dicari :

```
GET / HTTP/1.1
Host: ec2-18-223-184-200.us-east-2.compute.amazonaws.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Fri, 23 Aug 2019 15:56:36 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Mon, 12 Aug 2019 16:50:05 GMT
ETag: "5ff-58fee50dc3fb0-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Pico-Flag: picoCTF{nongshim.shrimp.crackers}
Content-Length: 821
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

Flag = picoCTF{nongshim.shrimp.crackers}

WEBNET1

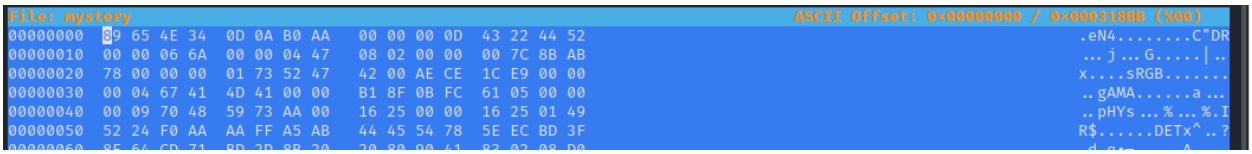
Diberikan sebuah file capture packet, lalu di sini saya agak kebingungan cukup lama untuk memahami hint yang diberikan soal, karena hint yang diberikan sama seperti soal webnet0 sebelumnya. Sampai akhirnya saya mencoba untuk memfilter packet berdasarkan protocol TLS, karena jumlahnya sedikit saya cek satu persatu dan saya menemukan flag pada packet no 55 :



Flag = picoCTF{honey.roasted.peanuts}

CORRUPT

Diberikan sebuah file yang tidak dapat dibaca (corrupt), kemudian sesuai hint soal diberitahu bahwa “try fixing the file header” lalu langsung saja saya buka menggunakan hexeditor di terminal :

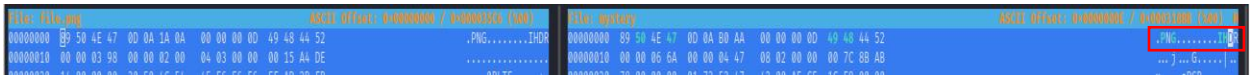


