PicoCTF Report

PicoCTF Report

29 October 2024 13:47

General Skill section

---

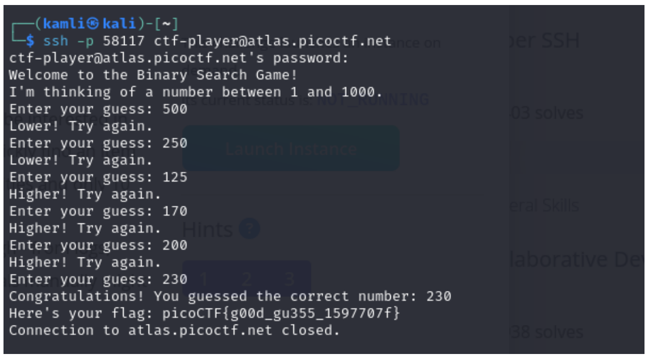
1 Binary Search(General skill)

Description - Want to play a game? As you use more of the shell, you might be interested
Binary search is a classic algorithm used to quickly find an item in a sorted list. Can you find
1000 possibilities and only 10 guesses.

Process -

- **Log in**: Use the provided credentials to access the instance.
- **Number guessing game**: The program asks you to guess a number until you get it ri
- **Success**: After several attempts, you guess the correct number.
- **Submit the flag**: Once you guess correctly, you submit the flag as instructed.

Solution



```
┌──(kamli㉿kali)-[~]
└─$ ssh -p 58117 ctf-player@atlas.picoctf.net
ctf-player@atlas.picoctf.net's password:
Welcome to the Binary Search Game!
I'm thinking of a number between 1 and 1000.
Enter your guess: 500
Lower! Try again.
Enter your guess: 250
Lower! Try again.
Enter your guess: 125
Higher! Try again.
Enter your guess: 170
Higher! Try again.
Enter your guess: 200
Higher! Try again.
Enter your guess: 230
Congratulations! You guessed the correct number: 230
Here's your flag: picoCTF{g00d_gu355_1597707f}
Connection to atlas.picoctf.net closed.
```

2 Time Machine (general skill)

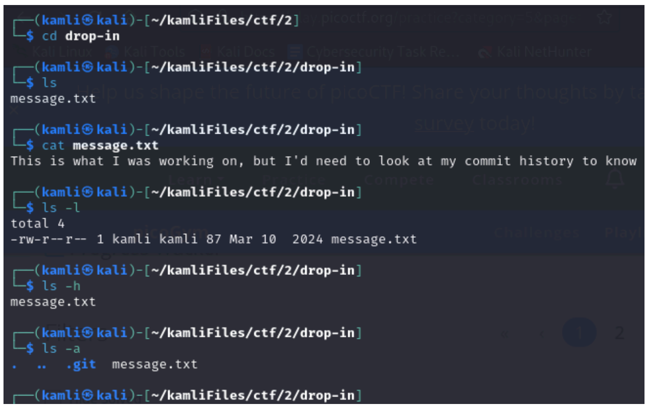
Description

What was I last working on? I remember writing a note to help me remember... You can
challenge files here:

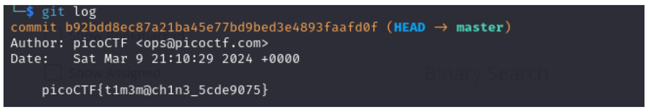
Process

- **First, I download the ZIP file.**
- **Then, I unzip it using the appropriate command.**
- **There are multiple files, so I navigate into the directory.**
- **Inside, I see a message.txt file, which I open using a text editor.**
- **I then use ls -a to check for any hidden files.**
- **I find a .git directory.**

Solution



```
┌──(kamli㉿kali)-[~/kamliFiles/ctf/2]
└─$ cd drop-in

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ls
message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ cat message.txt
This is what I was working on, but I'd need to look at my commit history to know

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ls -l
total 4
-rw-r--r-- 1 kamli kamli 87 Mar 10  2024 message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ls -h
message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ls -a
.  ..  .git  message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
```



```
└─$ git log
commit b92bdd8ec87a21ba45e77bd9bed3e4893faafd0f (HEAD → master)
Author: picoCTF <ops@picoctf.com>
Date:   Sat Mar 9 21:10:29 2024 +0000

    picoCTF{t1m3m@ch1n3_5cde9075}
```

3 Super SSH (general skill)

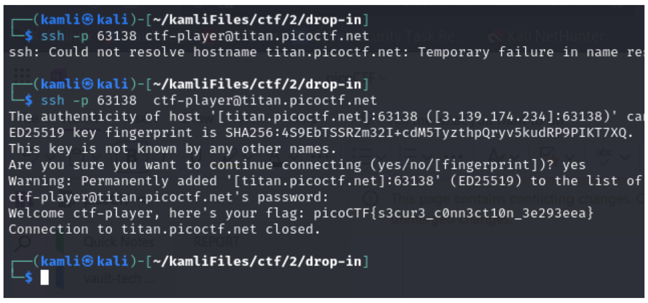
Description

Using a Secure Shell (SSH) is going to be pretty important.
Additional details will be available after launching your challenge instance

Process

- **First, I establish a connection using the information provided to me.**
- **Then, I type "yes" to confirm any prompts (such as adding the host to known h**
- **Next, I enter the provided password.**
- **Finally, I obtain the flag.**

Solution

- 

```
┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ssh -p 63138 ctf-player@titan.picoctf.net
ssh: Could not resolve hostname titan.picoctf.net: Temporary failure in name re

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ ssh -p 63138  ctf-player@titan.picoctf.net
The authenticity of host '[titan.picoctf.net]:63138 ([3.139.174.234]:63138)' ca
ED25519 key fingerprint is SHA256:4S9EbTSSRZm32I+cdM5TyzthpQryv5kudRP9PIKT7XQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[titan.picoctf.net]:63138' (ED25519) to the list of
ctf-player@titan.picoctf.net's password:
Welcome ctf-player, here's your flag: picoCTF{s3cur3_c0nn3ct10n_3e293eea}
Connection to titan.picoctf.net closed.

┌──(kamli㉿kali)-[~/kamliFiles/ctf/2/drop-in]
└─$ █
```

4 Commitment Issues(General skill)

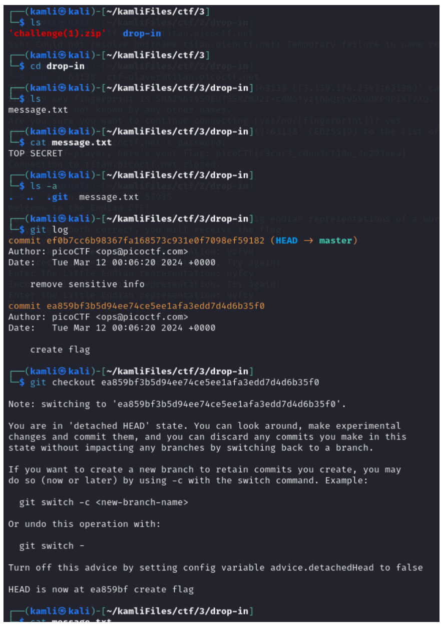
Description

I accidentally wrote the flag down. Good thing I deleted it! You download the challenge files

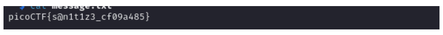
Process

- **First, I download the challenge file.**
- **Then, I unzip the challenge file.**
- **Next, I navigate to the extracted directory.**
- **I see a message.txt file, and I use cat to view its contents.**
- **I run ls -a to check for any hidden files.**
- **I find a .git directory.**
- **I run git log to find a commit.**
- **Using the commit hash, I run git checkout <commit-hash>.**
- **Finally, I find and retrieve the flag.**

