# Assignment1

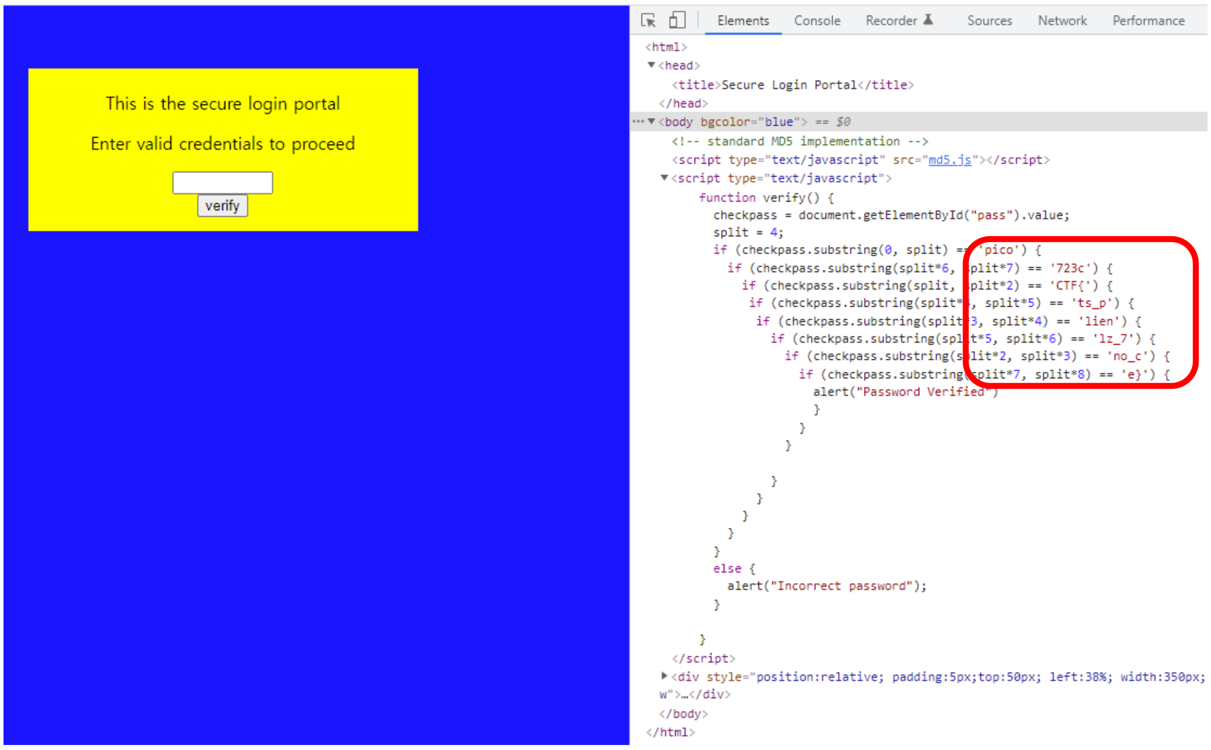
## 1. Web Exploitation

### 1.1 **dont-use-client-side**

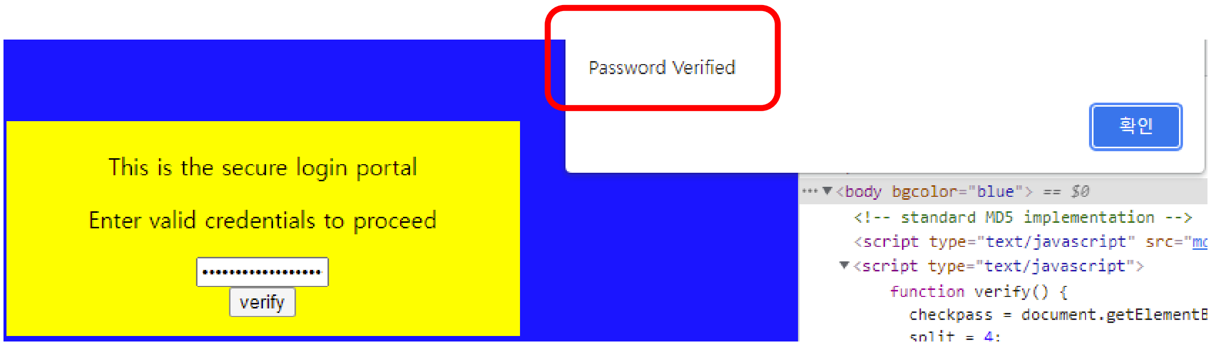
. **Flag**: picoCTF{no_clients_plz_7723ce}

. **Approach**

Step1: When I opened Chrome Devtools to look at the code, I stumbled across the flag. The flag was scattered along with the serial numbers in the code.


This is the secure login portal
Enter valid credentials to proceed
verify
Elements
Console
Recorder
Sources
Network
Performance

```
<html>
▼<head>
   <title>Secure Login Portal</title>
 </head>
…▼<body bgcolor="blue"> == $0
   <!-- standard MD5 implementation -->
   <script type="text/javascript" src="md5.js"></script>
  ▼<script type="text/javascript">
      function verify() {
        checkpass = document.getElementById("pass").value;
        split = 4;
        if (checkpass.substring(0, split) == 'pico') {
          if (checkpass.substring(split*6, split*7) == '723c') {
            if (checkpass.substring(split, split*2) == 'CTF{') {
             if (checkpass.substring(split*4, split*5) == 'ts_p') {
              if (checkpass.substring(split*3, split*4) == 'lien') {
                if (checkpass.substring(split*5, split*6) == 'lz_7') {
                  if (checkpass.substring(split*2, split*3) == 'no_c') {
                    if (checkpass.substring(split*7, split*8) == 'e}') {
                      alert("Password Verified")
                      }
                    }
                  }


                }
              }
            }
          }
        }
        else {
          alert("Incorrect password");
        }

      }
   </script>
  ▶<div style="position:relative; padding:5px;top:50px; left:38%; width:350px;
  w">…</div>
  </body>
</html>
```


This is the secure login portal
Enter valid credentials to proceed
verify
확인

```
Password Verified
```

```
…▼<body bgcolor="blue"> == $0
    <!-- standard MD5 implementation -->
    <script type="text/javascript" src="md
   ▼<script type="text/javascript">
       function verify() {
         checkpass = document.getElementE
         split = 4;
```

## 1.2 picobrowser

. **Flag**: picoCTF{p1c0_s3cr3t_ag3nt_51414fa7}

. **Approach**

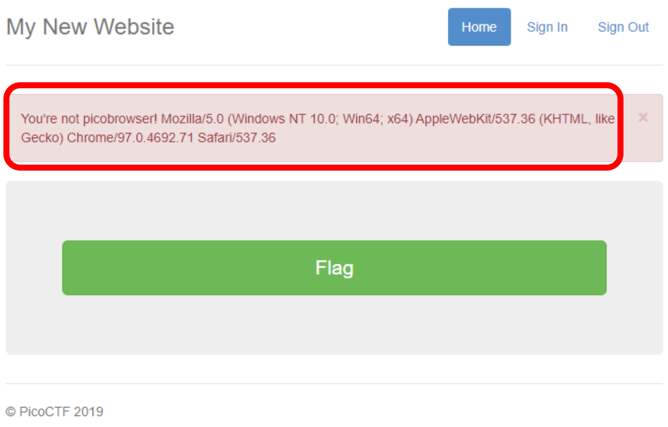
Step1: The message "You're not picobrowser" was showed up with my browser type when I clicked Flag button.