Terlihat bahwa header file tersebut diawali dengan hexa 89 dan saya berasumsi file tersebut adalah file PNG (reference : https://www.garykessler.net/library/file_sigs.html).

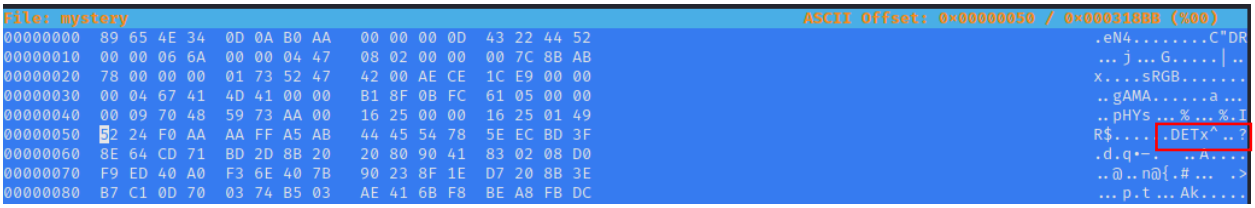
```
89 50 4E 47 0D 0A 1A 0A            %PNG....
```

PNG [Portable Network Graphics file](#)
Trailer: 49 45 4E 44 AE 42 60 82 (IEND®B`,...)

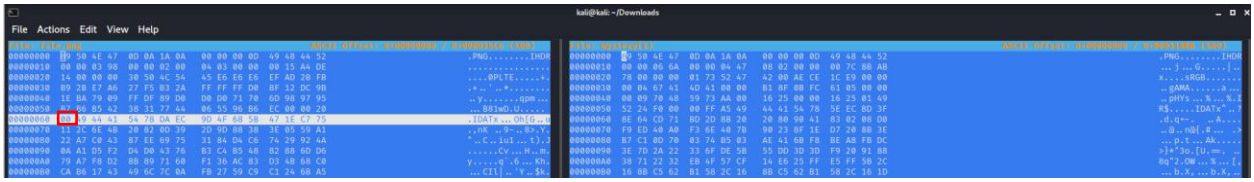
Kemudian, untuk memperbaikinya saya mengunduh file png dari internet dan saya bandingkan header nya dengan file corrupt tadi. Pertama saya perbaiki header-nya menjadi PNG dan IHDR :



Setelah itu, saya coba buka filenya dan ternyata masih corrupt. Kemudian saya coba cek lagi ternyata masih terdapat kesalahan yaitu di bagian berikut :



Lalu saya perbaiki menjadi IDAT (Image Data) sesuai seperti file png yang saya unduh dari internet. Kemudian saya save lagi dan ternyata masih ada kesalahan. Lalu saya menemukan bahwa di depan kata IDAT tersebut hexa yang benar harus bernilai 0 seperti pada gambar :



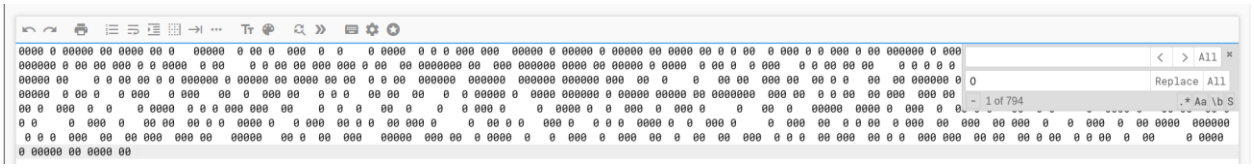
Selanjutnya, saya ubah hexa yang di file corrupt tersebut di depan kata IDAT tadi menjadi 00 dan file berhasil dibuka :