Solution

- 

```
┌──(kamli㉿kali)-[~/kamliFiles/ctf/3]
└─$ ls
'challenge(1).zip'   drop-in

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3]
└─$ cd drop-in

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ ls
message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ cat message.txt
TOP SECRET

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ ls -a
.  ..  .git  message.txt

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ git log
commit ef0b7cc6b98367fa168573c931e0f7098ef59182 (HEAD → master)
Author: picoCTF <ops@picoctf.com>
Date:   Tue Mar 12 00:06:20 2024 +0000

    remove sensitive info

commit ea859bf3b5d94ee74ce5ee1afa3edd7d4d6b35f0
Author: picoCTF <ops@picoctf.com>
Date:   Tue Mar 12 00:06:20 2024 +0000

    create flag

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ git checkout ea859bf3b5d94ee74ce5ee1afa3edd7d4d6b35f0

Note: switching to 'ea859bf3b5d94ee74ce5ee1afa3edd7d4d6b35f0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at ea859bf create flag

┌──(kamli㉿kali)-[~/kamliFiles/ctf/3/drop-in]
└─$ cat message.txt
```



```
$ cat message.txt
picoCTF{s@n1t1z3_cf09a485}
```

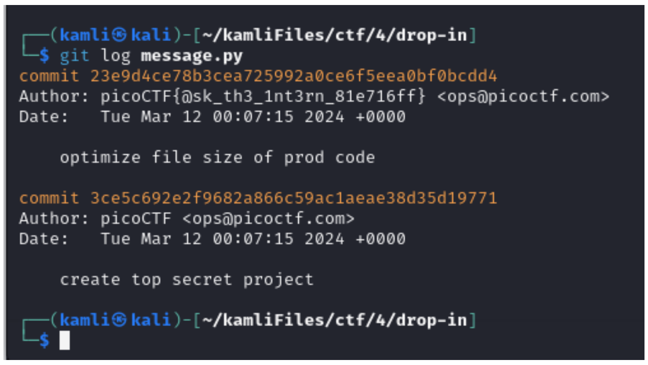
5 Blame Game(General skill)

Description

Someone's commits seems to be preventing the program from working. Who is it?

Process

- **First, I unzip the challenge file and navigate to the extracted directory.**
- **Inside, I see a message.txt file, which I open.**
- **I don't understand the contents of the message.txt file.**
- **I run the command git log message.py to check the version history of messa**
- **Finally, I find the flag within the commit history.**



```
┌──(kamli㉿kali)-[~/kamliFiles/ctf/4/drop-in]
└─$ git log message.py
commit 23e9d4ce78b3cea725992a0ce6f5eea0bf0bcdd4
Author: picoCTF{@sk_th3_1nt3rn_81e716ff} <ops@picoctf.com>
Date:   Tue Mar 12 00:07:15 2024 +0000

    optimize file size of prod code

commit 3ce5c692e2f9682a866c59ac1aeae38d35d19771
Author: picoCTF <ops@picoctf.com>
Date:   Tue Mar 12 00:07:15 2024 +0000

    create top secret project

┌──(kamli㉿kali)-[~/kamliFiles/ctf/4/drop-in]
└─$ 
```

6 repetitions(General skill)

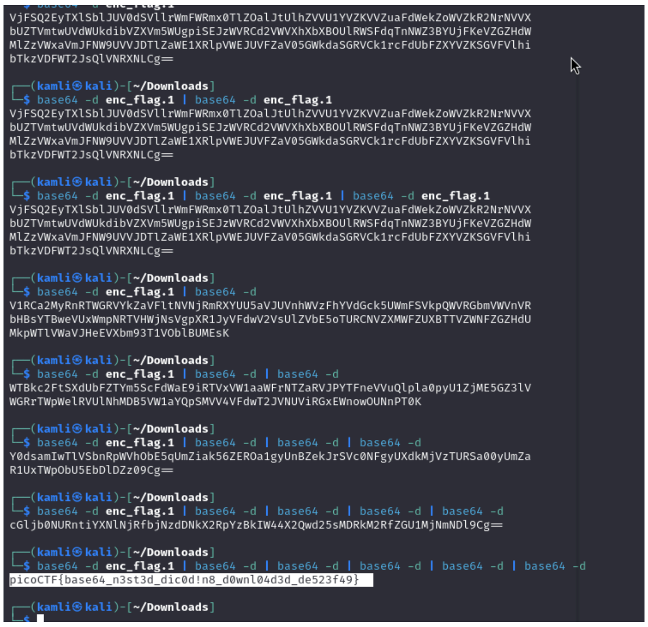
Description

Can you make sense of this file?

Process

- **I download the file.**
- **Then, I review its contents.**
- **I notice that the file is base64 encoded, so I decide to decode it.**
- **I perform multiple rounds of decoding until I reach the final decoded content.**

Solution



```
└─$ base64 -d enc_flag.1
VjFSQ2EyTXlSblJUV0dSVllrWmFWRmx0TlZOalJtUlhZVVU1YVZKVVZuaFdWekZoWVZkR2NrNVVX
bUZTVmtwUVdWUkdibVZXVm5WUgpiSEJzWVRCd2VWVXhXbXBOUlRWSFdqTnNWZ3BYUjFKeVZGZHdW
MlZzVWxaVmJFNW9UVVJDTlZaWE1XRlpVWEJUVFZaV05GWkdaSGRVCk1rcFdUbFZXYVZKSGVFVlhi
bTkzVDFWT2JsQlVNRXNLCg==

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d enc_flag.1
VjFSQ2EyTXlSblJUV0dSVllrWmFWRmx0TlZOalJtUlhZVVU1YVZKVVZuaFdWekZoWVZkR2NrNVVX
bUZTVmtwUVdWUkdibVZXVm5WUgpiSEJzWVRCd2VWVXhXbXBOUlRWSFdqTnNWZ3BYUjFKeVZGZHdW
MlZzVWxaVmJFNW9UVVJDTlZaWE1XRlpVWEJUVFZaV05GWkdaSGRVCk1rcFdUbFZXYVZKSGVFVlhi
bTkzVDFWT2JsQlVNRXNLCg==

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d enc_flag.1 | base64 -d enc_flag.1
VjFSQ2EyTXlSblJUV0dSVllrWmFWRmx0TlZOalJtUlhZVVU1YVZKVVZuaFdWekZoWVZkR2NrNVVX
bUZTVmtwUVdWUkdibVZXVm5WUgpiSEJzWVRCd2VWVXhXbXBOUlRWSFdqTnNWZ3BYUjFKeVZGZHdW
MlZzVWxaVmJFNW9UVVJDTlZaWE1XRlpVWEJUVFZaV05GWkdaSGRVCk1rcFdUbFZXYVZKSGVFVlhi
bTkzVDFWT2JsQlVNRXNLCg==

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d
V1RCa2MyRnRTWGRVYkZaVFltNVNjRmRXYUU5aVJUVnhWVzFhYVdGck5UWmFSVkpQWVRGbmVWVnVR
bHBsYTBweVUxWmpNRTVHWjNsVgpXR1JyVFdwV2VsRVVUbE5oTURCNVZXMWFZUXBTTVZWNFZGZHdU
MkpWTlVWaVJHeEVXbm93T1VOblBUMEsK

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d | base64 -d
WTBkc2FtSXdUbFZTYm5ScFdWaE9iRTVxVW1aaWFrNTZaRVJPYTFneVVuQlpla0pyU1ZjME5GZ3lV
WGRrTWpWelRVUlNhMDB5VW1aYQpSMVV4VFdwT2JsQlVNRGxEWnowOUNnPT0K

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d | base64 -d | base64 -d
Y0dsamIwTlVSbnRpWVhOobE5qUmZiak56ZEROa1gyUnBZekJrSVc0NFgyUXdkMjVzTURSa00yUmZa
R1UxTWpObU5EbDlDZz09Cg==

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d | base64 -d | base64 -d | base64 -d
cGljb0NURntiYXNlNjRfbjNzdDNkX2RpYzBkIW44X2Qwd25sMDRkM2RfZGU1MjNmNDl9Cg==

┌──(kamli㉿kali)-[~/Downloads]
└─$ base64 -d enc_flag.1 | base64 -d | base64 -d | base64 -d | base64 -d | base64 -d
picoCTF{base64_n3st3d_dic0d!n8_d0wnl04d3d_de523f49}

┌──(kamli㉿kali)-[~/Downloads]
└─$ 
```

7 Big Zip(general skill)

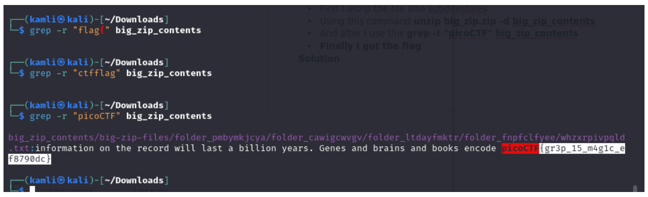
Description

Unzip this archive and find the flag.