Step2: So I checked User-agent option in Chrome Devtools with reloading. There was a flag document with stylesheet, script in Network tab.

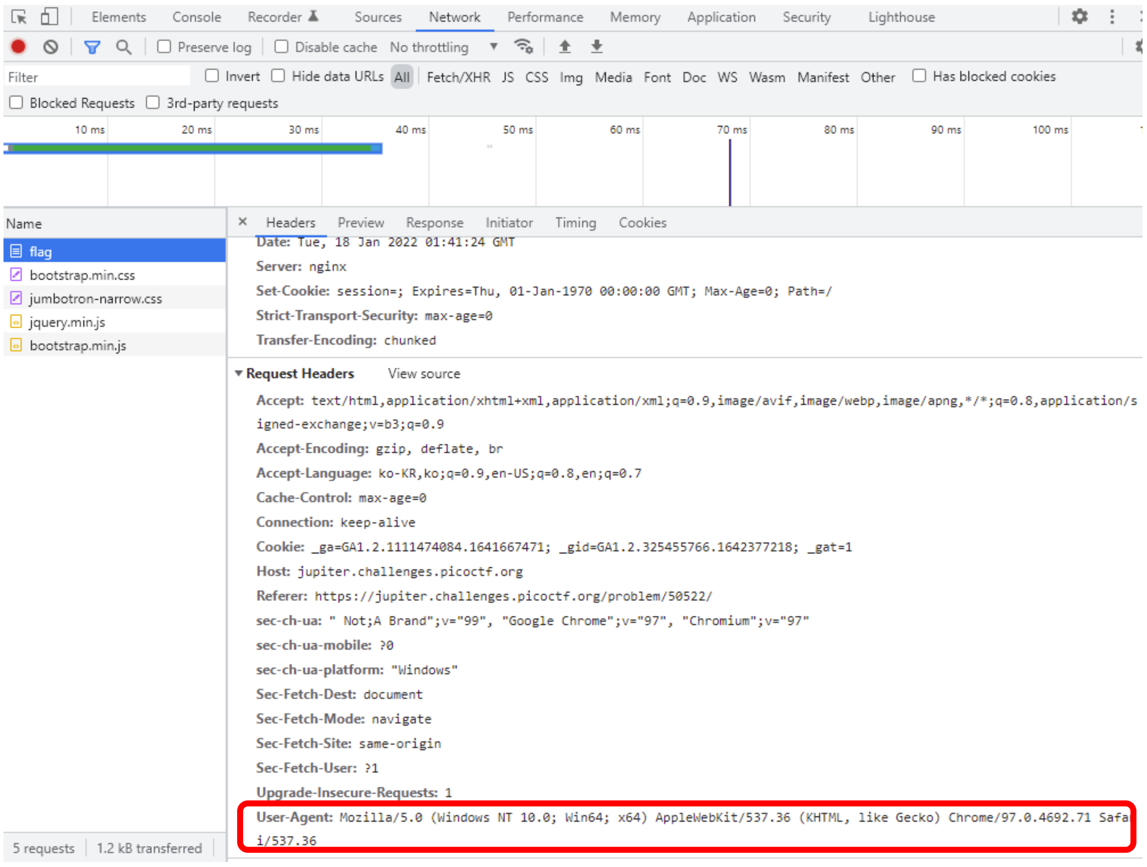
Step3: I tried to update User-agent type of "picobrowser" with curl command in my Ubuntu terminal. And I found the flag in the terminal after updating the User-agent type.

Reference site for curl command: https://reqbin.com/req/c-ekublyqq/curl-user-agent

Command: curl --user-agent "picobrowser" https://jupiter.challenges.picoctf.org/problem/50522/flag