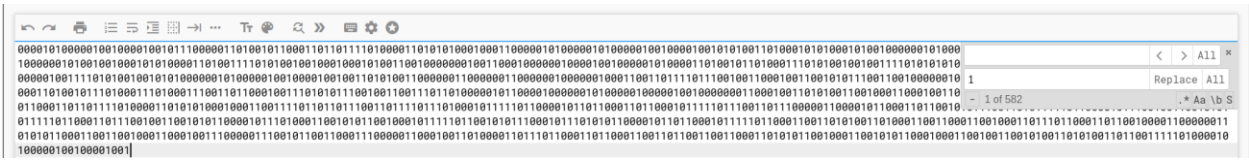
Flag = picoCTF{c0rrupt10n_1847995}

WHITEPAGES

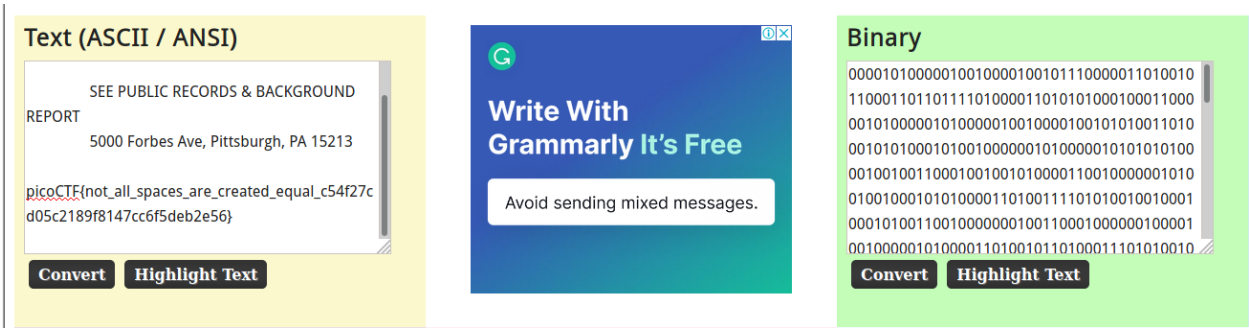
Diberikan sebuah file berupa halaman kosong lalu saya coba lihat menggunakan text editor online dan ternyata terdapat dua warna spasi :

Lalu, saya coba replace spasi warna merah menjadi angka biner 0 dan spasi warna putih menjadi angka biner 1 :





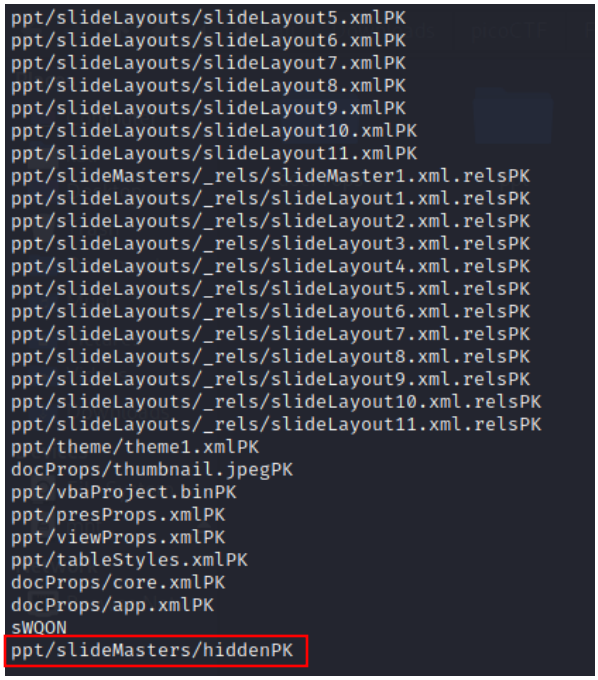
Selanjutnya, saya konversi bilangan biner tersebut ke ascii dan didapat flag :



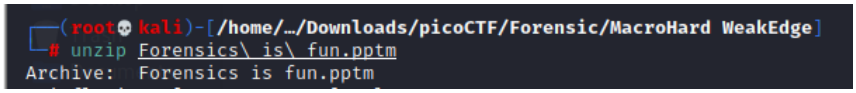
Flag = picoCTF{not_all_spaces_are_created_equal_c54f27cd05c2189f8147cc6f5deb2e56}

MACROHARD WEAKEDGE

Diberikan sebuah file PowerPoint, kemudian saya melakukan strings dan ternyata di akhir strings terdapat sebuah file hidden yang saya asumsikan berisi flag :



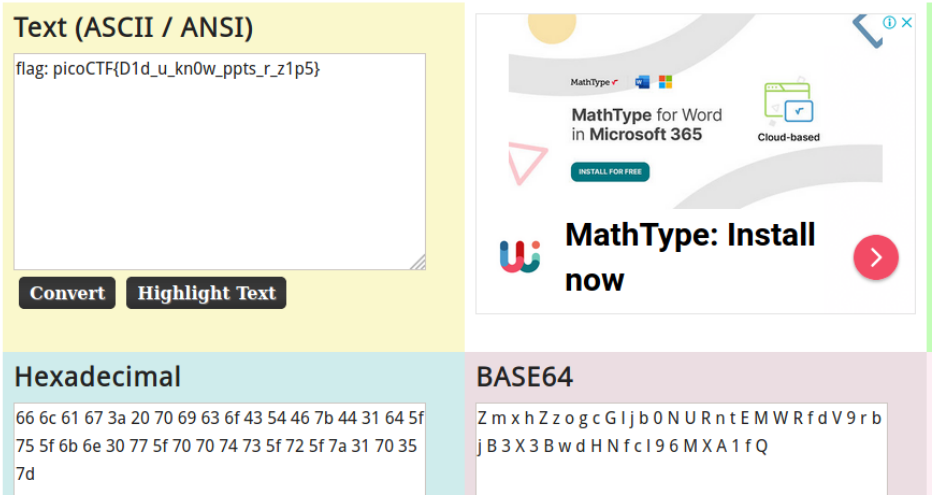
Kemudian saya mencoba menggunakan tools zsteg dan binwalk namun tidak berhasil. Selanjutnya saya mencoba meng-unzip file tersebut dan ternyata berhasil dilakukan :



Lalu sesuai hint yang saya dapat di awal, saya masuk ke direktori ppt/slideMasters dan benar saja di sana terdapat file text yang berisi base64 :