Process

- **First, I unzip the file and its subdirectories using the following command:** unzip big big_zip_contents
- Then, I search for the flag within the unzipped content using: grep -r "picoCTF" big_zip_c
- Finally, I find the flag.

Solution



```
┌──(kamli㉿kali)-[~/Downloads]
└─$ grep -r "flag{" big_zip_contents

┌──(kamli㉿kali)-[~/Downloads]
└─$ grep -r "ctfflag" big_zip_contents

┌──(kamli㉿kali)-[~/Downloads]
└─$ grep -r "picoCTF" big_zip_contents
big_zip_contents/big-zip-files/folder_pmbymkjcya/folder_cawigcwvgv/folder_ltdayfmktr/folder_fnpfclfyee/whzxrpivpqld
.txt:information on the record will last a billion years. Genes and brains and books encode picoCTF{gr3p_15_m4g1c_e
f8790dc}

┌──(kamli㉿kali)-[~/Downloads]
└─$
```

8 First Find (general skill)

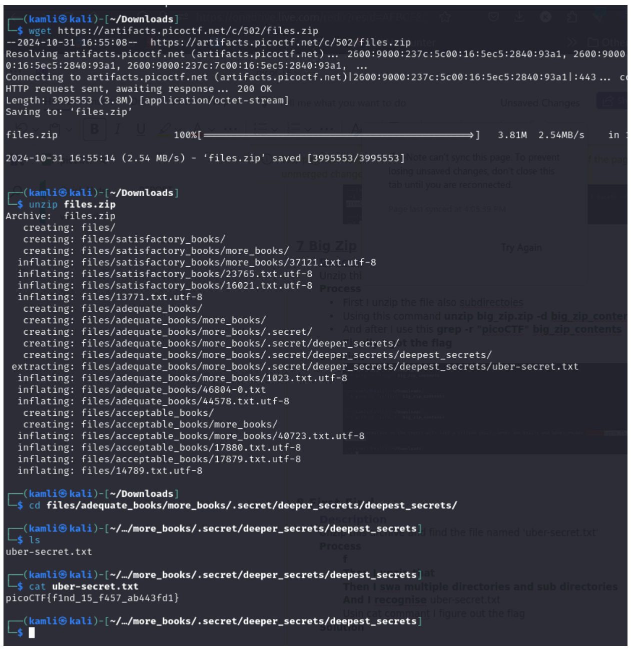
Description

Unzip this archive and find the file named 'uber-secret.txt'

Process

- **First, I download the ZIP file.**
- **Then, I unzip it.**
- **I see multiple directories and subdirectories.**
- **I recognize a file named uber-secret.txt.**
- **Using the cat command, I read the file and find the flag:**
- **cat uber-secret.txt**

Solution



```
┌──(kamli㉿kali)-[~/Downloads]
└─$ wget https://artifacts.picoctf.net/c/502/files.zip
--2024-10-31 16:55:08--  https://artifacts.picoctf.net/c/502/files.zip
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 2600:9000:237c:5c00:16:5ec5:2840:93a1, 2600:9000
0:16:5ec5:2840:93a1, 2600:9000:237c:7c00:16:5ec5:2840:93a1, ...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|2600:9000:237c:5c00:16:5ec5:2840:93a1|:443... co
HTTP request sent, awaiting response... 200 OK
Length: 3995553 (3.8M) [application/octet-stream]
Saving to: 'files.zip'

files.zip           100%[===============================================>]   3.81M  2.54MB/s    in 

2024-10-31 16:55:14 (2.54 MB/s) - 'files.zip' saved [3995553/3995553]

┌──(kamli㉿kali)-[~/Downloads]
└─$ unzip files.zip
Archive:  files.zip
   creating: files/
   creating: files/satisfactory_books/
   creating: files/satisfactory_books/more_books/
  inflating: files/satisfactory_books/more_books/37121.txt.utf-8
  inflating: files/satisfactory_books/23765.txt.utf-8
  inflating: files/satisfactory_books/16021.txt.utf-8
  inflating: files/13771.txt.utf-8
   creating: files/adequate_books/
   creating: files/adequate_books/more_books/
   creating: files/adequate_books/more_books/.secret/
   creating: files/adequate_books/more_books/.secret/deeper_secrets/
   creating: files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/
 extracting: files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/uber-secret.txt
  inflating: files/adequate_books/more_books/1023.txt.utf-8
  inflating: files/adequate_books/46804-0.txt
  inflating: files/adequate_books/44578.txt.utf-8
   creating: files/acceptable_books/
   creating: files/acceptable_books/more_books/
  inflating: files/acceptable_books/more_books/40723.txt.utf-8
  inflating: files/acceptable_books/17880.txt.utf-8
  inflating: files/acceptable_books/17879.txt.utf-8
  inflating: files/14789.txt.utf-8

┌──(kamli㉿kali)-[~/Downloads]
└─$ cd files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/

┌──(kamli㉿kali)-[~/…/more_books/.secret/deeper_secrets/deepest_secrets]
└─$ ls
uber-secret.txt

┌──(kamli㉿kali)-[~/…/more_books/.secret/deeper_secrets/deepest_secrets]
└─$ cat uber-secret.txt
picoCTF{f1nd_15_f457_ab443fd1}

┌──(kamli㉿kali)-[~/…/more_books/.secret/deeper_secrets/deepest_secrets]
└─$
```

9 Runme.py(general skill)

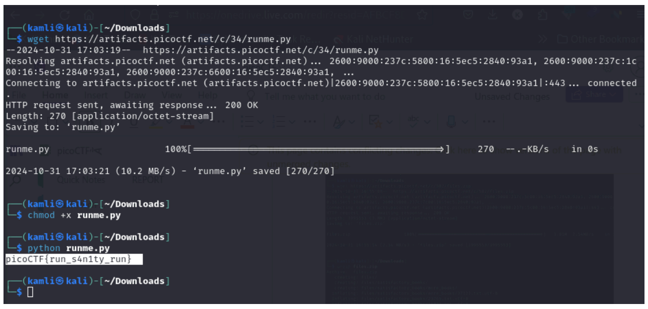
Description

Run the runme.py script to get the flag. Download the script with your browser or with wget i

Process

- **First I download the file using wget <link>**
- **Then I saw there is a file runme.py**
- **I run that file**
- **Finally I get the flag**