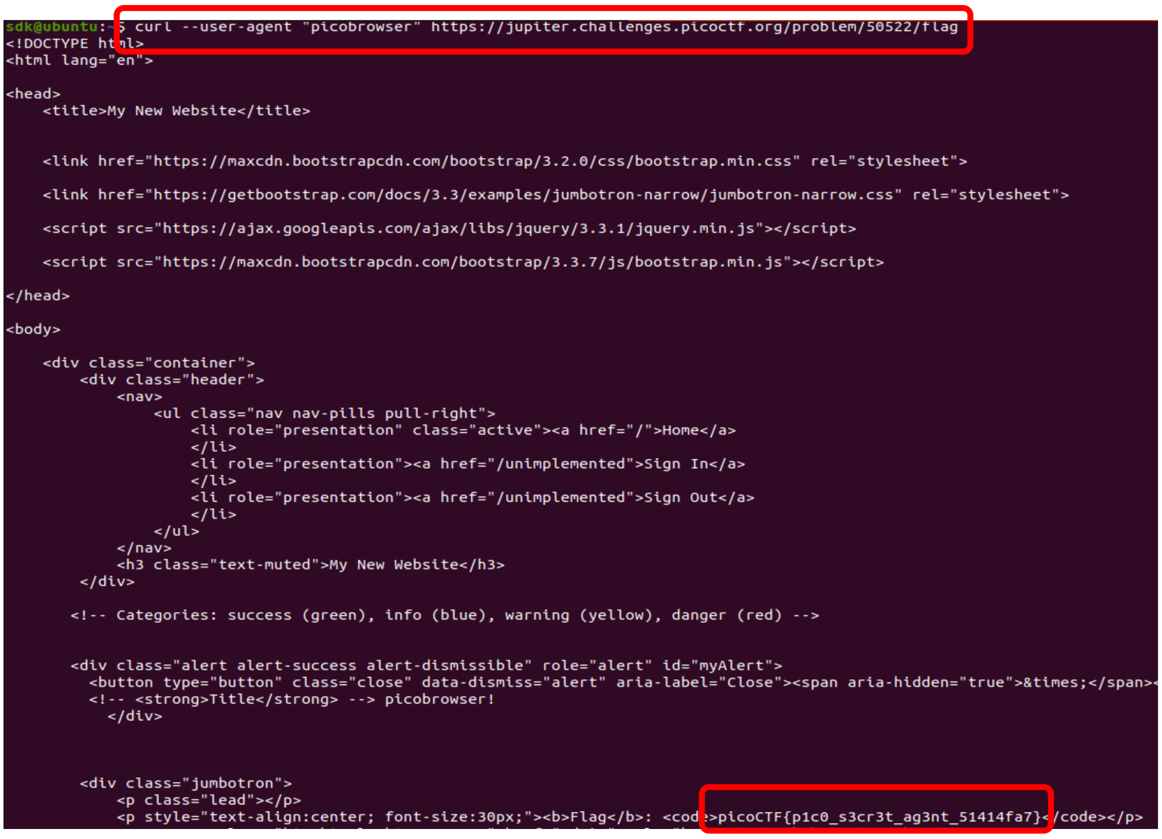


You're not picobrowser! Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/537.36



```
flag
bootstrap.min.css
jumbotron-narrow.css
jquery.min.js
bootstrap.min.js
```

```
Date: Tue, 18 Jan 2022 01:41:24 GMT
Server: nginx
Set-Cookie: session=; Expires=Thu, 01-Jan-1970 00:00:00 GMT; Max-Age=0; Path=/
Strict-Transport-Security: max-age=0
Transfer-Encoding: chunked
▼ Request Headers     View source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Cookie: _ga=GA1.2.1111474084.1641667471; _gid=GA1.2.325455766.1642377218; _gat=1
Host: jupiter.challenges.picoctf.org
Referer: https://jupiter.challenges.picoctf.org/problem/50522/
sec-ch-ua: " Not;A Brand";v="99", "Google Chrome";v="97", "Chromium";v="97"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safar
i/537.36
```



```
sdk@ubuntu:~$ curl --user-agent "picobrowser" https://jupiter.challenges.picoctf.org/problem/50522/flag
<!DOCTYPE html>
<html lang="en">

<head>
    <title>My New Website</title>


    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css" rel="stylesheet">

    <link href="https://getbootstrap.com/docs/3.3/examples/jumbotron-narrow/jumbotron-narrow.css" rel="stylesheet">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

</head>

<body>

    <div class="container">
        <div class="header">
            <nav>
                <ul class="nav nav-pills pull-right">
                    <li role="presentation" class="active"><a href="/">Home</a>
                    </li>
                    <li role="presentation"><a href="/unimplemented">Sign In</a>
                    </li>
                    <li role="presentation"><a href="/unimplemented">Sign Out</a>
                    </li>
                </ul>
            </nav>
            <h3 class="text-muted">My New Website</h3>
        </div>

       <!-- Categories: success (green), info (blue), warning (yellow), danger (red) -->


       <div class="alert alert-success alert-dismissible" role="alert" id="myAlert">
         <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span><
         <!-- <strong>Title</strong> --> picobrowser!
           </div>



        <div class="jumbotron">
            <p class="lead"></p>
            <p style="text-align:center; font-size:30px;"><b>Flag</b>: <code>picoCTF{p1c0_s3cr3t_ag3nt_51414fa7}</code></p>
```

## 2. Cryptography

### 2.1 la cifra de

. **Flag**: picoCTF{b311a50_0r_v1gn3r3_c1ph3ra966878a}

. **Approach**:

Step1: I had the cipher texts from the command in the question.

Step2: I did not know what kind of cipher it was, so I tried to figure it out. By applying the ciphertext in a cipher identifier site, I obtained a list of cipher types with high possible ranking from the site.

Step3: I tried to find out the cipher type from the list with the entire cipher text, but I could not find it.

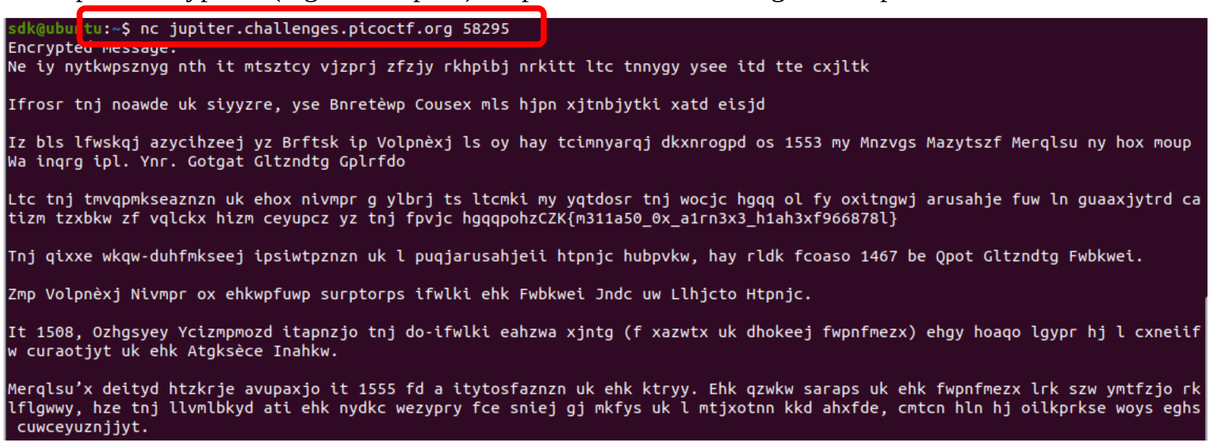
Step4: So I tried to find the cipher type with only one sentence of the cipher text.

Step5: The Substituion cipher asked for a dictionary and I could not try it because I did not know it. Next I tried the Vigenere cipher and it showed me the decrypted English sentence.

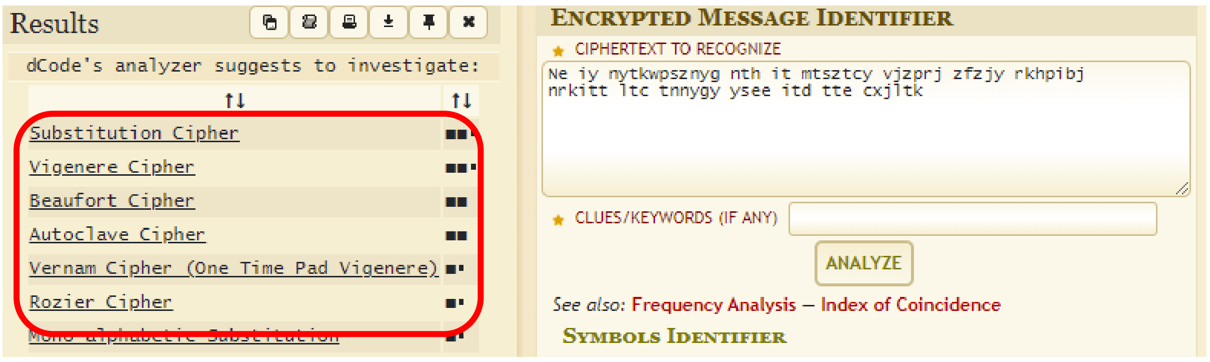
Step6: I found the flag by trying a sentence with curly brackets that might be the flag.

