

Writeup PicoCTF

Category Forensic



Created by

rainxmei

pens

Daftar Isi

[EASY].....	2
Riddle Registry	2
Hidden in plain sight	3



[EASY]

Riddle Registry

Hi, intrepid investigator! 📄🔍 You've stumbled upon a peculiar PDF filled with what seems like nothing more than garbled nonsense. But beware! Not everything is as it appears. Amidst the chaos lies a hidden treasure—an elusive flag waiting to be uncovered. Find the PDF file here [Hidden Confidential Document](#) and uncover the flag within the metadata.

Attachment: [confidential.pdf](#)

Kita diberikan sebuah file pdf, pertama cek dulu metadata file:

```
(kali㉿kali)-[~/pico_fore/riddle_rergency]
$ exiftool confidential.pdf
ExifTool Version Number : 13.25
File Name : confidential.pdf
Directory : .
File Size : 183 kB
File Modification Date/Time : 2025:11:24 12:52:59-05:00
File Access Date/Time : 2025:12:11 11:01:07-05:00
File Inode Change Date/Time : 2025:11:24 12:55:40-05:00
File Permissions : -rwxrwx---
File Type : PDF
File Type Extension : pdf
MIME Type : application/pdf
PDF Version : 1.7
Linearized : No
Page Count : 1
Producer : PyPDF2
Author : cGljb0NURntwdXp6bDNkX20zdGFkYXRhX2YwdW5kIV9jOGY5MWQ2OH0=
```

Wah ternyata ada metadata yang mencurigakan yaitu "Author" yang menggunakan enkripsi base64, coba kita decode text tersebut:

```
(kali㉿kali)-[~/pico_fore/riddle_rergency]
$ echo "cGljb0NURntwdXp6bDNkX20zdGFkYXRhX2YwdW5kIV9jOGY5MWQ2OH0=" | base64
-d
picoCTF{puzzl3d_m3tadata_f0und!_c8f91d68}
```

Yeayyy ternyata text tersebut berisi flag.

Flag : picoCTF{puzzl3d_m3tadata_f0und!_c8f91d68}

Hidden in plain sight

You're given a seemingly ordinary JPG image. Something is tucked away out of sight inside the file. Your task is to discover the hidden payload and extract the flag. Download the jpg image [here](#).

Attachment: [img.jpg](#)

Kita diberikan file img.jpg, pertama kita cek dulu metadata file:

```
(kali㉿kali)-[~/pico_fore/hidden_in_plainsight]
└─$ exiftool img.jpg
ExifTool Version Number      : 13.25
File Name                   : img.jpg
Directory                   :
File Size                    : 74 kB
File Modification Date/Time : 2025:11:24 13:49:53-05:00
File Access Date/Time       : 2025:12:11 11:31:43-05:00
File Inode Change Date/Time: 2025:11:24 13:51:07-05:00
File Permissions            : -rwxrwx—
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.01
Resolution Unit             : None
X Resolution                : 1
Y Resolution                : 1
Comment                     : c3RlZ2hpZGU6Y0VGNmVuZHjzbVE9
Image Width                 : 640
Image Height                : 640
Encoding Process            : Baseline DCT, Huffman coding
Bits Per Sample             : 8
Color Components            : 3
Y Cb Cr Sub Sampling       : YCbCr4:2:0 (2 2)
Image Size                  : 640×640
Megapixels                  : 0.410
```

Nah terlihat ada yang mencurigakan dibagian "Comment", kita decode isi dari "Comment" dulu menggunakan base64.

```
(kali㉿kali)-[~/pico_fore/hidden_in_plainsight]
└─$ echo "c3RlZ2hpZGU6Y0VGNmVuZHjzbVE9" |base64 -d
steghide:cEF6endvcmQ=
```



```
(kali㉿kali)-[~/pico_fore/hidden_in_plainsight]
└─$ echo "cEF6endvcmQ=" |base64 -d
pAzzword
```

Ternyata isinya adalah sebuah clue "steghide" dan sebuah password, langsung saja gunakan tools steghide.

```
(kali㉿kali)-[~/pico_fore/hidden_in_plainsight]
└─$ steghide extract -sf img.jpg -p pAzzword
wrote extracted data to "flag.txt".
```



```
(kali㉿kali)-[~/pico_fore/hidden_in_plainsight]
└─$ cat flag.txt
picoCTF{h1dd3n_1n_1m4g3_92f08d7c}
```

Flag : picoCTF{h1dd3n_1n_1m4g3_92f08d7c}

Flag in Flame

The SOC team discovered a suspiciously large log file after a recent breach. When they opened it, they found an enormous block of encoded text instead of typical logs. Could there be something hidden within? Your mission is to inspect the resulting file and reveal the real purpose of it. The team is relying on your skills to uncover any concealed information within this unusual log. Download the encoded data here: Logs Data. Be prepared—the file is large, and examining it thoroughly is crucial.

Attachment: [encodeFile.png](#)

Kita diberikan file "encodeFile.png", coba kita buka file tersebut berisi gambar apa.



Wahh, ada ternyata ada text yang mencurigakan, sepertinya itu adalah enkripsi hex. Langsung saja kita coba decode text tersebut.

```
└─(kali㉿kali)-[~/pico_fore/flag_in_flame]
└$ echo "7069636F4354467B666F72656E736963735F616E616C797369735F69735F616D61
7A696E675F37383265353563397D" |xxd -r -p
picoCTF{forensics_analysis_is_amazing_782e55c9}
```

Yeayy, sudah ketemu flagnya.

Flag : picoCTF{forensics_analysis_is_amazing_782e55c9}



Corrupted file

This file seems broken... or is it? Maybe a couple of bytes could make all the difference. Can you figure out how to bring it back to life?

Attachment: file

Kita diberikan sebuah file, kita coba cek dulu itu file apa

```
➜ ~/abyan/pico-ctf/corrupted-file ➜ 07:22:45 ⓘ
> open file
gio: file:///home/kali/abyan/pico-ctf/corrupted-file/file: Failed to find default application
for content type 'application/octet-stream'
```

Sepertinya ada kesalahan pada file tersebut, coba kita cek magic number file tersebut

```
➜ ~/abyan/pico-ctf/corrupted-file ➜ 07:23:05 ⓘ
> xxd file
00000000: 5c78 ffe0 0010 4a46 4946 0001 0100 0001 \x....JFIF.....
00000010: 0001 0000 ffdb 0043 0008 0606 0706 0508 .....C.....
00000020: 0707 0709 0908 0a0c 140d 0c0b 0b0c 1912 .....
00000030: 130f 141d 1a1f 1e1d 1a1c 1c20 242e 2720 .....$.'.
00000040: 222c 231c 1c28 3729 2c30 3134 3434 1f27 ",#..(7),01444.'.
00000050: 393d 3832 3c2e 3334 32ff db00 4301 0909 9=82<.342...C...
```

Nah kan, seharusnya magic number file jpeg bukan seperti itu, coba kita ubah magic number nya dengan hexedit, setelah diganti coba kita buka file tersebut

picoCTF{r3st0r1ng_th3_by73s_2326ca93}

Yeayy, ketemu flagnya

picoCTF{r3st0r1ng_th3_by73s_2326ca93}



Disko 1

Can you find the flag in this disk image? Download the disk image

Attachment: disko-1.dd.gz

Kita diberikan file disk image, coba kita extract dulu file nya dengan menggunakan command seperti ini

Gunzip disko-1.dd.gz

Kemudian coba kita cek file tersebut menggunakan strings

```
➜  ~/abyan/pico-ctf/disko-1 07:51:19
) strings disko-1.dd | grep "pico"
:/icons/appicon
# $Id: piconv,v 2.8 2016/08/04 03:15:58 dankogai Exp $
piconv -- iconv(1), reinvented in perl
  piconv [-f from_encoding] [-t to_encoding]
  piconv -l
  piconv -r encoding_alias
  piconv -h
B<piconv> is perl version of B<iconv>, a character encoding converter
a technology demonstrator for Perl 5.8.0, but you can use piconv in the
piconv converts the character encoding of either STDIN or files
Therefore, when both -f and -t are omitted, B<piconv> just acts
picoCTF{1t5_ju5t_4_5tr1n9_e3408eef}
```

Nah ketemu flagnya

picoCTF{1t5_ju5t_4_5tr1n9_e3408eef}



RED
RED, RED, RED, RED
Attachment: red.png

Kita diberikan file png, kita coba cek terlebih dahulu menggunakan strings

```
08:00:12
~/abyan/pico-ctf/red
strings red.png
IHDR
tEXtPoem
Crimson heart, vibrant and bold,
Hearts flutter at your sight.
Evenings glow softly red,
Cherries burst with sweet life.
Kisses linger with your warmth.
Love deep as merlot.
Scarlet leaves falling softly,
Bold in every stroke.x
IDATx
IEND
```

C, H, E, C, K, L, S, B.