```
┌──(root💀kali)-[/home/…/Downloads/picoCTF/Forensic/MacroHard WeakEdge]
└─# cd ppt/slideMasters

┌──(root💀kali)-[/home/…/Forensic/MacroHard WeakEdge/ppt/slideMasters]
└─# ls
hidden  _rels  slideMaster1.xml

┌──(root💀kali)-[/home/…/Forensic/MacroHard WeakEdge/ppt/slideMasters]
└─# cat hidden
Z m x h Z z o g c G l j b 0 N U R n t E M W R f d V 9 r b j B 3 X 3 B w d H N f c
l 9 6 M X A 1 f Q
```

Base64 tersebut saya konversi ke ascii dan didapat flag nya :



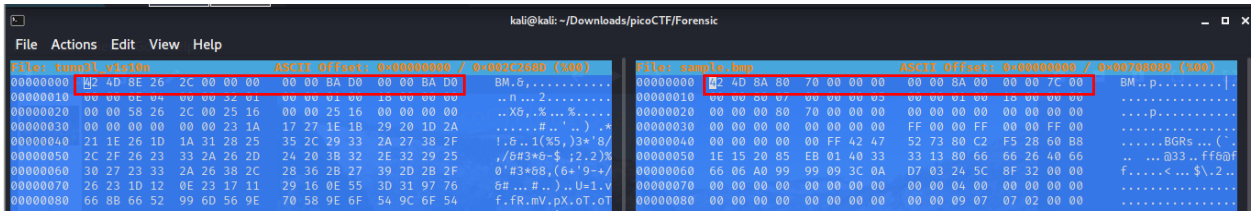
Flag = picoCTF{D1d_u_kn0w_ppts_r_z1p5}

TUNN3L V1S10N

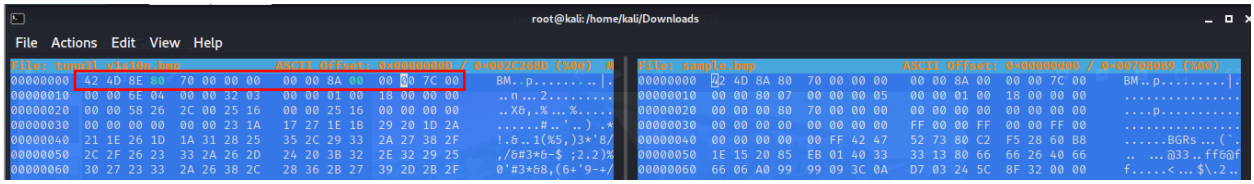
Diberikan sebuah file tanpa ekstensi, lalu saya langsung identifikasi tetapi ternyata file tersebut merupakan file corrupt yang tidak dapat dibaca :

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# file tunn3l_v1s10n
tunn3l_v1s10n: data
```

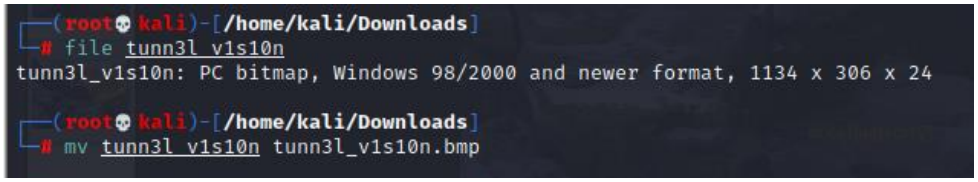
Setelah itu di sini saya berasumsi bahwa kita harus memperbaiki header file nya, dan setelah saya buka menggunakan hexeditor header nya berawalan BM yang artinya file tersebut adalah file BMP. Kemudian saya bandingkan header file tersebut dengan file BMP yang sebenarnya dan benar saja terdapat nilai hexa yang salah :



Kemudian saya perbaiki menjadi seperti berikut :



Kemudian untuk memastikan saya cek lagi dan berhasil diubah ke BMP :