Solution



```
┌──(kamli㉿kali)-[~/Downloads]
└─$ wget https://artifacts.picoctf.net/c/34/runme.py
--2024-10-31 17:03:19--  https://artifacts.picoctf.net/c/34/runme.py
Resolving artifacts.picoctf.net (artifacts.picoctf.net)... 2600:9000:237c:5800:16:5ec5:2840:93a1, 2600:9000:237c:1c
00:16:5ec5:2840:93a1, 2600:9000:237c:6600:16:5ec5:2840:93a1, ...
Connecting to artifacts.picoctf.net (artifacts.picoctf.net)|2600:9000:237c:5800:16:5ec5:2840:93a1|:443... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 270 [application/octet-stream]
Saving to: ‘runme.py’

runme.py              100%[===================================>]     270  --.-KB/s    in 0s

2024-10-31 17:03:21 (10.2 MB/s) - ‘runme.py’ saved [270/270]

┌──(kamli㉿kali)-[~/Downloads]
└─$ chmod +x runme.py

┌──(kamli㉿kali)-[~/Downloads]
└─$ python runme.py
picoCTF{run_s4n1ty_run}

┌──(kamli㉿kali)-[~/Downloads]
└─$ 
```

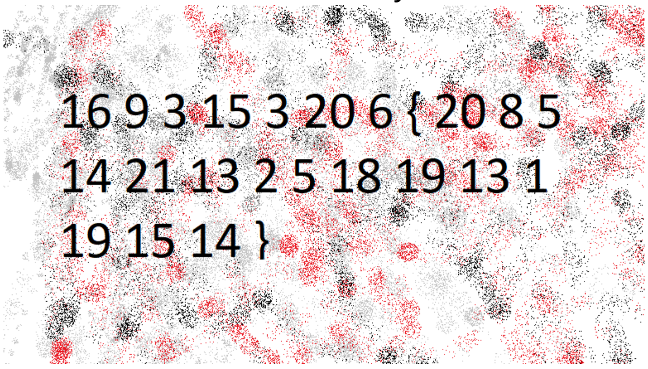
Cryptography section

----

11 The Numbers

Description

The numbers... what do they mean?



16 9 3 15 3 20 6 { 20 8 5
14 21 13 2 5 18 19 13 1
19 15 14 }

Process -

- **First, I notice that each number represents a letter of the alphabet.**
- **I convert each number to its corresponding letter (e.g., 1 = A, 2 = B, etc.).**
- **The hint specifies that the result should be in capital letters, so I ensure each le**
- **Finally, I obtain the flag: PICOCTF{THENUMBERSMASON}.**

12 13

Description

Cryptography can be easy, do you know what ROT13 is? cvpbPGS{abg_gbb_onq_bs_n

Processs

- **ROT13 is a cipher where each letter is replaced by the letter 13 places after i**
- **I applied the ROT13 transformation to each letter of the given text.**
- **I obtained a flag, but it was missing a letter, so I added the missing characte**
- **Finally, I got the complete flag: picoCTF{not_too_bad_of_a_problem}.**

13 interencdec

Description

Can you get the real meaning from this file. Download the file here.

Process

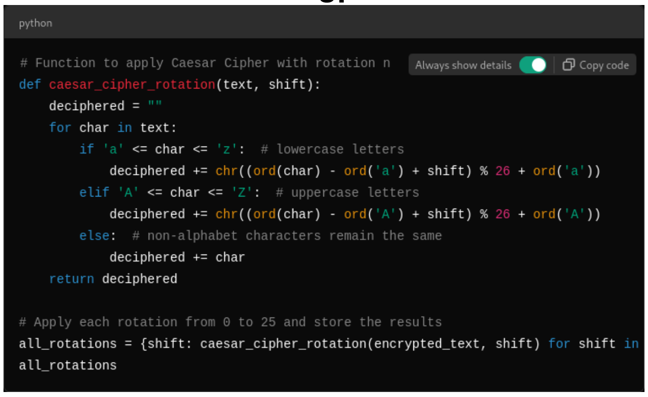
- **First, I download the file.**
- **Then, I decode the file using the base64 -d command: base64 -d <file>**
- **I copy the decoded content, which is inside the quotes.**
- **I paste the copied content into the website https://www.base64decode.or**
- **I decode the output again.**
- **Finally, I get the flag: picoCTF{caesar_d3cr9pt3d_a47c6d69}.**

14 rotation

Description

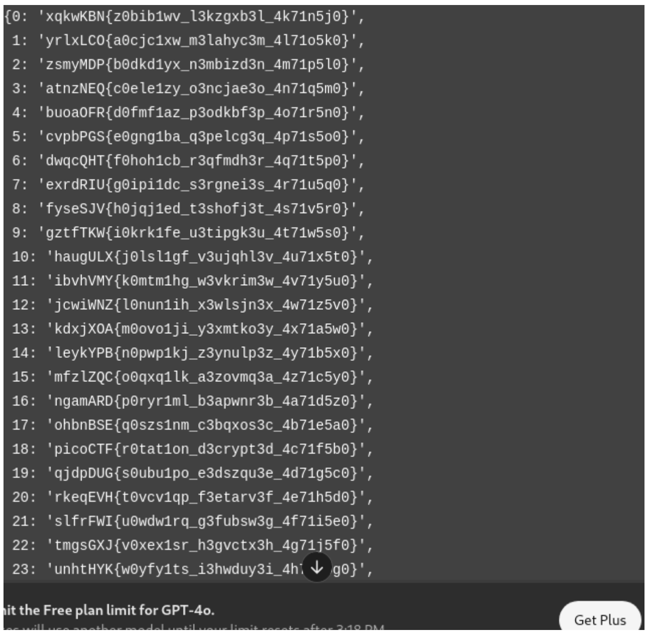
You will find the flag after decrypting this file Download the encrypted flag here.

- **Process**
  - **for this I use chatgpt**

- 

```python
# Function to apply Caesar Cipher with rotation n
def caesar_cipher_rotation(text, shift):
    deciphered = ""
    for char in text:
        if 'a' <= char <= 'z':  # lowercase letters
            deciphered += chr((ord(char) - ord('a') + shift) % 26 + ord('a'))
        elif 'A' <= char <= 'Z':  # uppercase letters
            deciphered += chr((ord(char) - ord('A') + shift) % 26 + ord('A'))
        else:  # non-alphabet characters remain the same
            deciphered += char
    return deciphered

# Apply each rotation from 0 to 25 and store the results
all_rotations = {shift: caesar_cipher_rotation(encrypted_text, shift) for shift in
all_rotations
```

- **First, I try to use ROT13 on the flag, but it doesn't give the correct answer.**
- **Then, I try cipher rotation, applying multiple rotations.**
- **After several rotations, I find that applying a rotation of 18 gives the correct res**
- **Finally, I get the flag.**

- 

```
{0: 'xqkwKBN{z0bib1wv_l3kzgxb3l_4k71n5j0}',
 1: 'yrlxLCO{a0cjc1xw_m3lahyc3m_4l71o5k0}',
 2: 'zsmyMDP{b0dkd1yx_n3mbizd3n_4m71p5l0}',
 3: 'atnzNEQ{c0ele1zy_o3ncjae3o_4n71q5m0}',
 4: 'buoaOFR{d0fmf1az_p3odkbf3p_4o71r5n0}',
 5: 'cvpbPGS{e0gng1ba_q3pelcg3q_4p71s5o0}',
 6: 'dwqcQHT{f0hoh1cb_r3qfmdh3r_4q71t5p0}',
 7: 'exrdRIU{g0ipi1dc_s3rgnei3s_4r71u5q0}',
 8: 'fyseSJV{h0jqj1ed_t3shofj3t_4s71v5r0}',
 9: 'gztfTKW{i0krk1fe_u3tipgk3u_4t71w5s0}',
 10: 'haugULX{j0lsl1gf_v3ujqhl3v_4u71x5t0}',
 11: 'ibvhVMY{k0mtm1hg_w3vkrim3w_4v71y5u0}',
 12: 'jcwiWNZ{l0nun1ih_x3wlsjn3x_4w71z5v0}',
 13: 'kdxjXOA{m0ovo1ji_y3xmtko3y_4x71a5w0}',
 14: 'leykYPB{n0pwp1kj_z3ynulp3z_4y71b5x0}',
 15: 'mfzlZQC{o0qxq1lk_a3zovmq3a_4z71c5y0}',
 16: 'ngamARD{p0ryr1ml_b3apwnr3b_4a71d5z0}',
 17: 'ohbnBSE{q0szs1nm_c3bqxos3c_4b71e5a0}',
 18: 'picoCTF{r0tat1on_d3crypt3d_4c71f5b0}',
 19: 'qjdpDUG{s0ubu1po_e3dszqu3e_4d71g5c0}',
 20: 'rkeqEVH{t0vcv1qp_f3etarv3f_4e71h5d0}',
 21: 'slfrFWI{u0wdw1rq_g3fubsw3g_4f71i5e0}',
 22: 'tmgsGXJ{v0xex1sr_h3gvctx3h_4g71j5f0}',
 23: 'unhtHYK{w0yfy1ts_i3hwduy3i_4h7 g0}',
```