Wow ketika kita lihat pada huruf pertama puisi tersebut ternyata memberikan kita sebuah petunjuk, untuk mengecek lsb kita coba pakai zsteg

```
08:08:02
~/abyan/pico-ctf/red
zsteg red.png
meta Poem .. text: "Crimson heart, vibrant and bold,\nHearts flutter at your sight.\nEvenings glow softly red,\nCherries burst with sweet life.\nKisses linger with your warmth.\nLove deep as merlot.\nScarlet leaves falling softly,\nBold in every stroke."
b1,rgba,lsb,xy .. text: "cGljb@NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ==cGljb@NURntyM2RfMXNfdGgzX3Vs
dDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ==cGljb@NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zN
TVffQ=="
b1,rgba,msb,xy .. file: OpenPGP Public Key
b2,g,lsb,xy .. text: "ET@UETPETUUT@TUUTD@PDUDDDPE"
b2,rgb,lsb,xy .. file: OpenPGP Secret Key
b2,bgr,msb,xy .. file: OpenPGP Public Key
b2,rgba,lsb,xy .. file: OpenPGP Secret Key
b2,rgba,msb,xy .. text: "CIkiiiII"
b2,abgr,lsb,xy .. file: OpenPGP Secret Key
b2,abgr,msb,xy .. text: "iiiaakikk"
b3,rgba,msb,xy .. text: "#wb#wp#7p"
b3,abgr,msb,xy .. text: "7r'wb#7p"
b4,b,lsb,xy .. file: 0421 Alliant compact executable not stripped
```

Setelah kita check ternyata ada text yang dienkripsi dengan base64

```
08:08:13
~/abyan/pico-ctf/red
echo "cGljb@NURntyM2RfMXNfdGgzX3VsdDFtNHQzX2N1cjNfZjByXzU0ZG4zNTVffQ==" | base64 -d
picoCTF{r3d_1s_th3_ult1m4t3_c0r3_f0r_54dn355_}%
```

Horee kita dapat flagnya

picoCTF{r3d_1s_th3_ult1m4t3_c0r3_f0r_54dn355_}

Phantom Intruder

A digital ghost has breached my defenses, and my sensitive data has been stolen! 🤡💻 Your mission is to uncover how this phantom intruder infiltrated my system and retrieve the hidden flag. To solve this challenge, you'll need to analyze the provided PCAP file and track down the attack method. The attacker has cleverly concealed his moves in well timely manner. Dive into the network traffic, apply the right filters and show off your forensic prowess and unmask the digital intruder! Find the PCAP file here [Network Traffic PCAP file](#) and try to get the flag.

Attachment: myNetworkTraffic.pcap

kita diberi file traffic, coba kita buka diwireshark

192.168.1.2	TCP	52 20 → 80 [SYN] Seq=0 Win=8192 Len=12
192.168.1.2	TCP	52 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=12
192.168.1.2	TCP	52 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=12
192.168.1.2	TCP	52 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=12
192.168.1.2	TCP	52 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=12
192.168.1.2	TCP	52 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=12

Coba kita lihat data di len=12

```
bnRfdGg0dA==
```

Wah ternyata isinya enkripsi base64, coba kita lihat semua yang len=12, kemudian susun semua potongan flag tersebut

picoCTF{1t_w4snt_th4t_34sy_tbh_4r_36f4a666}



Verify

People keep trying to trick my players with imitation flags. I want to make sure they get the real thing! I'm going to provide the SHA-256 hash and a decrypt script to help you know that my flags are legitimate.ssh -p 61536 ctf-player@rhea.picoctf.netUsing the password 6dd28e9b. Accept the fingerprint with yes, and ls once connected to begin. Remember, in a shell, passwords are hidden!

- Checksum:
03b52eabed517324828b9e09cbbf8a7b0911f348f76cf989ba6d51acede6d5d8
- To decrypt the file once you've verified the hash, run ./decrypt.sh files/<file>.

Attachment:

Kita disuruh untuk menggunakan ssh dan kita juga diberi passwordnya, langsung saja kita masuk

```
ctf-player@pico-chall$ ls
checksum.txt  decrypt.sh  files
```

Coba kita lihat isi dari direktori files

```
ctf-player@pico-chall$ ls
00011a60 63MqIIVV CLCyTz85 IlQwVZcY QbiQCWzR Ww6oTYL8 eZ4ehccg lxv6mvZ6 sAy34VP4
022cvpdN 64nJLBv CNvsyU3W ItYR0Da2 QcIkjhJ0 X5rRZ32p efpCZmHN m1NnTZoo sKi8TaSn
04nLilRD 69JSHBh1 Cb22Z7F0 Izq2bmb5 QgmCuMSV XGTpUJIw ekSs7xW4 m3bsNhyN sNI2Q6oa
0MT2Wrui 6vgioqew Ce5TrzJu J16J63tC Qxrlj2uk XJiFKTlc eknrQKQh mWWBZCgt sRaKyq1f
0SGMttrR 76rj6cv7 ClWGbsxu JVEoV1Bn R3rVsQa8 XcmGmkwD fZHENAvZ mdFDpW9k svmpIxV
0fcDySFB 7j2g9w9w DCn7KnqG Jcwq4RxP RAXeLvjl YAZlvEou g2E6RkkX n2XnM9Nc tFGQywrr
0hHVJSPh 7ye3lPVb DSKHZ66z Jk8UBmcS RFLWtody YKQLrxBm g2nu6vlR n7Vs8Bjh tY2epsSy
0mPyFlfa 870YaC5g Dmsex2Ug KCL5hW02 RHjAw3hj YdTzkUcM gLAo3J0D n8r2Ejk9 tuma818A
0xknvebh 8Shyigig EBBQm7M KKRWbrqC RPVIP1xx Z2rLxuyp gem657x8 nld0sSfJ uMpXxbqr
1kTWMoOI 8d0Ncqme EG1lW2KR KUIfl2m7 RQWaIGxG ZWIiY84t hnqxyUX2 nnZ33FAt uUI8gJNi
1mG1W6Ts 8hKIVq38 EXQ6Di05 KbGMgDus RVejZvvP ZXqAvkEc hZxbAqts nrwKQbjk ugeJ5RN3
2JKdkggW 8rIuGenM F6yHlWpt KlqDh1ZQ RdYwRe68 ZdPbKJh1 haNCaZmC oNb9jru vJDrHtxo
2Jr8UtbZ 91cL0GeN FJBePm2b L58tVhf Rgs7l9CZ ZyNsHVFw iGwCDzaU o196tAtc vMv1M1qs
2K4XCmfE 9DNFzhUK FLsBEmlR L7gltLCF Sc0tAOiz aGVRRt1d iILvZZya oiy29oCW vWguQ8rQ
2MYWkWLC 9KIFXofB F0xKdaVP LCLocE1C SwrcVnay aHFaEXkf ih6levXk p1LgEqdu vc1wGQhn
2QpRnoZQ 9pluLfgA FtMorZ65 LMavH6JA T5IkmdqtJ aIz8E0Iy j1v0LBVe p5INCxLV vjypfsoh
2emuPV0b ADMuzktV G7enzzui Lq3dNaLV T6AHhqdE afLk75a0 j1lhVDLw p5INQHq8 vsGKdf0J
2gP5wDgq AEJxVLNY Gcv1H8Qs LrYo1dnv TPBDRCiJ b9Ycg3Tz j0bn0z94 pXJHJUbH w1XGgnr9
2w5vJLLG AG0EyD4N GhrShrXN MPes8YHI TRyxUzw bDK7A26M jVkxEmtq pb1E0Y3Z w8DmFhfg
2yMtx5qd AVdbk5eX Gtk4Kn9w N8vFO6DF TXsLzqsp bDZN0f4B jVlaDg4q pnycz11G woaiQu5g
303DzMmf AXFWLqwI GufDk3Mb NC6PZddL TeaXjOeh bcZupFpi jcMzi4V0 q53EoTzu wvhWmTPt
33CFCJ0y AdzCNBLc HDLWGApz NS9xPzIA TeyHF78l bjtBjwTc jdYv9CQ3 qCTrc9yM xQJV5GcG
36tjtTw0F AiUxYmz8 HiEYL84k Ndyi6bnx ToT9QPKf c3Z3JN0m jzmPa02D qHwcKaSC xlqX0qhL
3KZwXc8s AmsN0Lkj HJIPzwjJ NxdIqu0S TtPblPd6 cIDWC9cb KDPV8ASY qK35XLHM yACAAkqG
3Vs8v8kW ArUDDIQ3 HMq6348V 05tEUfhw U3BoYTr9 dDoFZTxh KKVVpY8S qSn3WAyi yYRsKiU0
3qDKN57P Azqf6EEw HUjCgnh4 OH3906gp UDI6pN8S dKYP6pnk KWjYWILD qV83Dmye yg7uBent
4CwloraZ B8pBCEvG HWRVc59e OIYZeUCB UF1urDfG dKissxYdK kZ6DTcql qWv24Da7 zYz6howf
4Xqpqs6B BN0HxLxE Hmr54gXd OPqDb0IH UUiDNDLO deppMJSV kbumrMcy qZ7TLGA0 zjkul95p
```

Wow banyak banget ternyata, kalo kita decrypt satu-satu akan memakan waktu yang banyak, kita buat script agar lebih cepat, begini scriptnya

```
ctf-player@pico-chall$ cat > script.sh << 'EOF'
>#!/bin/bash
> for file in files/*; do
>     hasil=$(./decrypt.sh "$file") # perlu ./ sebelum decrypt.sh
>     echo "$hasil"
> done
> EOF
ctf-player@pico-chall$ chmod 777 script.sh
ctf-player@pico-chall$ ./script.sh
picoCTF{trust_but_verify_00011a60}
bad magic number
Error: Failed to decrypt 'files/022cvpdN'. This flag is fake! Keep looking!
bad magic number
Error: Failed to decrypt 'files/04nLilRD'. This flag is fake! Keep looking!
bad magic number
```

Setelah itu jangan lupa kasih permission ke script nya dan jalankan script tersebut, yeayy dapat flagnya

picoCTF{trust_but_verify_00011a60}

pens

Secret of the Polyglot

The Network Operations Center (NOC) of your local institution picked up a suspicious file, they're getting conflicting information on what type of file it is. They've brought you in as an external expert to examine the file. Can you extract all the information from this strange file? Download the suspicious file [here](#).