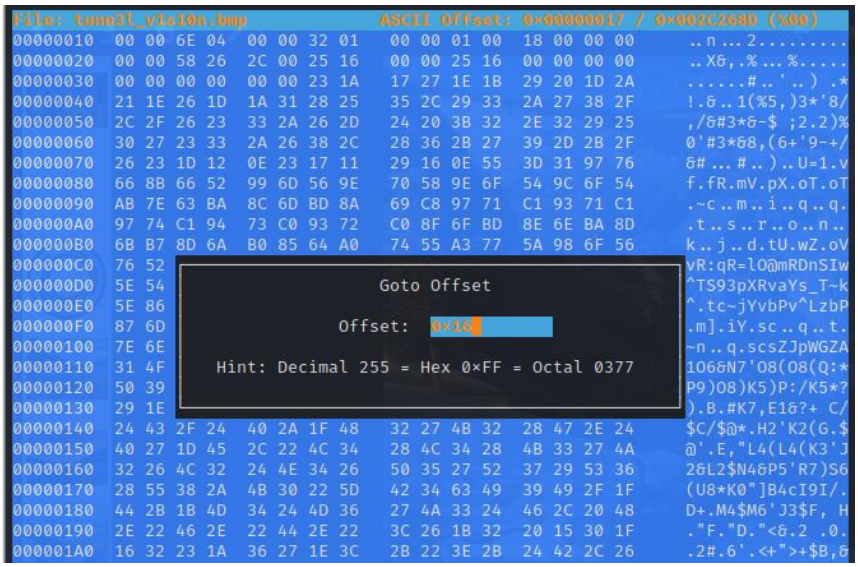


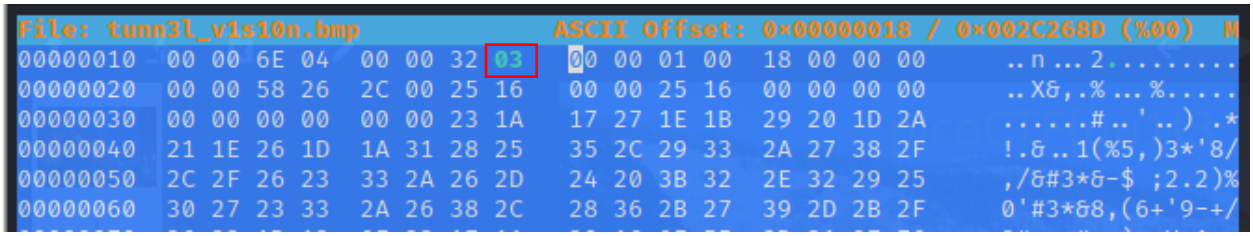
Setelah itu saya coba buka kembali dan ternyata muncul flag palsu :

Di sini saya berusaha memahami lagi hint soal, dan saya kemudian menemukan bahwa file BMP bisa diperbesar ukuran height gambarnya : (reference : https://en.wikipedia.org/wiki/BMP_file_format)

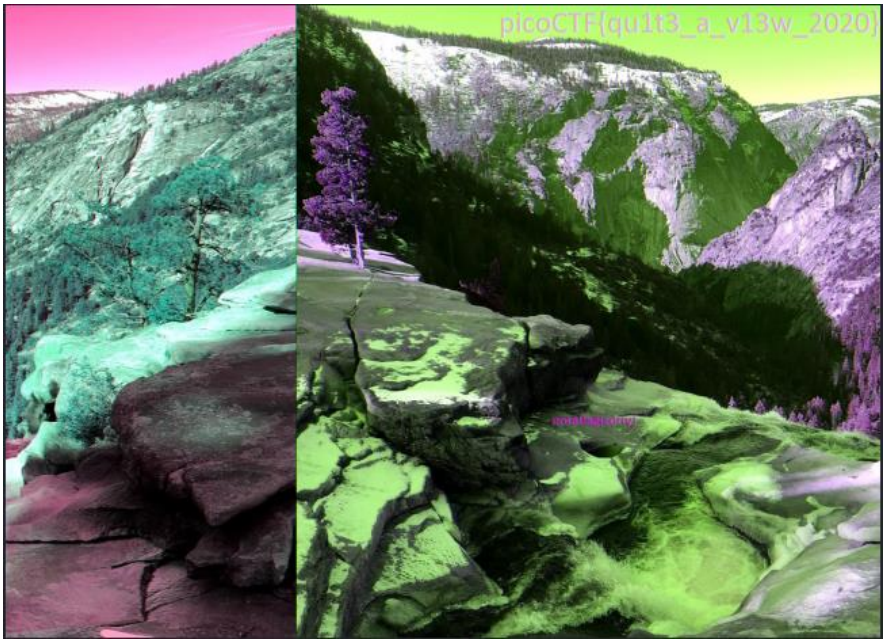
Offset (hex)	Offset (dec)	Size (bytes)	Windows BITMAPINFOHEADER[2]
0E	14	4	the size of this header, in bytes (40)
12	18	4	the bitmap width in pixels (signed integer)
16	22	4	the bitmap height in pixels (signed integer)
1A	26	2	the number of color planes (must be 1)
1C	28	2	the number of bits per pixel, which is the color depth of the image. Typical values are 1, 4, 8, 16, 24 and 32.
1E	30	4	the compression method being used. See the next table for a list of possible values
22	34	4	the image size. This is the size of the raw bitmap data; a dummy 0 can be given for BI_RGB bitmaps.
26	38	4	the horizontal resolution of the image. (pixel per metre, signed integer)
2A	42	4	the vertical resolution of the image. (pixel per metre, signed integer)
2E	46	4	the number of colors in the color palette, or 0 to default to 2^n
32	50	4	the number of important colors used, or 0 when every color is important; generally ignored

Lalu saya pergi ke offset (Ctrl+T) dan mengetikkan hexa 16 untuk mencari letak hexa untuk height file dan saya menemukan itu berada di hex 32, namun setelah saya coba rubah itu bukan hex untuk height file. Kemudian saya coba rubah hex 01 yang ada di sebelah hex 32 tersebut dan ternyata berhasil :





Saya merubah nilai nya menjadi hex 03 dan didapat flag yang tertera di atas gambar :



Flag = picoCTF{qu1t3_a_v13w_2020}