- **picoCTF{r0tat1on_d3crypt3d_4c71f5b0}**

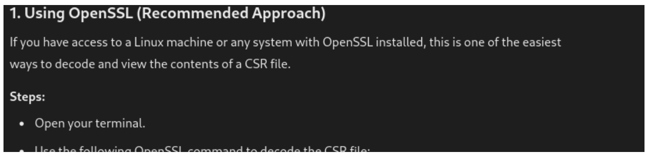
15 ReadMyCert

Description

How about we take you on an adventure on exploring certificate signing requests Tak
file here.

Process

- **For this, I asked ChatGPT how to solve it.**
- **ChatGPT suggested using the OpenSSL approach.**
-

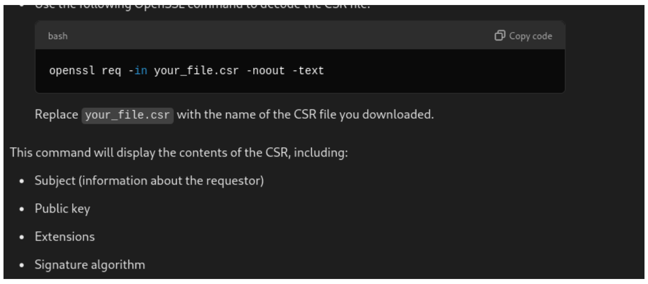


1. Using OpenSSL (Recommended Approach)

If you have access to a Linux machine or any system with OpenSSL installed, this is one of the easiest ways to decode and view the contents of a CSR file.

Steps:

- Open your terminal.

- 

Use the following OpenSSL command to decode the CSR file:

bash Copy code

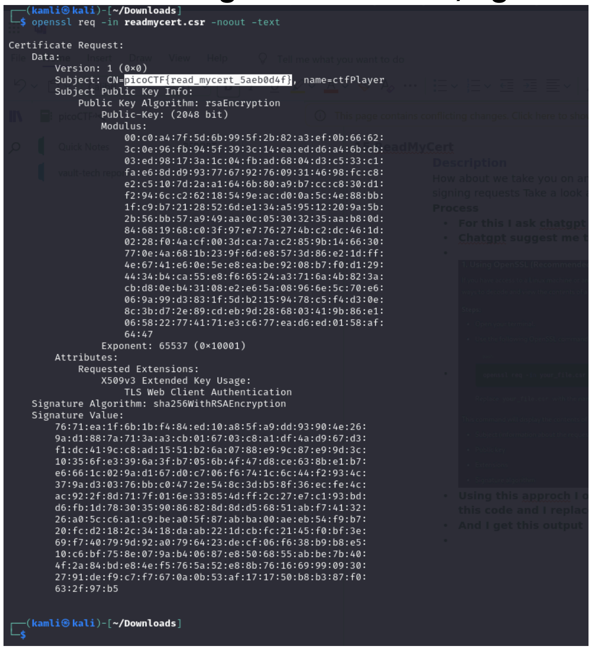
```
openssl req -in your_file.csr -noout -text
```

Replace `your_file.csr` with the name of the CSR file you downloaded.

This command will display the contents of the CSR, including:

- Subject (information about the requestor)
- Public key
- Extensions
- Signature algorithm

- **Using this approach, I open my Kali Linux terminal.**
- **I use the suggested OpenSSL command, replacing the file with the appropri**
- **After executing the command, I get the desired output**



```
┌──(kamli㉿kali)-[~/Downloads]
└─$ openssl req -in readmycert.csr -noout -text

Certificate Request:
    Data:
        Version: 1 (0×0)
        Subject: CN=picoCTF{read_mycert_5aeb0d4f}, name=ctfPlayer
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c0:a4:7f:5d:6b:99:5f:2b:82:a3:ef:0b:66:62:
                    3c:0e:96:fb:94:5f:39:3c:14:ea:ed:d6:a4:6b:cb:
                    03:ed:98:17:3a:1c:04:fb:ad:68:04:d3:c5:33:c1:
                    fa:e6:8d:d9:93:77:67:92:76:09:31:46:98:fc:c8:
                    e2:c5:10:7d:2a:a1:64:6b:80:a9:b7:cc:c8:30:d1:
                    f2:94:6c:c2:62:18:54:9e:ac:d0:0a:5c:4e:88:bb:
                    1f:c9:b7:21:28:52:6d:e1:34:a5:95:12:20:9a:5b:
                    2b:56:bb:57:a9:49:aa:0c:05:30:32:35:aa:b8:0d:
                    84:68:19:68:c0:3f:97:e7:76:27:4b:c2:dc:46:1d:
                    02:28:f0:4a:cf:00:3d:ca:7a:c2:85:9b:14:66:30:
                    77:0e:4a:68:1b:23:9f:6d:e8:57:3d:86:e2:1d:ff:
                    4e:67:41:e6:0e:5e:e8:ea:be:92:08:b7:f0:d1:29:
                    44:34:b4:ca:55:e8:f6:65:24:a3:71:6a:4b:82:3a:
                    cb:d8:0e:b4:31:08:e2:e6:5a:08:96:6e:5c:70:e6:
                    06:9a:99:d3:83:1f:5d:b2:15:94:78:c5:f4:d3:0e:
                    8c:3b:d7:2e:89:cd:eb:9d:28:68:03:41:9b:86:e1:
                    06:58:22:77:41:71:e3:c6:77:ea:d6:ed:01:58:af:
                    64:47
                Exponent: 65537 (0×10001)
        Attributes:
            Requested Extensions:
                X509v3 Extended Key Usage:
                    TLS Web Client Authentication
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        76:71:ea:1f:6b:1b:f4:84:ed:10:a8:5f:a9:dd:93:90:4e:26:
        9a:d1:88:7a:71:3a:a3:cb:01:67:03:c8:a1:df:4a:d9:67:d3:
        f1:dc:41:9c:c8:ad:15:51:b2:6a:07:88:e9:9c:87:e9:9d:3c:
        10:35:6f:e3:39:6a:3f:b7:05:6b:4f:47:d8:ce:63:8b:e1:b7:
        e6:66:1c:02:9a:d1:67:d0:c7:06:f6:74:1c:6c:44:f2:93:4c:
        37:9a:d3:03:76:bb:c0:47:2e:54:8c:3d:b5:8f:36:ec:fe:4c:
        ac:92:2f:8d:71:7f:01:6e:33:85:4d:ff:2c:27:e7:c1:93:bd:
        d6:fb:1d:78:30:35:90:86:82:8d:8d:d5:68:51:ab:f7:41:32:
        26:a0:5c:c6:a1:c9:be:a0:5f:87:ab:ba:00:ae:eb:54:f9:b7:
        20:fc:d2:18:2c:34:18:da:ab:22:1d:cb:fc:21:45:f0:bf:3e:
        69:f7:40:79:9d:92:a0:79:64:23:de:cf:06:f6:38:b9:b8:e5:
        10:c6:bf:75:8e:07:9a:b4:06:87:e8:50:68:55:ab:be:7b:40:
        4f:2a:84:bd:e8:4e:f5:76:5a:52:e8:8b:76:16:69:99:09:30:
        27:91:de:f9:c7:f7:67:0a:0b:53:af:17:17:50:b8:b3:87:f0:
        63:2f:97:b5

┌──(kamli㉿kali)-[~/Downloads]
└─$
```