Attachment: flag2of2-final.pdf

Jadi kita di berikan file pdf, langsung saja kita lihat isinya apa

```
1n_pn9_&_pdf_1f991f77}
```

Itu adalah potongan dari flagnya, flag tersebut memberikan petunjuk kita "in png & pdf", coba kita cek menggunakan seperti berikut

```
~/abyan/pico-ctf/secret-of-the-polygot 13:04:26
> file flag2of2-final.pdf
flag2of2-final.pdf: PNG image data, 50 x 50, 8-bit/color RGBA, non-interlaced
```

Pas dicek jadi png, hmmm mencurigakan coba kita ganti extensionnya menjadi png

```
~/abyan/pico-ctf/secret-of-the-polygot 13:04:30
> mv flag2of2-final.pdf flag2of2-final.png
```

Kemudian kita buka apa isi dari file tersebut



Yeayy kita dapat potongan selanjutnya

```
picoCTF{f1u3n7_1n_pn9_&_pdf_1f991f77}
```

CanYouSee

How about some hide and seek? Download this file [here](#).

Attachment: ukn_reality.jpg

Kita diberi file jpg, coba kita lihat metadatanya

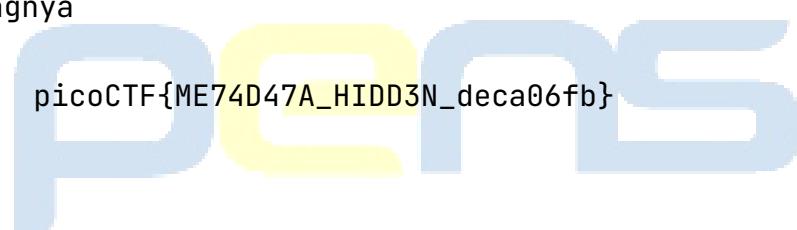
```
~/abyan/pico-ctf/canyousee 20:30:54
❯ exiftool ukn_reality.jpg
ExifTool Version Number      : 13.36
File Name                   : ukn_reality.jpg
Directory                   : .
File Size                   : 2.3 MB
File Modification Date/Time : 2024:02:15 17:40:14-05:00
File Access Date/Time       : 2025:12:21 20:30:54-05:00
File Inode Change Date/Time: 2025:12:21 20:30:42-05:00
File Permissions            : -rw-r--r--
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.01
Resolution Unit              : inches
X Resolution                 : 72
Y Resolution                 : 72
XMP Toolkit                  : Image::ExifTool 11.88
Attribution URL             : cG1jb0NURntNRTc0RDQ3QV9ISUREM05fZGVjYTA2ZmJ9Cg==
Image Width                  : 4308
Image Height                 : 2875
Encoding Process             : Baseline DCT, Huffman coding
Bits Per Sample              : 8
Color Components              : 3
YCbCr Sub Sampling          : YCbCr4:2:0 (2 2)
Image Size                   : 4308x2875
Megapixels                   : 12.4
```

Wow ada enkripsi base64, coba kita decode

```
~/abyan/pico-ctf/canyousee 20:31:18
❯ echo "cG1jb0NURntNRTc0RDQ3QV9ISUREM05fZGVjYTA2ZmJ9Cg==" | base64 -d
picoCTF{ME74D47A_HIDD3N_deca06fb}
```

Horee ketemu flagnya

picoCTF{ME74D47A_HIDD3N_deca06fb}



Information

Files can always be changed in a secret way. Can you find the

Attachment: cat.jpg

Kita diberikan file jpg, coba kita lihat metadatanya

```
~$ exiftool cat.jpg
ExifTool Version Number      : 13.36
File Name                   : cat.jpg
Directory                  : .
File Size                   : 878 kB
File Modification Date/Time : 2025:12:21 20:38:03-05:00
File Access Date/Time       : 2025:12:21 20:38:00-05:00
File Inode Change Date/Time: 2025:12:21 20:41:25-05:00
File Permissions            : -rw-rw-r--
File Type                   : JPEG
File Type Extension         : jpg
MIME Type                   : image/jpeg
JFIF Version                : 1.02
Resolution Unit              : None
X Resolution                 : 1
Y Resolution                 : 1
Current IPTC Digest        : 7a78f3d9cfb1ce42ab5a3aa30573d617
Copyright Notice             : PicoCTF
Application Record Version   : 4
XMP Toolkit                  : Image::ExifTool 10.80
License                      : cGljb0NURnt0aGVfbTN0YWRhdGFFMXNfbW9kaWZpZWR9
Rights                        : PicoCTF
Image Width                  : 2560
Image Height                 : 1598
Encoding Process              : Baseline DCT, Huffman coding
Bits Per Sample               : 8
Color Components              : 3
Y Cb Cr Sub Sampling         : YCbCr4:2:0 (2 2)
Image Size                   : 2560x1598
Megapixels                     : 4.1
```

Wahh ada text yang dienkripsi menggunakan base64 coba kita decode

```
~$ echo "cGljb0NURnt0aGVfbTN0YWRhdGFFMXNfbW9kaWZpZWR9" | base64 -d
picoCTF{the_m3tadata_1s_modified}
```

picoCTF{the_m3tadata_1s_modified}

Glory of the Garden

This file contains more than it seems. Get the flag from [garden.jpg](#).

Attachment: garden.jpg

kita diberikan file jpg, seperti biasa kita coba lihat metadatanya terlebih dahulu

```
~/abyan/pico-ctf/forensic/easy/glory-of-the-garden 20:55:05
❯ exiftool garden.jpg
ExifTool Version Number : 13.36
File Name : garden.jpg
Directory : .
File Size : 2.3 MB
File Modification Date/Time : 2025:12:21 20:48:50-05:00
File Access Date/Time : 2025:12:21 20:49:45-05:00
File Inode Change Date/Time : 2025:12:21 20:49:39-05:00
File Permissions : -rw-rw-r--
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
JFIF Version : 1.01
Resolution Unit : inches
X Resolution : 72
Y Resolution : 72
Profile CMM Type : Linotronic
Profile Version : 2.1.0
Profile Class : Display Device Profile
Color Space Data : RGB
Profile Connection Space : XYZ
Profile Date Time : 1998:02:09 06:49:00
Profile File Signature : acsp
Primary Platform : Microsoft Corporation
CMM Flags : Not Embedded, Independent
Device Manufacturer : Hewlett-Packard
Device Model : sRGB
Device Attributes : Reflective, Glossy, Positive, Color
Rendering Intent : Perceptual
Connection Space Illuminant : 0.9642 1 0.82491
Profile Creator : Hewlett-Packard
Profile ID : 0
```

Hmm sepertinya tidak ada yang aneh, coba kita lihat menggunakan strings, begini command-nya

```
~/abyan/pico-ctf/forensic/easy/glory-of-the-garden 20:55:12
❯ strings garden.jpg | grep "pico"
Here is a flag: picoCTF{more_than_m33ts_the_3y339cbe6dc}
```

Nah kan, yeayy kita dapat flagnya

picoCTF{more_than_m33ts_the_3y339cbe6dc}

[MEDIUM]

Disko 2

Can you find the flag in this disk image? The right one is Linux!
One wrong step and its all gone! Download the disk image

Attachment: disko-2.dd

Kita diberi file disk image dan kita juga dikasih clue untuk ekstrak yang bagian linux pada disk tersebut, kita lihat dulu yang partisi linux dimana

```
~$ ls ~/abyan/pico-ctf/forensic/medium/disko-2
mmls disk0-2.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot Start End Length Description
000: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
001: ----- 0000000000 0000002047 0000002048 Unallocated
002: 000:000 0000002048 0000053247 0000051200 Linux (0x83)
003: 000:001 0000053248 0000118783 0000065536 Win95 FAT32 (0x0b)
004: ----- 0000118784 0000204799 0000086016 Unallocated
```

Kita bisa lihat bahwa disk image tersebut memiliki 512byte sectors dan partisi linux di 2048, langsung saja kita extract

```
~$ dd if=disk0-2.dd bs=512 skip=2048 count=51200 of=partlinux.img
51200+0 records in
51200+0 records out
26214400 bytes (26 MB, 25 MiB) copied, 0.0775536 s, 338 MB/s
```

Kemudian kita coba lihat menggunakan strings

```
~$ strings partlinux.img | grep "pico"
picoCTF{4_P4Rt_1t_i5_055dd175}
piconv
:/icons/appicon
piconv
piconv
piconv
# $Id: piconv,v 2.8 2016/08/04 03:15:58 dankogai Exp $
piconv -- iconv(1), reinvented in perl
  piconv [-f from_encoding] [-t to_encoding]
  piconv -l
```

Yeayy kita dapat flagnya

picoCTF{4_P4Rt_1t_i5_055dd175}

Disko 3

Can you find the flag in this disk image? This time, its not as plain as you think it is! Download the disk image.