Cipher identifier site: https://www.dcode.fr/cipher-identifier

Cipher decrypt site(vigenere-cipher): https://www.dcode.fr/vigenere-cipher



```
sdk@ubuntu:~$ nc jupiter.challenges.picoctf.org 58295
Encrypted message:
Ne iy nytkwpsznyg nth it mtsztcy vjzprj zfzjy rkhpibj nrkitt ltc tnnygy ysee itd tte cxjltk

Ifrosr tnj noawde uk siyyzre, yse Bnretèwp Cousex mls hjpn xjtnbjytki xatd eisjd

Iz bls lfwskqj azycihzeej yz Brftsk ip Volpnèxj ls oy hay tcimnyarqj dkxnrogpd os 1553 my Mnzvgs Mazytszf Merqlsu ny hox moup
Wa inqrg ipl. Ynr. Gotgat Gltzndtg Gplrfdo

Ltc tnj tmvqpmkseaznzn uk ehox nivmpr g ylbrj ts ltcmki my yqtdosr tnj wocjc hgqq ol fy oxitngwj arusahje fuw ln guaaxjytrd ca
tizm tzxbkw zf vqlckx hizm ceyupcz yz tnj fpvjc hgqqpohzCZK{m311a50_0x_a1rn3x3_h1ah3xf966878l}

Tnj qixxe wkqw-duhfmkseej ipsiwtpznzn uk l puqjarusahjeii htpnjc hubpvkw, hay rldk fcoaso 1467 be Qpot Gltzndtg Fwbkwei.

Zmp Volpnèxj Nivmpr ox ehkwpfuwp surptorps ifwlki ehk Fwbkwei Jndc uw Llhjcto Htpnjc.

It 1508, Ozhgsyey Ycizmpmozd itapnzjo tnj do-ifwlki eahzwa xjntg (f xazwtx uk dhokeej fwpnfmezx) ehgy hoaqo lgypr hj l cxneiif
w curaotjyt uk ehk Atgksèce Inahkw.

Merqlsu’x deityd htzkrje avupaxjo it 1555 fd a itytosfaznzn uk ehk ktryy. Ehk qzwkw saraps uk ehk fwpnfmezx lrk szw ymtfzjo rk
lflgwwy, hze tnj llvmlbkyd ati ehk nydkc wezypry fce sniej gj mkfys uk l mtjxotnn kkd ahxfde, cmtcn hln hj oilkprkse woys eghs
 cuwceyuznjjyt.
```



Results

dCode's analyzer suggests to investigate:

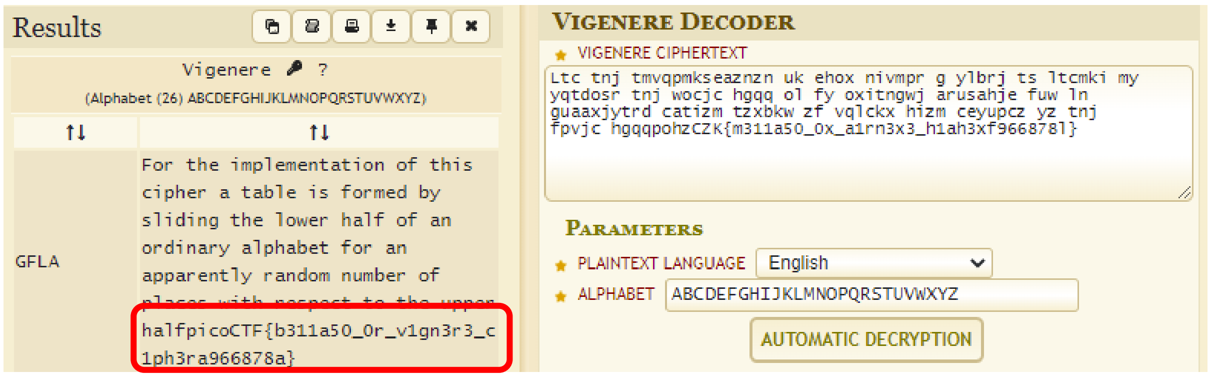
Substitution Cipher
Vigenere Cipher
Beaufort Cipher
Autoclave Cipher
Vernam Cipher (One Time Pad Vigenere)
Rozier Cipher
Mono-alphabetic Substitution

ENCRYPTED MESSAGE IDENTIFIER

CIPHERTEXT TO RECOGNIZE

Ne iy nytkwpsznyg nth it mtsztcy vjzprj zfzjy rkhpibj nrkitt ltc tnnygy ysee itd tte cxjltk

CLUES/KEYWORDS (IF ANY)


Results

Vigenere 🔑 ?

(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)

| ⇅ | ⇅ |
|---|---|
| GFLA | For the implementation of this cipher a table is formed by sliding the lower half of an ordinary alphabet for an apparently random number of places with respect to the upper halfpicoCTF{b311a50_0r_v1gn3r3_c1ph3ra966878a} |

VIGENERE DECODER

VIGENERE CIPHERTEXT

```
Ltc tnj tmvqpmkseaznzn uk ehox nivmpr g ylbrj ts ltcmki my
yqtdosr tnj wocjc hgqq ol fy oxitngwj arusahje fuw ln
guaaxjytrd catizm tzxbkw zf vqlckx hizm ceyupcz yz tnj
fpvjc hgqqpohzCZK{m311a50_0x_a1rn3x3_h1ah3xf966878l}
```

PARAMETERS

PLAINTEXT LANGUAGE: English

ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

## 2.2 john_pollard

. **Flag**: picoCTF{73176001,67867967}

. **Approach**:

Step1. public key and private key in RSA are created by large prime factors, p and q, and modulus n (n=p*q).

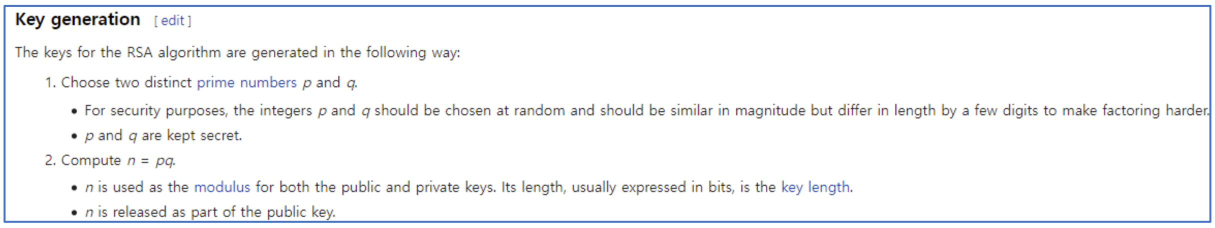
public key consists of n(modulus) and e(exponent), and the key can be derived from certificate.

n can be derived from public key. p and q can be obtained from prime factorization of n.

So the process to get p,q is like this: certificate -> public key -> prime factorization with n -> p,q

The hint of in the problem says that the flag is in the format picoCTF{p,q}.