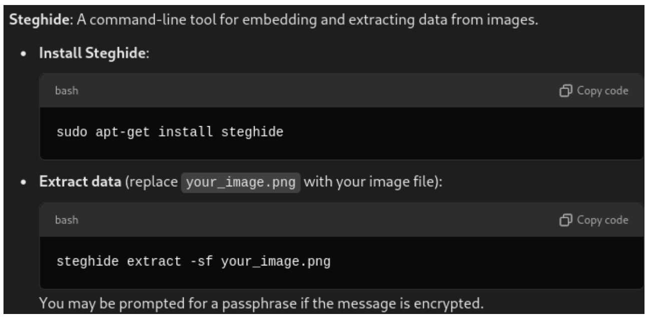
- **After executing the command, I get the output.**
- **Finally, I obtain the flag: picoCTF{read_mycert_5aeb0d4f}.**

16 HideToSee

Description

How about some hide and seek heh? Look at this image here.

Process

- **For this, I asked ChatGPT how to solve it.**
- **ChatGPT provided several approaches.**
- **I decided to use steghide to solve the challenge.**
- 

Steghide: A command-line tool for embedding and extracting data from images.

- **Install Steghide**:

bash Copy code

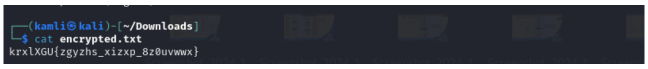
```
sudo apt-get install steghide
```

- **Extract data** (replace `your_image.png` with your image file):

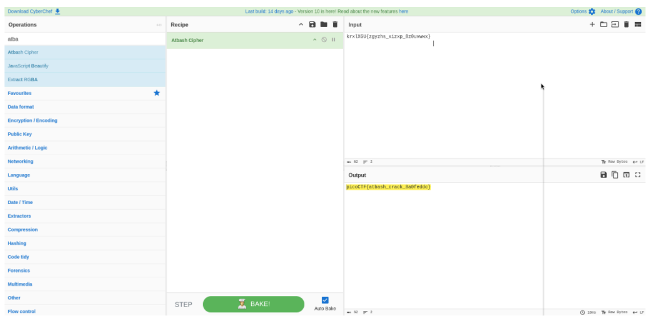
bash Copy code

```
steghide extract -sf your_image.png
```

You may be prompted for a passphrase if the message is encrypted.

- **I replace the image name as instructed.**
- **Running steghide generates an encrypted.txt file.**
- **I open the encrypted.txt file to check its contents.**
- 

```
┌──(kamli㉿kali)-[~/Downloads]
└─$ cat encrypted.txt
krxlXGU{zgyzhs_xizxp_8z0uvwwx}
```

- **I find a text that looks like a flag, but it seems scrambled.**
- **I use CipherChef, select the Atbash cipher, and apply it to the text.**
- **After decoding, I paste the resulting text.**
- **Finally, I get the correct flag.**
- 

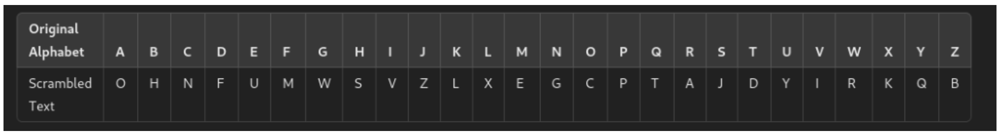
Input
```
krxlXGU{zgyzhs_xizxp_8z0uvwwx}
```
Output
```
picoCTF{atbash_crack_8a0feddc}
```

17 substitution0

Description

A message has come in but it seems to be all scrambled. Luckily it seems to have the
Can you crack this substitution cipher?

Process

- **According to the hint, I carefully observe the beginning of the message.**
- **I notice that each character represents an original letter of the alphabet.**
- **I create a table to map each character to its corresponding alphabet letter.**
- 

Original Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Scrambled Text	O	H	N	F	U	M	W	S	V	Z	L	X	E	G	C	P	T	A	J	D	Y	I	R	K	Q	B

- **At the end, the message looks like a flag.**
- **I compare that message to the table I created.**
- **After mapping the characters using the table, I obtain the flag:**
- **picoCTF{5UB5717U710N_3V0LU710N_03055505}**

18 substitution2

Description

It seems that another encrypted message has been intercepted. The encryptor se
their lesson though and now there isn't any punctuation! Can you still crack the cip

Process

- **For this, I go to dcode.fr.**
- **On the site, I select the "Monoalphabetic Substitution Decoder."**
- **I paste the encoded message into the decoder.**
- **After decoding, I get the flag:**
- **PICOCTF{N6R4M_4N41Y515_15_73D10U5_702F03FC}**

19 spelling-quiz

Description

I found the flag, but my brother wrote a program to encrypt all his text files. He has a s
guide too, but I don't know if that helps.

Process

- **First, I unzip the file.**
- **Inside, I find three files: two .txt files and one .py file.**

- **I analyze each file, but they don't make any sense.**
- **Then, I visit the website www.boxentriq.com to analyze the message.**
- **I paste the message into the website.**
- **The website tells me that it's a monoalphabetic cipher and provides a link to cipher tool.**
- **I go to the tool and paste words from quiz.txt. I also paste the encrypted flag**
- **After some trial and error, I manage to decode part of the message. Some se making sense. I put the sentence inside picoCTF{} and finally get the compl**
- **The final flag is:**
- **picoCTF{perhaps_the_dog_jumped_over_was_just_tired}**

Reverse Engineering section

---

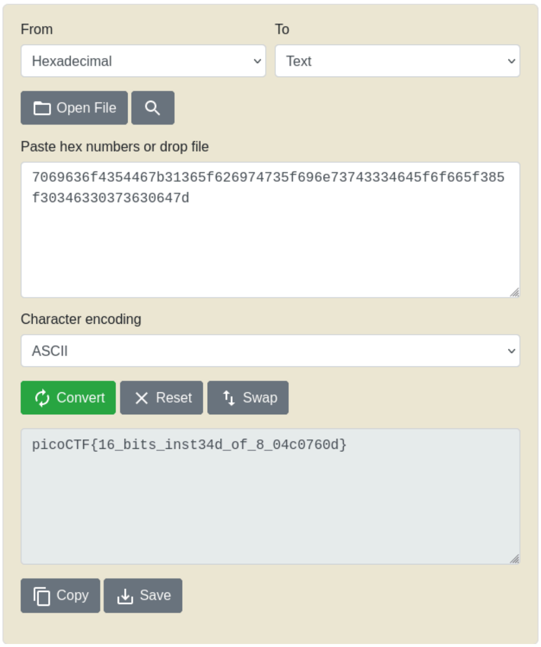
20 Transformation

Description

I wonder what this really is... [enc](enc) ''.join([chr((ord(flag[i]) << 8) + ord(flag[i + 1])) for i in 2)])

Process

- First I download the enc file
- Then I use a python script (**picoCTF 2021 Transformation**)
-

- Here I get a hex text I copy this and use a web site that convert this hex to ascii
- **https://www.rapidtables.com/convert/number/hex-to-ascii.html?x=**
- 