Attachment: disko-3.dd

Kita diberikan disk image, coba kita terlebih dahulu

```
~$ fdisk -l disk0-3.dd
Disk disk0-3.dd: 100 MiB, 104857600 bytes, 204800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

Hmm tidak ada partisi apapun ternyata, coba kita check apakah ini benar disk image

```
~$ file disk0-3.dd
disk0-3.dd: DOS/MBR boot sector, code offset 0x58+2, OEM-ID "mkfs.fat", Media descriptor 0xf8
, sectors/track 32, heads 8, sectors 204800 (volumes > 32 MB), FAT (32 bit), sectors/FAT 1576
, serial number 0x49838d0b, unlabeled
```

Ternyata emang benar disk image berarti emang tidak ada partisi apapun, coba kita langsung mount saja, command nya seperti ini

```
~$ sudo mount -o ro,loop disk0-3.dd mnt/disk
```

Coba kita lihat-lihat ada apa didalamnya

```
~$ ls ~a/pico-ctf/forensic/medium/disk0-3/mnt/disk/log
alternatives.log      installer          vmware-network.4.log
alternatives.log.2.gz  journal           vmware-network.5.log
apt                  kern.log.3.gz       vmware-network.6.log
boot.log              kern.log.4.gz       vmware-network.7.log
boot.log.1             lastlog            vmware-network.8.log
boot.log.5             lightdm            vmware-network.log
boot.log.6             macchanger.log.4.gz   vmware-vmsvc-root.1.log
daemon.log            mysql              vmware-vmsvc-root.2.log
debug                private             vmware-vmsvc-root.3.log
dpkg.log.1            stunnel4           vmware-vmsvc-root.log
dpkg.log.2.gz          syslog.3.gz        vmware-vmtoolsd-root.log
dpkg.log.4.gz          syslog.4.gz        wtmp
dpkg.log.5.gz          sysstat            Xorg.0.log
faillog               vmware-network.1.log  Xorg.0.log.old
flag.gz               vmware-network.2.log
inetsim               vmware-network.3.log
```

Wah ternyata ada flag.gz, langsung saja kita buka isinya ada apa

```
~$ cat flag
Here is your flag
picoCTF{n3v3r_z1p_2_h1d3_7e0a17da}
```

Horee sudah ketemu flagnya

picoCTF{n3v3r_z1p_2_h1d3_7e0a17da}

Event

One of the employees at your company has their computer infected by malware! Turns out every time they try to switch on the computer, it shuts down right after they log in. The story given by the employee is as follows:

1. They installed software using an installer they downloaded online
2. They ran the installed software but it seemed to do nothing
3. Now every time they bootup and login to their computer, a black command prompt screen quickly opens and closes and their computer shuts down instantly.

See if you can find evidence for the each of these events and retrieve the flag (split into 3 pieces) from the correct logs!

Attachment: Windows_logs.evtx

Kita diberikan file evtx, pertama-tama kita dump file evtx tersebut Disini saya coba bikin script untuk dump evtx

Evtx_dump.py

```
import Evtx.Evtx as evtx
import Evtx.Views as e_views

def main():
    import argparse

    parser = argparse.ArgumentParser(
        description="Dump a binary EVT file into XML.")
    parser.add_argument("evt", type=str,
                       help="Path to the Windows EVT event log file")
    args = parser.parse_args()

    with evt.Evtx(args.evt) as log:
        print(e_views.XML_HEADER)
        print("<Events>")
        for record in log.records():
            print(record.xml())
        print("</Events>")

if __name__ == "__main__":
    main()
```

```
~/abyan/pico-ctf/forensic/medium/event-viewing 1m 49s 05:11:40
python3 ~/abyan/pico-ctf/script/evtx_dump.py Windows_Logs.evtx > Windows_Logs.xml
```

Karena flag dienkripsi menggunakan base64 dan isi dari Windows_logs.xml banyak sekali, jadi saya membuat script untuk mendekripsi text yang dienkripsi dengan base64

```
import base64
import string
import re

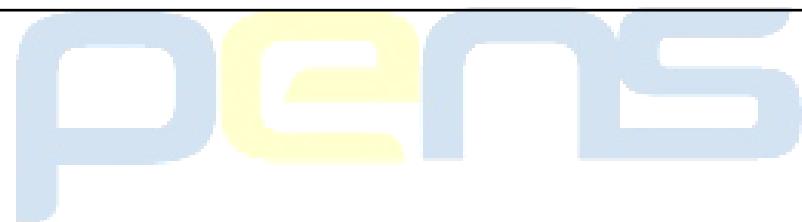
def is_printable(s):
    return all(c in string.printable for c in s)

def find_base64_in_text(text, min_length=8):
    base64_pattern = r'[A-Za-z0-9+/]{'+ str(min_length) + \
r',}{0,2}'
    candidates = re.findall(base64_pattern, text)
    print(f"[DEBUG] Total kandidat base64 ditemukan oleh \
regex: {len(candidates)}")

    found = []
    for candidate in candidates:
        try:
            decoded_bytes = base64.b64decode(candidate,
            validate=True)
            decoded_text = decoded_bytes.decode('utf-8')
            if is_printable(decoded_text):
                found.append((candidate, decoded_text))
        except Exception:
            continue

    return found

def main():
    input_file = 'Windows_Logs.xml'
```



```
output_file = 'hasil_deteksi.txt'

with open(input_file, 'r', encoding='utf-8') as f:
    text = f.read()

results = find_base64_in_text(text)

if results:
    with open(output_file, 'w', encoding='utf-8') as f:
        f.write(f"{len(results)} base64 string\n"
                "ditemukan:\n\n")
        for i, (encoded, decoded) in enumerate(
            results, 1):
            f.write(f"\n#{i}\nEncoded: {encoded}\n"
                    "\nDecoded: {decoded}\n\n")
    print(f"{len(results)} base64 string valid\n"
          "disimpan di '{output_file}'")
else:
    print("Tidak ditemukan base64 encoded string yang\n"
          "valid.")

if __name__ == '__main__':
    main()
```

Kita buka file hasil dari script base64_detector.py

picoCTF{Ev3nt_v13wv3r_1s_a_pr3tty_us3ful_t00l_81ba3fe9}



Bitlocker-1

Jacky is not very knowledgable about the best security passwords and used a simple password to encrypt their BitLocker drive. See if you can break through the encryption!

Attachment: bitlocker-1.dd

Kita diberi file disk image, dari hints kita disuruh untuk crack bitlocker tersebut, disini saya akan memakai bitlocker2john untuk mengkonversi ke format hash yang bisa dicrack

```
~/a/pico-ctf/script/john/run > bleeding-jumbo 08:12:16
> ./bitlocker2john -i ~/abyan/pico-ctf/forensic/medium/bitlocker-1/bitlocker-1.dd > ~/abyan/pico-ctf/forensic/medium/bitlocker-1/hash_bitlocker.txt

Signature found at 0x3
Version: 8
Invalid version, looking for a signature with valid version...

Signature found at 0x2195000
Version: 2 (Windows 7 or later)

VMK entry found at 0x21950c4

VMK encrypted with Recovery Password found at 0x21950e6
Searching AES-CCM from 0x2195102
Trying offset 0x2195195....
VMK encrypted with AES-CCM!

VMK entry found at 0x2195240

VMK encrypted with User Password found at 2195262
VMK encrypted with AES-CCM
```

Kemudian disebutkan pada soal bahwa password-nya itu simpel dan mudah berarti mungkin password tersebut ada di rockyou.txt, saya akan crack password menggunakan hashcat

```
~/a/pico-ctf/forensic/medium/bitlocker-1 > 46s 08:23:14
> hashcat -m 22100 hash_bitlocker.txt rockyou1.txt
hashcat (v7.1.2) starting

$bitlocker$0$16$cb4809fe9628471a411f8380e0f668db$1048576$12$d04d9c58eed6da010a000000$60$68156e
51e53f0a01c076a32ba2b2999afffce8530fbe5d84b4c19ac71f6c79375b87d40c2d871ed2b7b5559d71ba31b6779c
6f41412fd6869442d66d:jacqueline
```

Nah sudah ketemu password-nya adalah jacqueline, langsung saja kita coba apakah benar

```
~/abyan/pico-ctf/forensic/medium/bitlocker-1 > 08:28:52
> dislocker -ujacqueline bitlocker-1.dd hasil
~/abyan/pico-ctf/forensic/medium/bitlocker-1/hasil > 08:29:30
> strings dislocker-file | grep pico
picoCTF{us3_b3tt3r_p4ssw0rd5_pl5!_3242adb1}
```

Horee kita sudah dapat flagnya

picoCTF{us3_b3tt3r_p4ssw0rd5_pl5!_3242adb1}

Blast From the Past

The judge for these pictures is a real fan of antiques. Can you age this photo to the specifications? Set the timestamps on this picture to 1970:01:01 00:00:00.001+00:00 with as much precision as possible for each timestamp. In this example, +00:00 is a timezone adjustment. Any timezone is acceptable as long as the time is equivalent. As an example, this timestamp is acceptable as well: 1969:12:31 19:00:00.001-05:00. For timestamps without a timezone adjustment, put them in GMT time (+00:00). The checker program provides the timestamp needed for each.

Submit your modified picture here:

```
nc -w 2 mimas.picoctf.net 51847 < original_modified.jpg
```

Check your modified picture here:

```
nc mimas.picoctf.net 57306
```

Attachment: original.jpg

Kita diberikan file jpg, dari deskripsinya sepertinya kita disuruh untuk mengganti metadata time sesuai seperti yang diperintahkan, coba kita tes kirim file tersebut ke server kita lihat apa yang terjadi

```
Checking tag 1/7
Looking at IFD0: ModifyDate
Looking for '1970:01:01 00:00:00'
Found: 2023:11:20 15:46:23
Oops! That tag isn't right. Please try again.
```

Wow ada tujuh step pengecekan, untuk yang pertama kita disuruh mengubah metadata ModifyDate, langsung saja kita ubah menggunakan exiftool, begini caranya

```
~/.pico-ctf/forensic/medium/blast-from-the-past 11:45:35
> exiftool -ModifyDate="1970:01:01 00:00:00" original_modified.jpg
1 image files updated
```

Kita coba tes kirim filenya ke server, apakah sudah benar

```
Checking tag 1/7
Looking at IFD0: ModifyDate
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 2/7
Looking at ExifIFD: DateTimeOriginal
Looking for '1970:01:01 00:00:00'
Found: 2023:11:20 15:46:23
Oops! That tag isn't right. Please try again.
```

Untuk step kedua kita disuruh mengubah metadata DateTimeOriginal, begini caranya

```
~/.pico-ctf/forensic/medium/blast-from-the-past 11:48:30
> exiftool -DateTimeOriginal="1970:01:01 00:00:00" original_modified.jpg
1 image files updated
```

Kita tes kirim lagi ke server

```
Checking tag 1/7
Looking at IFD0: ModifyDate
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 2/7
Looking at ExifIFD: DateTimeOriginal
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 3/7
Looking at ExifIFD: CreateDate
Looking for '1970:01:01 00:00:00'
Found: 2023:11:20 15:46:23
Oops! That tag isn't right. Please try again.
```

Step ketiga kita disuruh mengubah metadata CreateDate, begini caranya

```
~/a/pico-ctf/forensic/medium/blast-from-the-past 11:50:19
> exiftool -CreateDate="1970:01:01 00:00:00" original_modified.jpg
1 image files updated
```

Kita tes kirim lagi ke server

```
Checking tag 1/7
Looking at IFD0: ModifyDate
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 2/7
Looking at ExifIFD: DateTimeOriginal
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 3/7
Looking at ExifIFD: CreateDate
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 4/7
Looking at Composite: SubSecCreateDate
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.703
Oops! That tag isn't right. Please try again.
```

Untuk step keempat kita disuruh mengubah metadata SubSecCreateDate, begini caranya

```
~/a/pico-ctf/forensic/medium/blast-from-the-past 11:52:06
> exiftool -SubSecCreateDate="1970:01:01 00:00:00.001" original_modified.jpg
1 image files updated
```

Kita tes kirim ke server

```
Checking tag 3/7
Looking at ExifIFD: CreateDate
Looking for '1970:01:01 00:00:00'
Found: 1970:01:01 00:00:00
Great job, you got that one!

Checking tag 4/7
Looking at Composite: SubSecCreateDate
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.001
Great job, you got that one!

Checking tag 5/7
Looking at Composite: SubSecDateTimeOriginal
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.703
Oops! That tag isn't right. Please try again.
```

Untuk step kelima kita disuruh mengubah metadata

SubSecDateTimeOriginal, begini caranya

```
➜ ~ a/pico-ctf/forensic/medium/blast-from-the-past 11:52:33
) exiftool -SubSecDateTimeOriginal="1970:01:01 00:00:00.001" original_modified.jpg
1 image files updated
```

Kita coba kirim ke server

```
Checking tag 4/7
Looking at Composite: SubSecCreateDate
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.001
Great job, you got that one!

Checking tag 5/7
Looking at Composite: SubSecDateTimeOriginal
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.001
Great job, you got that one!

Checking tag 6/7
Looking at Composite: SubSecModifyDate
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.703
Oops! That tag isn't right. Please try again.
```

Untuk step ke enam kita disuruh mengubah metadata SubSecModifyDate, begini caranya

```
➜ ~ a/pico-ctf/forensic/medium/blast-from-the-past 11:54:03
) exiftool -SubSecModifyDate="1970:01:01 00:00:00.001" original_modified.jpg
1 image files updated
```

Kita coba kirim keserver

```
Checking tag 5/7
Looking at Composite: SubSecDateTimeOriginal
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.001
Great job, you got that one!

Checking tag 6/7
Looking at Composite: SubSecModifyDate
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.001
Great job, you got that one!

Checking tag 7/7
Timezones do not have to match, as long as it's the equivalent time.
Looking at Samsung: TimeStamp
Looking for '1970:01:01 00:00:00.001+00:00'
Found: 2023:11:20 20:46:21.420+00:00
Oops! That tag isn't right. Please try again.
```

Untuk step ketujuh kita disuruh mengubah metadata TimeStamp, begini caranya

```
➜ ~ /a/pico-ctf/forensic/medium/blast-from-the-past 11:54:33
❯ exiftool -TimeStamp="1970:01:01 00:00:00.001+00:00" original_modified.jpg
Warning: Not an integer for XMP-apple-fi:TimeStamp
  0 image files updated
  1 image files unchanged
```

Hmm, warning not an integer, coba kita lihat struktur metadatanya dengan menggunakan -v3

```
exiftool -v3 original_modified.jpg
```

```
TimeStamp = 1700513181420
- Tag '0x0a01' (13 bytes):
  2b82c4: 31 37 30 30 35 31 33 31 38 31 34 32 30 [1700513181420]
```

Pantas saja tidak bisa, server membaca tag tersebut kemudian server mengkonversi integer itu ke format tanggal, jadi hasilnya tidak sesuai, kita akan edit hex-nya menggunakan hexedit menjadi 0000000000000001

```
TimeStamp = 0000000000000001
- Tag '0x0a01' (13 bytes):
  2b82c4: 30 30 30 30 30 30 30 30 30 30 30 30 31 [0000000000000001]
```

Kita coba kirim ke server

```
Checking tag 6/7
Looking at Composite: SubSecModifyDate
Looking for '1970:01:01 00:00:00.001'
Found: 1970:01:01 00:00:00.001
Great job, you got that one!

Checking tag 7/7
Timezones do not have to match, as long as it's the equivalent time.
Looking at Samsung: TimeStamp
Looking for '1970:01:01 00:00:00.001+00:00'
Found: 1970:01:01 00:00:00.001+00:00
Great job, you got that one!

You did it!
picoCTF{71m3_7r4v311ng_p1c7ur3_a4f2b526}
```

Yeay dapat flagnya

picoCTF{71m3_7r4v311ng_p1c7ur3_a4f2b526}

Mob Psycho

Can you handle APKs?

Attachment: mobPsycho.apk

Kita diberi file apk, langsung saja kita extract dan lihat dalamnya ada apa

```
➜ ~ a/pico-ctf/forensic/medium/mob-psych0
↳ unzip mobpsycho.apk
Archive: mobpsycho.apk

[...]
[...] ➜ ~ a/pico-ctf/forensic/medium/mob-psych0/extract
↳ ls
AndroidManifest.xml  classes3.dex  META-INF  resources.arsc
classes2.dex         classes.dex   res
```

Coba kita cari apakah ada file yang mengandung kata flag, begini command nya

```
➜ ~ a/pico-ctf/forensic/medium/mob-psych0/extract
↳ find . -type f -iname "*flag*"
./res/color/flag.txt
```

Wah sepertinya itu adalah flagnya, coba kita lihat

```
➜ ~ a/pico-ctf/forensic/medium/mob-psych0/extract
↳ cat ./res/color/flag.txt
7069636f4354467b6178386d433052553676655f4e5838356c346178386d436c5f62313132616535377d
```

Sepertinya itu adalah enkripsi hex, coba kita decode