Reference site: https://en.wikipedia.org/wiki/RSA_(cryptosystem)


**Key generation** [ edit ]

The keys for the RSA algorithm are generated in the following way:

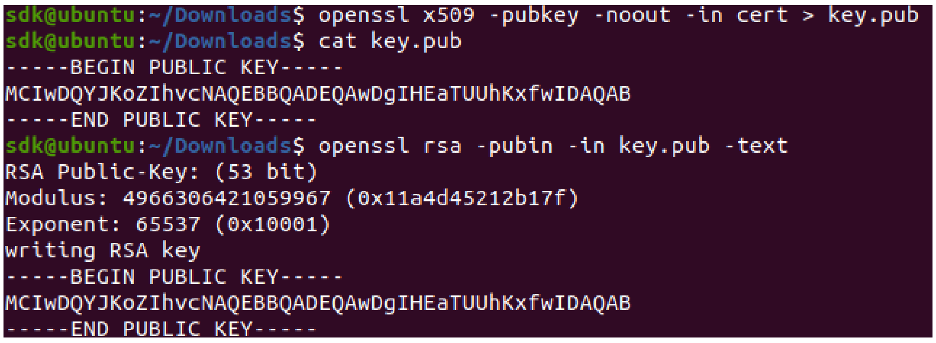
1. Choose two distinct prime numbers *p* and *q*.
   - For security purposes, the integers *p* and *q* should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   - *p* and *q* are kept secret.
2. Compute *n* = *pq*.
   - *n* is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   - *n* is released as part of the public key.

Step2. openssl was used to get public key, and n(modulus) in PKCS#1 format

. openssl x509 -pubkey -noout -in cert > key.pub

. openssl rsa -pubin -in key.pub -text

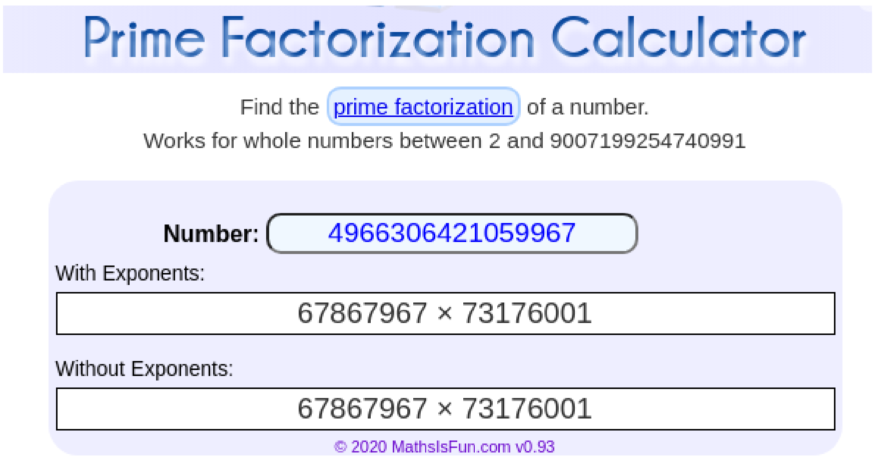
Reference site: https://superuser.com/questions/1644533/how-do-i-use-the-openssl-command-to-decode-a-public-key-pem-file


```
sdk@ubuntu:~/Downloads$ openssl x509 -pubkey -noout -in cert > key.pub
sdk@ubuntu:~/Downloads$ cat key.pub
-----BEGIN PUBLIC KEY-----
MCIwDQYJKoZIhvcNAQEBBQADEQAwDgIHEaTUUhKxfwIDAQAB
-----END PUBLIC KEY-----
sdk@ubuntu:~/Downloads$ openssl rsa -pubin -in key.pub -text
RSA Public-Key: (53 bit)
Modulus: 4966306421059967 (0x11a4d45212b17f)
Exponent: 65537 (0x10001)
writing RSA key
-----BEGIN PUBLIC KEY-----
MCIwDQYJKoZIhvcNAQEBBQADEQAwDgIHEaTUUhKxfwIDAQAB
-----END PUBLIC KEY-----
```

Step3. p and q were derived from the prime factorization calculator site.

n(modulus) is 4966306421059967 and p,q can be derived from prime factorization calculation.

Reference site for calculation: https://www.mathsisfun.com/numbers/prime-factorization-tool.html



Prime Factorization Calculator

Find the prime factorization of a number.
Works for whole numbers between 2 and 9007199254740991

Number: 4966306421059967

With Exponents:

67867967 × 73176001

Without Exponents:

67867967 × 73176001

© 2020 MathsIsFun.com v0.93

p,q can be 67867967, 73176001.

Setp4. The flag could be picoCTF{67867967,73176001} by the hint, "the format picoCTF{p,q}". But it doesn't work. So try swapping the numbers by another hint, "try swapping p and q if it does not work". The flag is picoCTF{73176001,67867967}

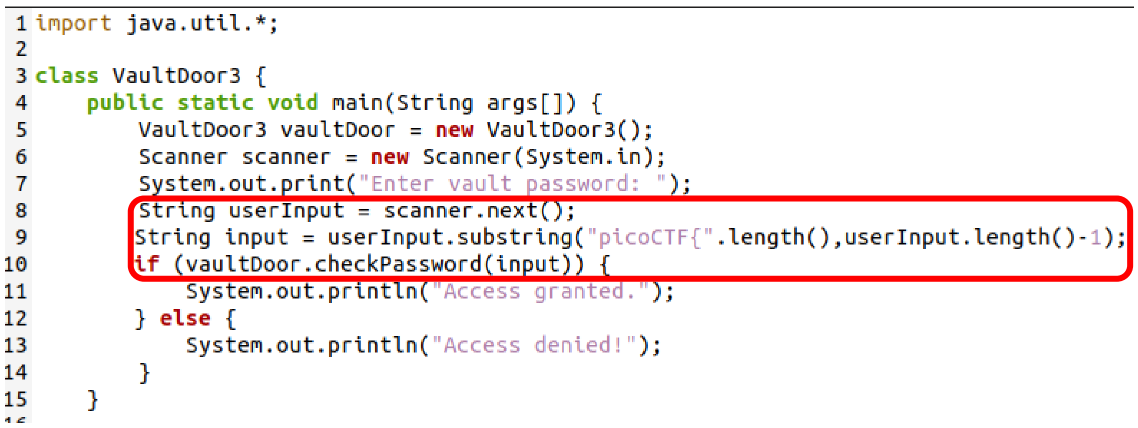
## 3. Reverse Engineering

### 3.1 vault-door3

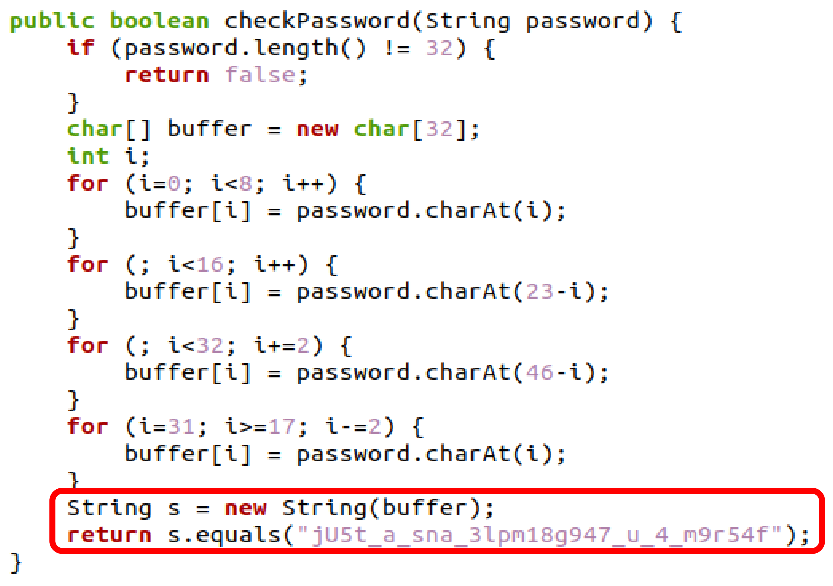
. **Flag**: picoCTF{jU5t_a_s1mpl3_an4gr4m_4_u_79958f}