From: Hexadecimal
To: Text

Paste hex numbers or drop file

```
7069636f4354467b31365f626974735f696e73743334645f6f665f385
f30346330373630647d
```

Character encoding: ASCII

```
picoCTF{16_bits_inst34d_of_8_04c0760d}
```

- Finally I get the flag
- **picoCTF{16_bits_inst34d_of_8_04c0760d}**

21 bloat.py

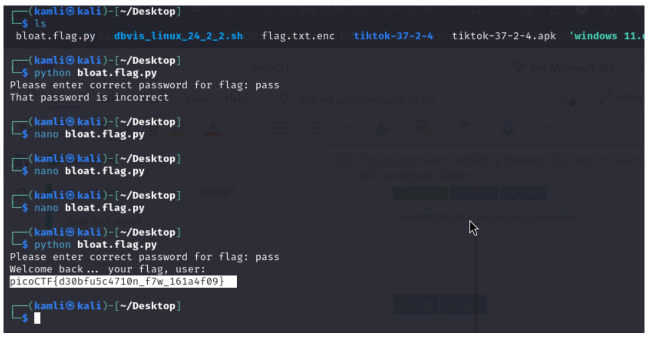
Description

Can you get the flag? Run this Python program in the same directory as this encrypte

Process

- **First, I download the files.**
- **I try to run the program, but my password is incorrect multiple times.**
- **Then, I remove the part of the code where it handles the login process.**
- **After modifying the code, I try logging in again and successfully log in.**
- **I find a YouTube video that helps me locate the flag: picoCTF 2022 | Reve bloat.py.**
-

- **Finally, I find the flag.**



```
┌──(kamli㉿kali)-[~/Desktop]
└─$ ls
 bloat.flag.py  dbvis_linux_24_2_2.sh  flag.txt.enc  tiktok-37-2-4  tiktok-37-2-4.apk 'windows 11.

┌──(kamli㉿kali)-[~/Desktop]
└─$ python bloat.flag.py
Please enter correct password for flag: pass
That password is incorrect

┌──(kamli㉿kali)-[~/Desktop]
└─$ nano bloat.flag.py

┌──(kamli㉿kali)-[~/Desktop]
└─$ nano bloat.flag.py

┌──(kamli㉿kali)-[~/Desktop]
└─$ nano bloat.flag.py

┌──(kamli㉿kali)-[~/Desktop]
└─$ python bloat.flag.py
Please enter correct password for flag: pass
Welcome back... your flag, user:
picoCTF{d30bfu5c4710n_f7w_161a4f09}

┌──(kamli㉿kali)-[~/Desktop]
└─$ 
```

22 Bbbbloat

Description

Can you get the flag? Reverse engineer this binary.

Process

- **First, I download the executable file.**
- **When I try to execute it, it asks for some number input, but I don't have that**
- **So, I decide to use reverse engineering to find the solution.**
- **I find a YouTube video that helps me with the reverse engineering process.**
- picoCTF 2022 | Reverse Engineering | Bbbbloat

- I follow this step and I get the numer
- Finally I found the flag
- **picoCTF{cu7_7h3_bl047_695036e3}**

23 crackme-py

Description

crackme.py

Process

- **First, I download the file.**
- **I try to understand the challenge, but it's not clear at first.**
- **I find a YouTube video that helps me solve the challenge: picoCTF Walkt py.**
-

- **After following the video, I find the flag:**
- **picoCTF{1|\/|_4_p34|\|ut_a79b6c2d}**
-

24 Shop

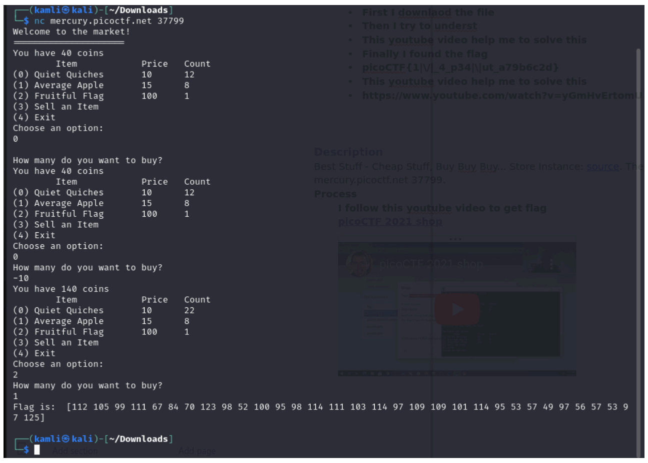
Description

Best Stuff - Cheap Stuff, Buy Buy Buy... Store Instance: source. The shop is open for mercury.picoctf.net 37799.

Process

I follow this youtube video to get flag

picoCTF 2021 shop



```
┌──(kamli㉿kali)-[~/Downloads]
└─$ nc mercury.picoctf.net 37799
Welcome to the market!
=====================
You have 40 coins
        Item            Price   Count
(0) Quiet Quiches       10      12
(1) Average Apple       15      8
(2) Fruitful Flag       100     1
(3) Sell an Item
(4) Exit
Choose an option:
0

How many do you want to buy?
You have 40 coins
        Item            Price   Count
(0) Quiet Quiches       10      12
(1) Average Apple       15      8
(2) Fruitful Flag       100     1
(3) Sell an Item
(4) Exit
Choose an option:
0
How many do you want to buy?
-10
You have 140 coins
        Item            Price   Count
(0) Quiet Quiches       10      22
(1) Average Apple       15      8
(2) Fruitful Flag       100     1
(3) Sell an Item
(4) Exit
Choose an option:
2
How many do you want to buy?
1
Flag is:  [112 105 99 111 67 84 70 123 98 52 100 95 98 114 111 103 114 97 109 109 101 114 95 53 57 49 97 56 57 53 9
7 125]

┌──(kamli㉿kali)-[~/Downloads]
└─$ █
```

Image upload failed. Please try again

picoCTF{b4d_brogrammer_591a895a}

25 " format string_0"

Category: Binary Exploitation

Problem Description: The challenge requires exploiting a heap overflow vulneral
source code to retrieve the flag.

Approach:

1. **Run the program** and enter a string longer than 64 bytes. This causes a heap
2. **After exploiting the heap overflow**, check for the flag.
3. **Finally, I retrieve the flag.**

Flag: picoCTF{7h3_cu570m3r_15_n3v3r_SEGFAULT_74f6c0e7}

26 "heap 1"

Category: Binary Exploitation

Problem Description: The challenge requires setting a variable str_var to "pico" i
flag.

Approach:

1. **Run the program** and inspect what the str_var is.
2. To set str_var to "pico," **add data to the buffer**. You need to provide input that
   manipulates the value of str_var.
3. **Enter a string** of the form p%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%x%p
   the format string vulnerability to overwrite str_var with "pico".
4. After entering the correct input, **check for the flag** and successfully retrieve it.

Flag: picoCTF{starting_to_get_the_hang_ce5bee9b}

27 "binary Search"

Category: General Skill

Problem Description: The challenge asks you to guess a number between 1 anc
chances to retrieve the flag.

Approach:

1. **Run the program** and begin guessing numbers one by one.
2. Since the program follows a **binary search algorithm**, it uses a divide-and-co
   narrow down the possible range of numbers.
3. By leveraging this approach, you can efficiently guess the correct number by h
   space with each guess.

4. After making the correct guess on the **10th attempt**, you retrieve the flag.

Flag: picoCTF{g00d_gu355_2e90d29b}

28 "time machine"

Category: General Skill

Problem Description: The challenge asks you to find flags within git files.

Approach:

1. **Download the challenge.zip file** and unzip it.
2. After unzipping, **navigate to the drop-in directory** created.
3. Once inside the directory, **run the command git log** to view the commit histor
4. From the git logs, you can identify and retrieve the flag.