```
➜ ~ a/pico-ctf/forensic/medium/mob-psych0/extract
↳ echo "7069636f4354467b6178386d433052553676655f4e5838356c346178386d436c5f62313132616535377d"
| xxd -r -p
picoCTF{ax8mCORU6ve_NX85l4ax8mCl_b112ae57}%
```

Nah, kita sudah dapat flagnya

picoCTF{ax8mCORU6ve_NX85l4ax8mCl_b112ae57}



Endianness-v2

Here's a file that was recovered from a 32-bits system that organized the bytes a weird way. We're not even sure what type of file it is.

Attachment: challengeFile

disini kita diberikan sebuah file, coba kita cek itu file apa

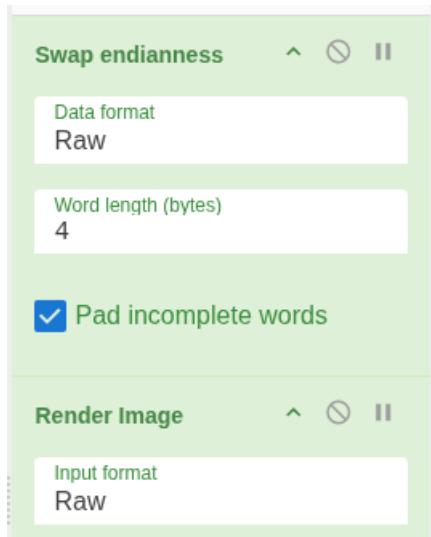
```
~/abyan/pico-ctf/forensic/medium/endianness-v2 11:44:51
> file challengefile
challengefile: data
```

Hmm kayaknya file tersebut rusak, sepertinya jika kita lihat pada soal terdapat sebuah petunjuk bahwa file tersebut berasal dari sistem 32-bits dan pada judul soalnya adalah endianness, sepertinya ini byte pada file nya rusak karena file tersebut dari sistem 32 bit, coba kita lihat hex-nya begini caranya

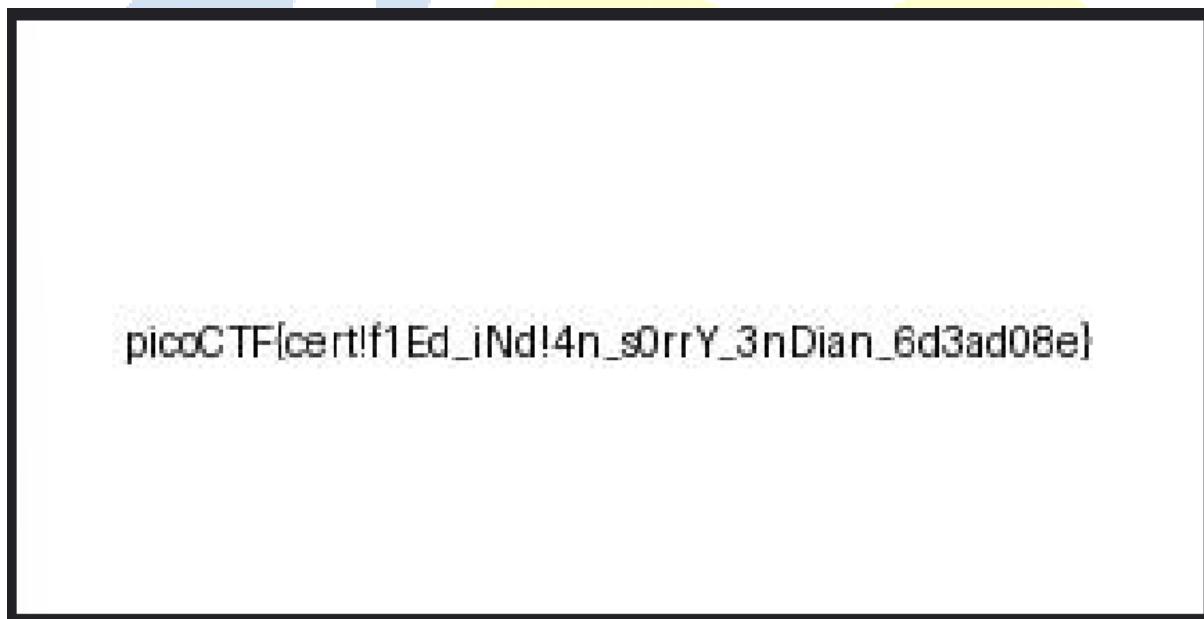
hexedit challengeFile

00000000	E0 FF D8 FF	46 4A 10 00	01 00 46 49	01 00 00 01FJ....FI....
00000010	00 00 01 00	43 00 DB FF	06 06 08 00	08 05 06 07C.....
00000020	09 07 07 07	0C 0A 08 09	0B 0C 0D 14	12 19 0C 0B
00000030	1D 14 0F 13	1D 1E 1F 1A	20 1C 1C 1A	20 27 2E 24 '.\$
00000040	1C 23 2C 22	29 37 28 1C	34 31 30 2C	27 1F 34 34	.#,")7(.410,'.44
00000050	32 38 3D 39	34 33 2E 3C	00 DB FF 32	09 09 01 43	28=943.<...2...C
00000060	0C 0B 0C 09	18 0D 0D 18	21 1C 21 32	32 32 32 32!..!22222
00000070	32 32 32 32	32 32 32 32	32 32 32 32	32 32 32 32	2222222222222222
00000080	32 32 32 32	32 32 32 32	32 32 32 32	32 32 32 32	2222222222222222
00000090	32 32 32 32	32 32 32 32	32 32 32 32	C0 FF 32 32	22222222222222..22
000000A0	00 08 11 00	03 2C 01 96	02 00 22 01	11 03 01 11,...."
000000B0	00 C4 FF 01	01 00 00 1F	01 01 01 05	00 01 01 01
000000C0	00 00 00 00	01 00 00 00	05 04 03 02	09 08 07 06
000000D0	C4 FF 0B 0A	00 10 B5 00	03 03 01 02	05 03 04 02
000000E0	00 04 04 05	01 7D 01 00	04 00 03 02	21 12 05 11}.....!...
000000F0	13 06 41 31	22 07 61 51	81 32 14 71	23 08 A1 91	..A1".aQ.2.q#...
00000100	15 C1 B1 42	24 F0 D1 52	82 72 62 33	17 16 0A 09	...B\$..R.rb3...
00000110	25 1A 19 18	29 28 27 26	36 35 34 2A	3A 39 38 37	%...)(`&654*:987
00000120	46 45 44 43	4A 49 48 47	56 55 54 53	5A 59 58 57	FEDCJIHGVTTSZYXW
00000130	66 65 64 63	6A 69 68 67	76 75 74 73	7A 79 78 77	fedcjihgvutszyxw
00000140	86 85 84 83	8A 89 88 87	95 94 93 92	99 98 97 96
00000150	A4 A3 A2 9A	A8 A7 A6 A5	B3 B2 AA A9	B7 B6 B5 B4
00000160	C2 BA B9 B8	C6 C5 C4 C3	CA C9 C8 C7	D5 D4 D3 D2
00000170	D9 D8 D7 D6	E3 E2 E1 DA	E7 E6 E5 E4	F1 EA E9 E8
00000180	F5 F4 F3 F2	F9 F8 F7 F6	00 C4 FF FA	03 00 01 1F
00000190	01 01 01 01	01 01 01 01	00 00 00 01	01 00 00 00
000001A0	05 04 03 02	09 08 07 06	C4 FF 0B 0A	00 11 B5 00
000001B0	04 02 01 02	07 04 03 04	00 04 04 05	00 77 02 01w..
000001C0	11 03 02 01	31 21 05 04	51 41 12 06	13 71 61 071!..QA...qa.
000001D0	08 81 32 22	A1 91 42 14	23 09 C1 B1	15 F0 52 33	..2"..B.#....R3
000001E0	0A D1 72 62	E1 34 24 16	18 17 F1 25	27 26 1A 19	..rb.4\$....%'&..
000001F0	35 2A 29 28	39 38 37 36	45 44 43 3A	49 48 47 46	5*) (9876EDC:IHGF

Hmm seharusnya magic number-nya jpg itu "FF D8 FF E0" tapi pada file tersebut "E0 FF D8 FF" sepertinya file kebalik tiap 4 bytes, kita akan perbaiki menggunakan cyber chef, begini caranya



cari swap endianness lalu pilih yang raw dan 4 bytes, langsung saja kita download file yang sudah diperbaiki



picoCTF{cert!f1Ed_iNd!4n_s0rry_3nDian_6d3ad08e}

picoCTF{cert!f1Ed_iNd!4n_s0rry_3nDian_6d3ad08e}

MSB

This image passes LSB statistical analysis, but we can't help but think there must be something to the visual artifacts present in this image...

Attachment: file.png

Kita diberikan file png, soal memberikan petunjuk bahwa pada LSB sudah dianalisis tetapi masih file masih rusak, saya curiga bahwa ada yang aneh pada bagian MSB-nya, langsung saja kita coba bikin script untuk extract MSB-nya

stego.py

```
#!/usr/bin/env python3
import argparse
from PIL import Image
import sys

def extract_bits(pixel_value, bit_positions):
    """
    Extract specific bits from a pixel value.
    bit_positions: string of 8 chars, '1' means extract
    that bit, '0' means skip
    Example: '10000000' extracts MSB, '00000001' extracts
    LSB
    """
    binary = format(pixel_value, '08b')
    extracted = []

    for i, should_extract in enumerate(bit_positions):
        if should_extract == '1':
            extracted.append(binary[i])

    return extracted

def decode_image(input_file, output_file, bit_pattern,
channel_order='RGB', extract_mode='column'):
    """
    Decode hidden data from an image using steganography.