. **Approach**:

Step1. The java program, VaultDoor3 has two functions. One is main to scan from user's input and then make it input as picoCTF string to check password. The other is to check if password is true or false with a string, "jU5t_a_sna_3lpm18g947_u_4_m9r54f" after looping process.

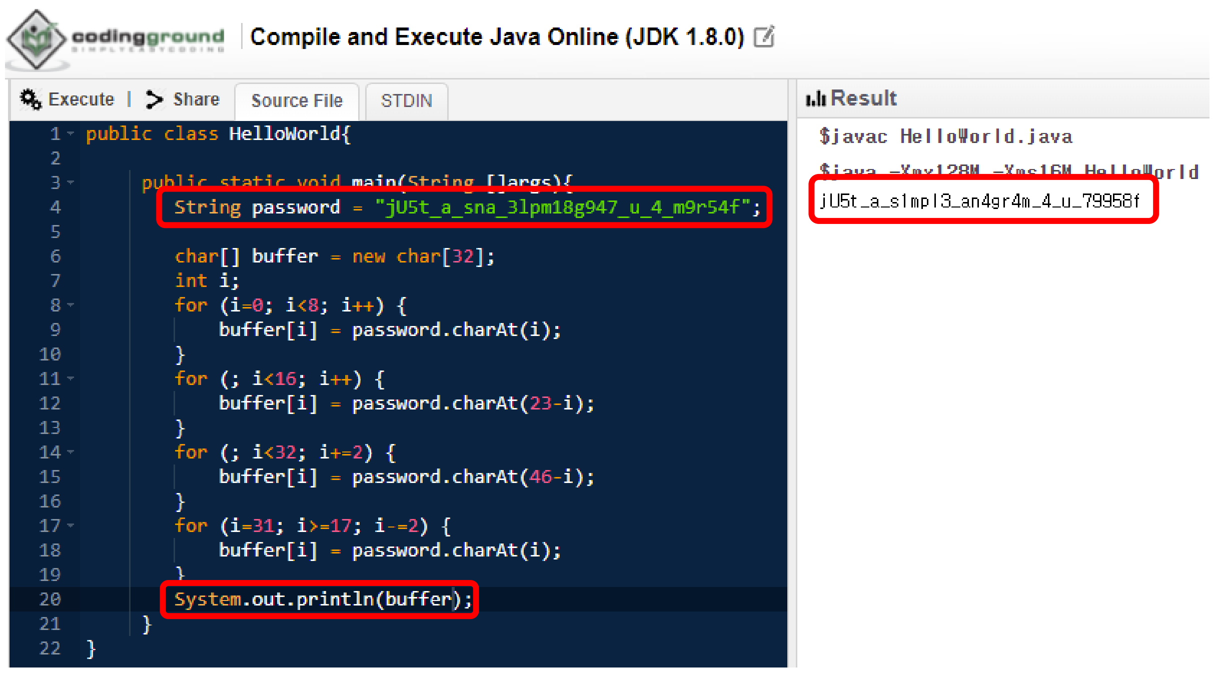


```
import java.util.*;

class VaultDoor3 {
    public static void main(String args[]) {
        VaultDoor3 vaultDoor = new VaultDoor3();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter vault password: ");
        String userInput = scanner.next();
        String input = userInput.substring("picoCTF{".length(),userInput.length()-1);
        if (vaultDoor.checkPassword(input)) {
            System.out.println("Access granted.");
        } else {
            System.out.println("Access denied!");
        }
    }
```



```java
public boolean checkPassword(String password) {
    if (password.length() != 32) {
        return false;
    }
    char[] buffer = new char[32];
    int i;
    for (i=0; i<8; i++) {
        buffer[i] = password.charAt(i);
    }
    for (; i<16; i++) {
        buffer[i] = password.charAt(23-i);
    }
    for (; i<32; i+=2) {
        buffer[i] = password.charAt(46-i);
    }
    for (i=31; i>=17; i-=2) {
        buffer[i] = password.charAt(i);
    }
    String s = new String(buffer);
    return s.equals("jU5t_a_sna_3lpm18g947_u_4_m9r54f");
}
```

Step2. The flag seemed to be from picoCTF{jU5t_a_sna_3lpm18g947_u_4_m9r54f}. However, the string for the flag should be reversed with the looping process. So, the program of the reverse process for the original string and result are as below. The flag is picoCTF{jU5t_a_s1mpl3_an4gr4m_4_u_79958f}



```java
public class HelloWorld{

     public static void main(String []args){
        String password = "jU5t_a_sna_3lpm18g947_u_4_m9r54f";

        char[] buffer = new char[32];
        int i;
        for (i=0; i<8; i++) {
            buffer[i] = password.charAt(i);
        }
        for (; i<16; i++) {
            buffer[i] = password.charAt(23-i);
        }
        for (; i<32; i+=2) {
            buffer[i] = password.charAt(46-i);
        }
        for (i=31; i>=17; i-=2) {
            buffer[i] = password.charAt(i);
        }
        System.out.println(buffer);
     }
}
```

```
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
jU5t_a_s1mpl3_an4gr4m_4_u_79958f
```

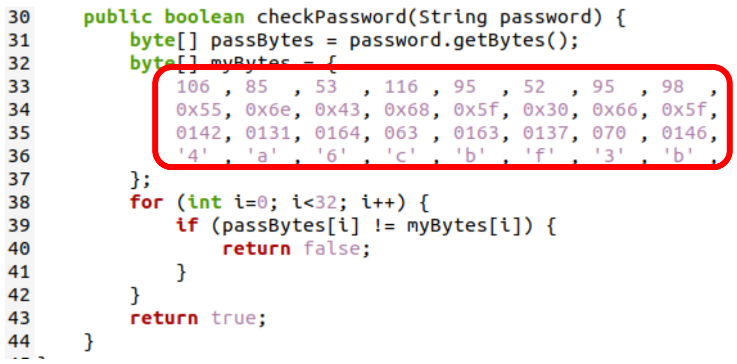
Java IDE online site: https://www.tutorialspoint.com/compile_java_online.php