Flag: picoCTF{t1m3m@ch1n3_b476ca06}

29 "super ssh"

Category: General Skill

Problem Description: The challenge requires you to log in via the shell to retriev

Approach:

1. **Open the terminal** and log in as the ctf-player user.
2. The system may ask for **fingerprint verification**. Grant the necessary permiss authentication.
3. **Enter the password** when prompted and successfully log in.
4. Once logged in, **retrieve the flag**.

Flag: picoCTF{s3cur3_c0nn3ct10n_07a987ac}

30 "endianness"

Category: General Skill

Problem Description: The challenge asks you to determine the small-endian anc representations of a word for the flag.

Approach:

1. **Open the terminal** and run the provided code.
2. **Provide a word** as input.
3. Visit a website like **CipherChef** and paste the word into the tool.
4. Select the **hex filter** to convert the word into its hexadecimal representation.
5. Once you have the hexadecimal code, apply the following transformations:
   - For **small-endian**, reverse the byte order, placing the smaller digits first. Fc output is 78953f4521, the small-endian format would be 21453f9578.
   - For **big-endian**, the hexadecimal code remains unchanged, such as 7895З
6. Using the above process, you will find the correct representation and retrieve t
7. **Flag: picoCTF{3ndi4n_sw4p_su33ess_02999450}**

31 "commitment issue"

Category: General Skill

Problem Description: The challenge requires you to find the flag by examining tl

Approach:

1. **Download the challenge.zip file** and unzip it.
2. Navigate to the **drop-in directory** where the files are located.
3. Run the command git log to view the commit history. You will find a code (com
4. Use the command git checkout <commit-hash> to check out the specific comn
5. Open the **message.txt** file using the cat command to reveal the flag.

Flag: picoCTF{s@n1t1z3_cf09a485}

33 Glory of the Garden (Forensic)

Description

This garden contains more than it seems.

Process

- **Download the challenge file** (which is an image).
- Visit the website **StegOnline** (https://www.stegonline.com/).
- Upload the image to the website.
- Use the website's tools to **extract the hidden data** from the image.
- After extraction, you will find the flag hidden within the image.
- **Flag:**
- **picoCTF{more_than_m33ts_the_3y3657BaB2C}**

34 "heap0"Category: Binary Exploitation

Problem Description: The challenge asks you to find a way to exploit a heap overflow v provided source code and obtain the flag.

Approach:

i. **Run the program** and select **option 2** from the available choices.
ii. **Enter a string** that is larger than **64 bytes** (to trigger the heap overflow).
iii. As the heap gets exploited, the program will reveal the flag.

Flag: picoCTF{my_first_heap_overflow_1ad0e1a6}

35 " big zip"

Category: General Skill

Problem Description: The challenge requires you to find the flag by using the grep com

Approach:

1. **Download the file** using the provided link.
2. **Unzip the file** to access its contents.
3. **Use the grep command** to search for the term "pico" in the files: grep -rl "pico" big-zi
4. This command will return the path of a .txt file containing the flag. In this case, the pat
5. big-zip-files/folder_pmbymkjcya/folder_cawigcwvgv/folder_ltdayfmktr/folder_fnpfclfyee
6. **Open the file** using the cat command to view the content and find the flag:
7. cat big-zip-files/folder_pmbymkjcya/folder_cawigcwvgv/folder_ltdayfmktr/folder_fnpfcl
8. Flag: picoCTF{gr3p_15_m4g1c_ef8790dc}

36 "first find"

Category: General Skill

Problem Description: The task is to find the flag using the filename “uber-secret.txt”.

Approach:

1. **Download the file** using the provided link.
2. **Unzip the file** to access its contents.
3. **Use the find command** to locate the file named "uber-secret.txt": find files | grep ub
4. The command will return the path of the uber-secret.txt file. In this case, the file path
   be: files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/uber-
5. **Open the file** using the cat command to view its contents and extract the flag: cat files/adequate_books/more_books/.secret/deeper_secrets/deepest_secrets/uber-sec
6. Flag: picoCTF{f1nd_15_f457_ab443fd1}

37 "Insp3ct0r"

Category: Web Exploitation

Problem Description: The goal is to inspect the web page and extract the flag from the

Approach:

1. **Open the Web Page:** Navigate to the web page provided in the challenge.
2. **Inspect the Source Code:**
   - Press Ctrl + U (or right-click the page and select "View Page Source") to open the so
     page.
3. **Locate Flag Parts:**
   - **First Part of the Flag:** Search within the HTML part of the source code for the first p
     be visible as plain text or within an HTML tag.
   - **Second Part of the Flag:** Look for the .css file linked in the HTML code. Open the lin
     search for the second part of the flag.
   - **Final Part of the Flag:** Similarly, locate the .js file referenced in the source code. Op
     for the final part of the flag.
4. **Combine the Flag Parts:** Once you have all the parts from the HTML, CSS, and JS
   form the full flag.
   Flag: picoCTF{tru3_d3t3ct1ve_0r_ju5t_lucky?f10be399}

38 "where are the robots"

Category: Web Exploitation

Problem Description: The goal is to find the secret files hidden on a web page.

Approach:

1. **Check the robots.txt File:**
   - Open the web page provided in the challenge.
   - After the URL of the web page, add /robots.txt. For example: http://example.com/robo
   - The robots.txt file typically contains directives for search engines about which pages
     It may also list secret files or directories that are intentionally hidden.

2 Look for Secret Files in robots.txt:

- In the robots.txt file, search for paths or files that are not intended to be publicly index
  These files could contain parts of the flag or additional clues.

- For example, you might find something like: Disallow: /477ce.html

3 Access the Secret File:

- Based on the information from robots.txt, go to the secret file by appending the file pa URL. For example:
- http://example.com/477ce.html

4 Retrieve the Flag:

- Once you access the file, you should find the flag.

Flag: picoCTF{ca1cu1at1ng_Mach1n3s_477ce}

39 "packer"

Category: Reverse Engineering

Problem Description: You are tasked with finding the secret files hidden in the pages of

Approach:

- **Download the File:**
  - First, download the file using the link provided in the challenge. This file is likely ar binary file.
- **Make the File Executable:**
  - Open a terminal in the directory where the file was downloaded.
  - Run the following command to make the file executable:
  - chmod +x out
  - This ensures that the file has the proper permissions to be executed.
- **Extract Strings from the File:**
  - Next, use the strings command to extract readable text from the binary file. The grep used to search for the term "flag" within the output. Run the following command: strin
  - This will return an encrypted or encoded string related to the flag. The output could lo hex or another encoded format
- **Decode the Flag:**
  - Once you have the encrypted or encoded flag, head to a website like **CipherChef** (https://www.cipherchef.com) for decoding.
  - Paste the encrypted flag into the tool and select the appropriate filter (for example, "F is in hexadecimal format).
  - The tool will decode the flag for you.
- **Retrieve the Flag:**
  - Once decoded, you will obtain the flag in its readable form.
- Flag: picoCTF{U9X_UnP4ck1N6_B1n4Ri3S_6ff964ef}

40 "PcapPoisoning"

Category: Forensics

Problem Description: You are tasked with finding the flag hidden in a .pcap (packet cap

Approach:

1. **Download the File:**
   - Start by downloading the .pcap file (named trace.pcap) using the provided link.
2. **Extract Strings from the .pcap File:**
   - Use the strings command to extract readable strings from the .pcap file. This com sequences of printable characters in binary files.
   - In the terminal, run the following command: strings trace.pcap | grep picoCTF

- This command will search through the strings extracted from the .pcap file and loc
  of the string "picoCTF."

3 Find the Flag:

- After running the command, you will find the flag within the output, as it will match
  picoCTF{...}.
- Flag: picoCTF{P64P_4N4L7S1S_SU55355FUL_f621fa37}