    Args:
        input_file: Path to input image
    
```

```

    output_file: Path to output text file
    bit_pattern: 8-character string indicating which
    bits to extract (e.g., '10000000' for MSB)
    channel_order: Order to read RGB channels
    (default: 'RGB')
    extract_mode: 'column' or 'row' traversal
    (default: 'column')
"""

try:
    img = Image.open(input_file)
    width, height = img.size

    # Convert to RGB if necessary
    if img.mode != 'RGB':
        img = img.convert('RGB')

    binary_data = []

    # Determine traversal pattern
    if extract_mode == 'row':
        coordinates = [(x, y) for y in range(height)
                        for x in range(width)]
    else: # column
        coordinates = [(x, y) for x in range(width)
                        for y in range(height)]

    # Extract bits from each pixel
    for x, y in coordinates:

        pixel = img.getpixel((x, y))

        # Extract bits from each channel based on order
        channel_data = {}
        channel_data['R'] = extract_bits(pixel[0],
                                         bit_pattern)
        channel_data['G'] = extract_bits(pixel[1],
                                         bit_pattern)
        channel_data['B'] = extract_bits(pixel[2],
                                         bit_pattern)

        # Add bits in specified channel order
        for channel in channel_order.upper():
            binary_data.extend(channel_data[channel])

```

```

# Convert binary to ASCII text
output_text = []
for i in range(0, len(binary_data) - 7, 8):
    byte = ''.join(binary_data[i:i+8])
    ascii_val = int(byte, 2)

    # Only include printable ASCII characters
    if 32 <= ascii_val <= 126:
        output_text.append(chr(ascii_val))
    elif ascii_val == 10 or ascii_val == 13: # newline or carriage return
        output_text.append(chr(ascii_val))

# Write output
with open(output_file, 'w', encoding='utf-8') as f:
    f.write(''.join(output_text))

print(f"✓ Successfully decoded {input_file}")
print(f"✓ Output saved to {output_file}")
print(f"✓ Extracted {len(output_text)} characters")

except FileNotFoundError:
    print(f"✗ Error: File '{input_file}' not found",
          file=sys.stderr)
    sys.exit(1)
except Exception as e:
    print(f"✗ Error: {str(e)}", file=sys.stderr)
    sys.exit(1)

def main():
    parser = argparse.ArgumentParser(
        prog='StegoDecode',
        description='Steganography decoder for extracting hidden data from images using MSB/LSB techniques',
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog='''
```

```

parser.add_argument('-i', '--input',
                    required=True,
                    help='Input image file (PNG, JPG,
                          etc.)')

parser.add_argument('-o', '--output',
                    required=True,
                    help='Output text file for decoded
                          message')

parser.add_argument('-b', '--bits',
                    default='00000001',
                    help='Bit pattern to extract (8
                          chars: 1=extract, 0=skip). '
                          'Examples: "10000000" for
                          MSB, "00000001" for LSB
                          (default: %(default)s)')

parser.add_argument('-c', '--channels',
                    default='RGB',
                    choices=['RGB', 'RBG', 'GRB',
                             'GBR', 'BRG', 'BGR'],
                    help='Channel order for extraction
                          (default: %(default)s)')

parser.add_argument('-m', '--mode',
                    default='column',
                    choices=['column', 'row'],
                    help='Pixel traversal mode
                          (default: %(default)s)')

args = parser.parse_args()

# Validate bit pattern
if len(args.bits) != 8 or not all(c in '01' for c in
args.bits):
    print("X Error: Bit pattern must be exactly 8
          characters of '0' or '1'", file=sys.stderr)
    sys.exit(1)

if args.bits == '00000000':

```

```

print("X Error: Bit pattern must have at least one
      '1'", file=sys.stderr)
sys.exit(1)

# Run decoder
decode_image(args.input, args.output, args.bits, args.
channels, args.mode)

if __name__ == '__main__':
    main()

```

Cara pakainya gini

```

~/abyan/pico-ctf/forensic/medium/msb 2m 30s 08:47:16
> python3 ~/abyan/pico-ctf/script/stego/stego.py -i Ninja-and-Prince-Genji-Ukiyoe-Utagawa-Kuni
sada.flag.png -o flag.txt -b 10000000 -m raw
✓ Successfully decoded Ninja-and-Prince-Genji-Ukiyoe-Utagawa-Kunisada.flag.png
✓ Output saved to flag.txt
✓ Extracted 539864 characters

```



```

~/abyan/pico-ctf/forensic/medium/msb 08:48:27
> cat flag.txt|grep pico
picoCTF{15_y0ur_que57_qu1x071c_0r_h3r01c_b5e03bc5}

```

Kita sudah dapat flagnya

picoCTF{15_y0ur_que57_qu1x071c_0r_h3r01c_b5e03bc5}

pens

Pcap-Poisoning

How about some hide and seek heh?

Attachment: trace.pcap

Kita diberikan file .pcap, pada deskripsi soal tidak diberikan petunjuk apapun coba kita cek terlebih dahulu file tersebut

```
➜ ~/abyan/pico-ctf/forensic/medium/pcap-poisoning ➜ 10:11:00 ⏺
❯ strings trace.pcap | grep pico
picoCTF{P64P_4N4L7S1S_SU55355FUL_fc4e803f}3~
```

wah ternyata flag-nya ada di strings

picoCTF{P64P_4N4L7S1S_SU55355FUL_fc4e803f}

pens

Dear Diary

If you can find the flag on this disk image, we can close the case for good!

Attachment: disk.flag.img

kita diberi file image, kita coba lihat dulu file tersebut

```
~/abyan/pico-ctf/forensic/medium/dear-diary 20:48:32
$ fdisk -l disk.flag.img
Disk disk.flag.img: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x6062d30a

Device      Boot   Start     End Sectors  Size Id Type
disk.flag.img1 *       2048  616447  614400 300M 83 Linux
disk.flag.img2        616448 1140735  524288 256M 82 Linux swap / Solaris
disk.flag.img3        1140736 2097151  956416 467M 83 Linux
```

Disk tersebut berisi 3 partisi, coba kita mount satu-satu, begini commandnya

```
~/abyan/pico-ctf/forensic/medium/dear-diary 20:56:59
$ sudo mount -o loop,offset=$((2048*512)) disk.flag.img /mnt/disk1
~/abyan/pico-ctf/forensic/medium/dear-diary 20:58:12
$ ls -la /mnt/disk1
total 23490
drwxr-xr-x 3 root root    1024 Dec  8  2023 .
drwxr-xr-x 3 root root    4096 Dec 29 20:56 ..
lrwxrwxrwx 1 root root      1 Dec  8  2023 boot -> .
-rw-r--r-- 1 root root  143958 Dec  8  2023 config-virt
-rw-r--r-- 1 root root    399 Dec  8  2023 extlinux.conf
-rw------- 1 root root 7979329 Dec  8  2023 initramfs-virt
-rw-r--r-- 1 root root  115468 Dec  8  2023 ldlinux.c32
-rw-r--r-- 1 root root  69632 Dec  8  2023 ldlinux.sys
-rw-r--r-- 1 root root 178532 Dec  8  2023 libcom32.c32
-rw-r--r-- 1 root root  23636 Dec  8  2023 libutil.c32
drwx----- 2 root root  12288 Dec  8  2023 lost+found
-rw-r--r-- 1 root root 11712 Dec  8  2023 mboot.c32
-rw-r--r-- 1 root root  26568 Dec  8  2023 menu.c32
-rw-r--r-- 1 root root 5117629 Dec  8  2023 System.map-virt
-rw-r--r-- 1 root root 27020 Dec  8  2023 vesamenu.c32
-rw-r--r-- 1 root root 10337120 Dec  8  2023 vmlinuz-virt
```

Coba kita lihat-lihat dulu apakah ada yang menarik, tapi sepertinya tidak ada yang menarik semua, kita lanjut ke partisi yang ketiga

```
~/abyan/pico-ctf/forensic/medium/dear-diary 21:02:13
$ sudo mount -o loop,offset=$((1140736*512)) disk.flag.img /mnt/disk3
mount: /mnt/disk3: cannot mount; probably corrupted filesystem on /dev/loop0.
      dmesg(1) may have more information after failed mount system call.
```

Wah kenapa ini kok error, sepertinya file system sengaja dibuat corrupt, coba kita mount tanpa journal dan read only karena file system tersebut corrupt

```
~$ ~/abyan/pico-ctf/forensic/medium/dear-diary 21:05:59
) # Mount dengan noload (ignore journal)
sudo mount -o loop,offset=$((1140736*512)),ro,noload disk.flag.img /mnt/disk3
```

Nah kan bisa, coba kita cek apa saja isinya

```
~$ ~/abyan/pico-ctf/forensic/medium/dear-diary 21:06:54
) ls -la /mnt/disk3
total 45
drwxr-xr-x 22 root root 1024 Dec  8 2023 .
drwxr-xr-x  4 root root 4096 Dec 29 20:59 ..
drwxr-xr-x  2 root root 3072 Dec  8 2023 bin
drwxr-xr-x  2 root root 1024 Dec  8 2023 boot
drwxr-xr-x  2 root root 1024 Dec  8 2023 dev
drwxr-xr-x 31 root root 3072 Feb 17 2024 etc
drwxr-xr-x  2 root root 1024 Dec  8 2023 home
drwxr-xr-x 10 root root 1024 Dec  8 2023 lib
drwx----- 2 root root 12288 Dec  8 2023 lost+found
drwxr-xr-x  5 root root 1024 Dec  8 2023 media
drwxr-xr-x  2 root root 1024 Dec  8 2023 mnt
drwxr-xr-x  2 root root 1024 Dec  8 2023 opt
drwxr-xr-x  2 root root 1024 Dec  8 2023 proc
drwx----- 3 root root 1024 Feb 17 2024 root
drwxr-xr-x  2 root root 1024 Dec  8 2023 run
drwxr-xr-x  2 root root 6144 Dec  8 2023 sbin
drwxr-xr-x  2 root root 1024 Dec  8 2023 srv
drwxr-xr-x  2 root root 1024 Dec  8 2023 swap
drwxr-xr-x  2 root root 1024 Dec  8 2023 sys
drwxrwxrwt  2 root root 1024 Dec  8 2023 tmp
drwxr-xr-x  8 root root 1024 Dec  8 2023 usr
drwxr-xr-x 11 root root 1024 Dec  8 2023 var
```

Biasanya flag tersimpan di home atau engga root, coba kita cek