## 3.2 vault-door4

. **Flag**: picoCTF{jU5t_4_bUnCh_0f_bYt3s_8f4a6cbf3b}

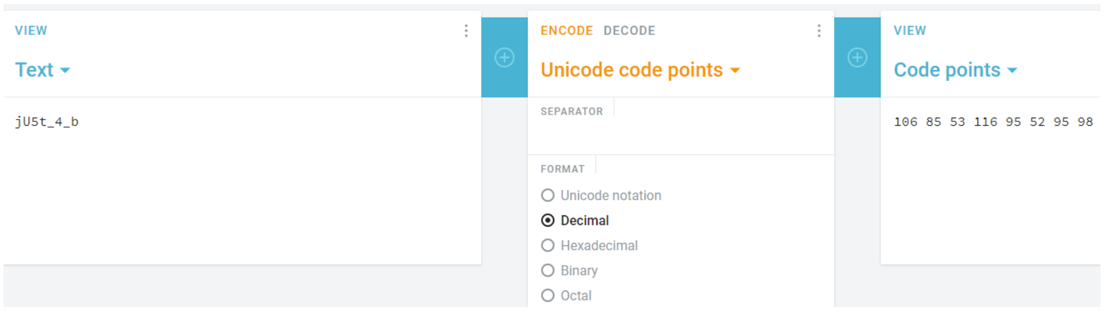
. **Approach**:

Step1. main function in VaultDoor4 java program is the same as that of vault-door3. The different part is checkPassword function. It is about converting from decimal, hexadecimal and octal. The last line is normal text.



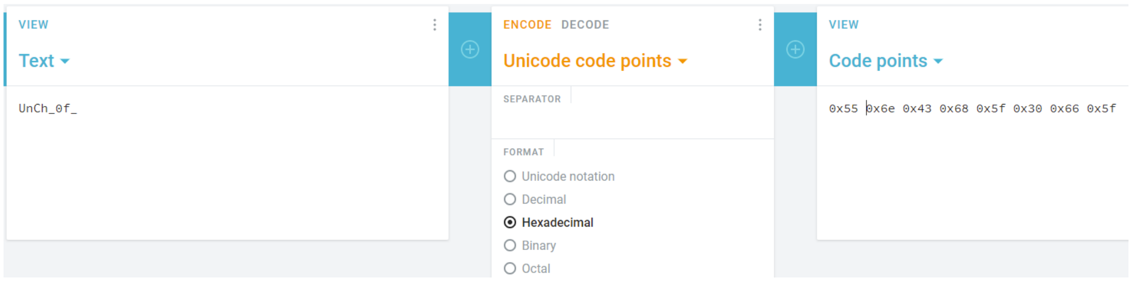
```
    public boolean checkPassword(String password) {
        byte[] passBytes = password.getBytes();
        byte[] myBytes = {
            106 , 85  , 53  , 116 , 95  , 52  , 95  , 98  ,
            0x55, 0x6e, 0x43, 0x68, 0x5f, 0x30, 0x66, 0x5f,
            0142, 0131, 0164, 063 , 0163, 0137, 070 , 0146,
            '4' , 'a' , '6' , 'c' , 'b' , 'f' , '3' , 'b' ,
        };
        for (int i=0; i<32; i++) {
            if (passBytes[i] != myBytes[i]) {
                return false;
            }
        }
        return true;
    }
```

Step2. The first line should be converted from decimal to text. The second line and the third line should take the similar process such as hexadecimal to text, octal to text. The strings are jU5t_4_b from the first line, UnCh_0f_ from the second line and bYt3s_8f from the third line.



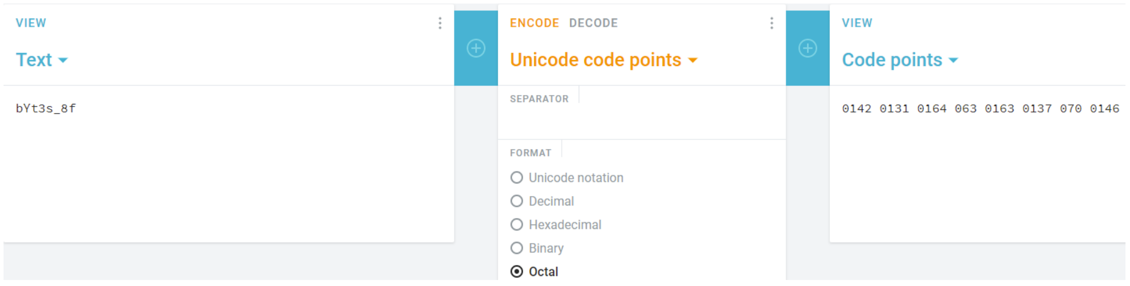
VIEW
Text
```
jU5t_4_b
```
ENCODE DECODE
Unicode code points
SEPARATOR
FORMAT
Unicode notation [✓ Decimal] Hexadecimal Binary Octal

VIEW
Code points
```
106 85 53 116 95 52 95 98
```



VIEW
Text
```
UnCh_0f_
```
ENCODE DECODE
Unicode code points
SEPARATOR
FORMAT
Unicode notation Decimal [✓ Hexadecimal] Binary Octal

VIEW
Code points
```
0x55 0x6e 0x43 0x68 0x5f 0x30 0x66 0x5f
```



VIEW
Text
```
bYt3s_8f
```
ENCODE DECODE
Unicode code points
SEPARATOR
FORMAT
Unicode notation Decimal Hexadecimal Binary [✓ Octal]

VIEW
Code points
```
0142 0131 0164 063 0163 0137 070 0146
```

Reference site: https://cryptii.com/pipes/text-octal

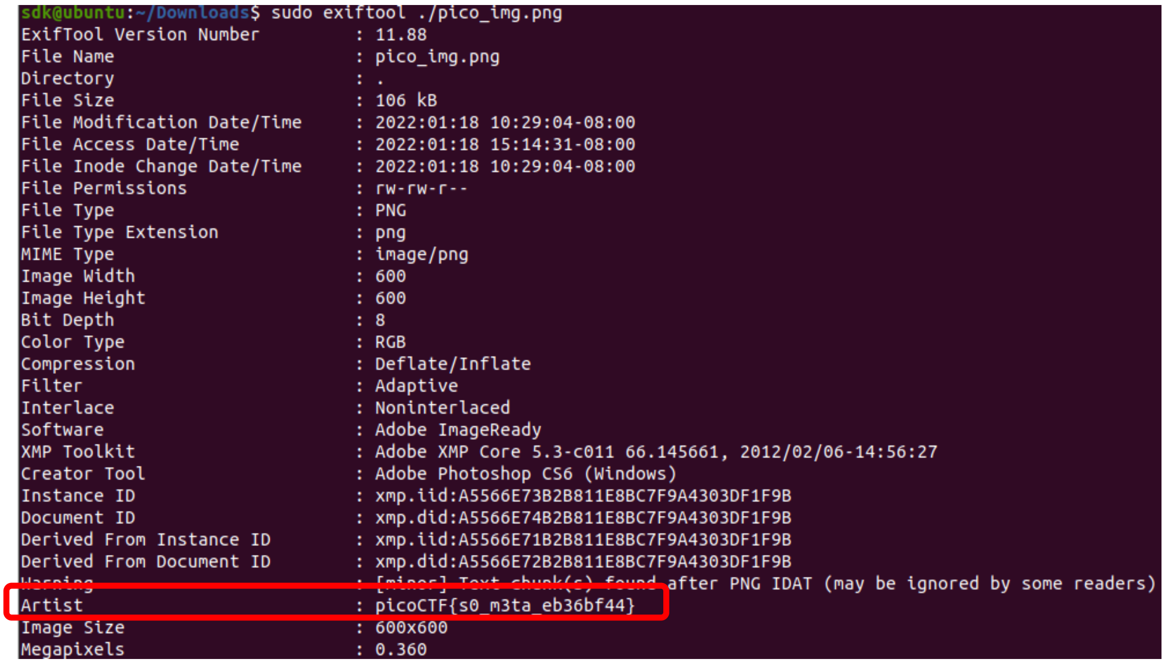
## 4. Forensics

### 4.1 So Meta

. **Flag**: picoCTF{s0_m3ta_eb36bf44}

. **Approach**:

Step1. There is no clue in the png image. The hint on the site is “What does meta mean in the context of files?” and “Ever heard of metadata?” So, I installed Exiftool on Ubuntu to read metadata of the file.



```
sdk@ubuntu:~/Downloads$ sudo exiftool ./pico_img.png
ExifTool Version Number         : 11.88
File Name                       : pico_img.png
Directory                       : .
File Size                       : 106 kB
File Modification Date/Time     : 2022:01:18 10:29:04-08:00
File Access Date/Time           : 2022:01:18 15:14:31-08:00
File Inode Change Date/Time     : 2022:01:18 10:29:04-08:00
File Permissions                : rw-rw-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 chunk(s) found after PNG IDAT (may be ignored by some readers)
Artist                          : picoCTF{s0_m3ta_eb36bf44}
Image Size                      : 600x600
Megapixels                      : 0.360
```

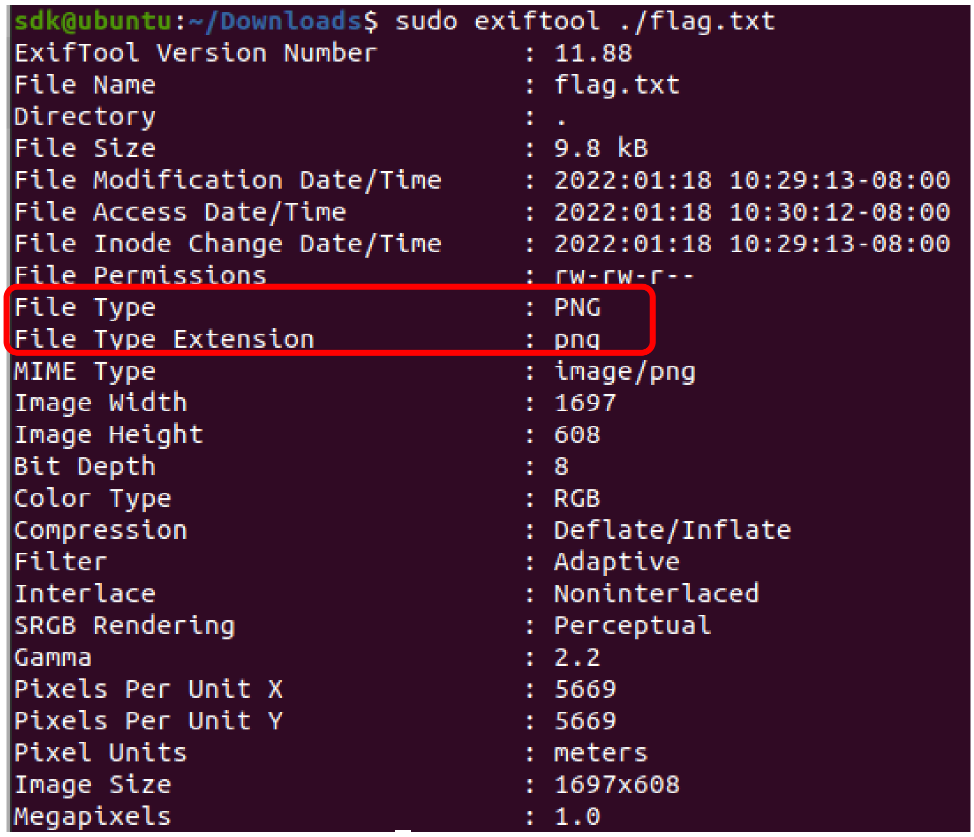
Step2. The flag is in the metadata as Artist information.

### 4.2 extensions

. **Flag**: picoCTF{now_you_know_about_extensions}

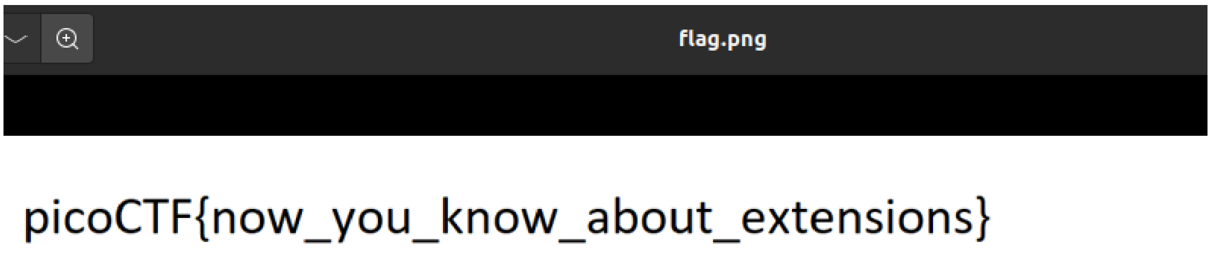
. **Approach**:

Step1. There is no clue from the file. But the hint on the site is “How do operating systems know what kind of file it is?”, Operating systems know what kind of file it is by extensions. I used to Exiftool to read the metadata of the file, and found that the file type and file type extension should be png.



```
sdk@ubuntu:~/Downloads$ sudo exiftool ./flag.txt
ExifTool Version Number         : 11.88
File Name                       : flag.txt
Directory                       : .
File Size                       : 9.8 kB
File Modification Date/Time     : 2022:01:18 10:29:13-08:00
File Access Date/Time           : 2022:01:18 10:30:12-08:00
File Inode Change Date/Time     : 2022:01:18 10:29:13-08:00
File Permissions                : rw-rw-r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1697
Image Height                    : 608
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Gamma                           : 2.2
Pixels Per Unit X               : 5669
Pixels Per Unit Y               : 5669
Pixel Units                     : meters
Image Size                      : 1697x608
Megapixels                      : 1.0
```

Step2. I found the flag after changing the file extension to png.



picoCTF{now_you_know_about_extensions}