```
~$ ~/abyan/pico-ctf/forensic/medium/dear-diary 21:15:40
) sudo ls -la /mnt/disk3/root
total 4
drwx----- 3 root root 1024 Feb 17 2024 .
drwxr-xr-x 22 root root 1024 Dec  8 2023 ..
-rw----- 1 root root 27 Feb 17 2024 .ash_history
drwxr-xr-x  2 root root 1024 Feb 17 2024 secret-secrets
```

Secret-secrets? Sepertinya mencurigakan ada sesuatu yang dirahasiakan

```
~$ ~/abyan/pico-ctf/forensic/medium/dear-diary 21:15:54
) sudo ls -la /mnt/disk3/root/secret-secrets
total 3
drwxr-xr-x 2 root root 1024 Feb 17 2024 .
drwx----- 3 root root 1024 Feb 17 2024 ..
-rwxr-xr-x 1 root root 21 Feb 17 2024 force-wait.sh
-rw-r--r-- 1 root root 0 Feb 17 2024 innocuous-file.txt
-rw-r--r-- 1 root root 0 Feb 17 2024 its-all-in-the-name
```

Wah sepertinya ada yang janggal pada file innocuous.txt, tapi kok size file tersebut kosong ya? Mungkin ada hubungannya sama petunjuk pada soal yaitu kita disuruh lihat binary data raw nya, coba kita lihat menggunakan hexdump, begini command nya

```
~$ ~/abyan/pico-ctf/forensic/medium/dear-diary 21:34:48
) hexdump -C disk.flag.img|less
```

```

00000170 ff e4 e8 1e 00 4f 70 65 72 61 74 69 6e 67 20 73 |.....Operating s|
00000180 79 73 74 65 6d 20 6c 6f 61 64 20 65 72 72 6f 72 |ystem load error|
00000190 2e 0d 0a 5e ac b4 0e 8a 3e 62 04 b3 07 cd 10 3c |...^....>b.....<|
000001a0 0a 75 f1 cd 18 f4 eb fd 00 00 00 00 00 00 00 00 |.u.....|
000001b0 00 00 00 00 00 00 00 00 0a d3 62 60 00 00 80 20 |.....b`...|
000001c0 21 00 83 5e 38 26 00 08 00 00 00 60 09 00 00 5e |!..^8&....^...^|
000001d0 39 26 82 01 3a 47 00 68 09 00 00 00 08 00 00 01 |9&.:G.h.....|
000001e0 3b 47 83 8a 08 82 00 68 11 00 00 98 0e 00 00 00 |;6.....h.....|
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 55 aa |.....U.|

/inno

```

Coba kita search menggunakan kata kunci inno, caranya dengan mengetik “/inno”, kemudian tunggu sampai ketemu karena file img tersebut besar kemungkinan lama

```

23c3c830 34 07 00 00 28 00 12 01 69 6e 6e 6f 63 75 6f 75 |4...(...innocuou|
23c3c840 73 2d 66 69 6c 65 2e 74 78 74 00 00 00 00 00 00 |s-file.txt.....|
23c3c850 00 00 00 00 00 00 00 00 35 07 00 00 9c 03 13 01 |.....5.....|
23c3c860 69 74 73 2d 61 6c 6c 2d 69 6e 2d 74 68 65 2d 6e |its-all-in-the-n|
23c3c870 61 6d 65 00 00 00 00 00 00 00 00 00 00 00 00 00 |ame.....|
23c3c880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|

```

Hmm tidak ada apa, coba kita tekan “n” untuk next, kemudian tunggu

```

2b6fe030 34 07 00 00 38 00 12 01 69 6e 6e 6f 63 75 6f 75 |4...8...innocuou|
2b6fe040 73 2d 66 69 6c 65 2e 74 78 74 00 00 00 00 00 00 |s-file.txt.....|
2b6fe050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
2b6fe060 00 00 00 00 00 00 00 00 35 07 00 00 8c 03 03 01 |.....5.....|
2b6fe070 70 69 63 00 00 00 00 00 00 00 00 00 00 00 00 00 |pic.....|
2b6fe080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|

```

“pic” sepertinya flag tersebut dipisah-pisah menjadi beberapa bagian kita coba cek selanjutnya

```

2b6ff830 34 07 00 00 1c 00 12 01 69 6e 6e 6f 63 75 6f 75 |4.....innocuou|
2b6ff840 73 2d 66 69 6c 65 2e 74 78 74 00 00 35 07 00 00 |s-file.txt..5...|
2b6ff850 a8 03 03 01 6f 43 54 00 00 00 00 00 00 00 00 00 |....oCT.....|
2b6ff860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|

```

Kata selanjutnya “oCT” sepertinya benar, cari semua bagian kemudian satukan bagian-bagian tersebut

picoCTF{1_533_n4m35_80d24b30}



Sleuthkit Intro

Download the disk image and use mmls on it to find the size of the Linux partition. Connect to the remote checker service to check your answer and get the flag. Note: if you are using the webshell, download and extract the disk image into /tmp not your home directory. Download disk imageAccess checker program: nc saturn.picoctf.net 58881

Attachment: disk.flag.img

Kita diberi disk image, soal memberitahu kita untuk menggunakan mmls dan menjawab pertanyaan pada server yang diberikan, langsung saja kita coba

```
~$ ./mmls disk.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot Start End Length Description
000: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
001: ----- 0000000000 0000002047 0000002048 Unallocated
002: 000:000 0000002048 0000204799 0000202752 Linux (0x83)
```

Kita coba menjawab pertanyaan yang ada di server

```
~$ nc saturn.picoctf.net 58881
What is the size of the Linux partition in the given disk image?
Length in sectors: 202752
202752
Great work!
picoCTF{mm15_f7w!}
```

Yeayy sudah dapat flagnya

picoCTF{mm15_f7w!}



Sleuthkit Apprentice

Download this disk image and find the flag. Note: if you are using the webshell, download and extract the disk image into /tmp not your home directory.

Attachment: disk.flag.img

Kita diberi file disk image, coba kita lihat menggunakan the sleuthkit

```
~/.pico-ctf/forensic/medium/sleuthkit-apprentice 02:20:00
> mmls disk.flag.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End        Length     Description
000: Meta    0000000000  0000000000  0000000001 Primary Table (#0)
001: -----  0000000000  0000002047  0000002048 Unallocated
002: 000:000  0000002048  0000206847  0000204800 Linux (0x83)
003: 000:001  0000206848  0000360447  0000153600 Linux Swap / Solaris x86 (0x82)
004: 000:002  0000360448  0000614399  0000253952 Linux (0x83)
```

Terdapat 3 partisi tetapi partisi yang kedua hanya swap, coba kita mount partisi yang pertama begini caranya

```
~/.pico-ctf/forensic/medium/sleuthkit-apprentice 02:21:33
> sudo mount -o loop,ro,offset=$((512*2048)) disk.flag.img mnt/
[sudo] password for kali:
```

Kita coba lihat apa saja yang ada didalam

```
~/.pico-ctf/forensic/medium/sleuthkit-apprentice/mnt 02:26:19
> ll
total 16M
lrwxrwxrwx 1 root root 1 Sep 29 2021 boot -> .
-rw-r--r-- 1 root root 121K Aug 27 2021 config-virt
-rw-r--r-- 1 root root 398 Sep 29 2021 extlinux.conf
-rw----- 1 root root 5.5M Sep 29 2021 initramfs-virt
-r--r--r-- 1 root root 114K Sep 29 2021 ldlinux.c32
-r--r--r-- 1 root root 68K Sep 29 2021 ldlinux.sys
-rw-r--r-- 1 root root 176K Sep 29 2021 libcom32.c32
-rw-r--r-- 1 root root 23K Sep 29 2021 libutil.c32
drwx----- 2 root root 12K Sep 29 2021 Lost+found
-rw-r--r-- 1 root root 12K Sep 29 2021 mboot.c32
-rw-r--r-- 1 root root 26K Sep 29 2021 menu.c32
-rw-r--r-- 1 root root 3.5M Aug 27 2021 System.map-virt
-rw-r--r-- 1 root root 27K Sep 29 2021 vesamenu.c32
-rw-r--r-- 1 root root 6.0M Aug 27 2021 vmlinuz-virt
```

Sepertinya tidak ada yang mencurigakan, kita lanjut ke partisi yang ketiga

```
~/.pico-ctf/forensic/medium/sleuthkit-apprentice 02:30:32
> sudo mount -o loop,ro,offset=$((512*360448)) disk.flag.img mnt3/
```

Coba kita lihat-lihat apa saja yang ada didalam partisi 3 ini

```
ll
total 39K
drwxr-xr-x 2 root root 3.0K Sep 29 2021 bin
drwxr-xr-x 2 root root 1.0K Sep 29 2021 boot
drwxr-xr-x 2 root root 1.0K Sep 29 2021 dev
drwxr-xr-x 27 root root 3.0K Sep 29 2021 etc
drwxr-xr-x 2 root root 1.0K Sep 29 2021 home
drwxr-xr-x 9 root root 1.0K Sep 29 2021 lib
drwx----- 2 root root 12K Sep 29 2021 lost+found
drwxr-xr-x 5 root root 1.0K Sep 29 2021 media
drwxr-xr-x 2 root root 1.0K Sep 29 2021 mnt
drwxr-xr-x 2 root root 1.0K Sep 29 2021 opt
drwxr-xr-x 2 root root 1.0K Sep 29 2021 proc
drwx----- 3 root root 1.0K Sep 29 2021 root
drwxr-xr-x 2 root root 1.0K Sep 29 2021 run
drwxr-xr-x 2 root root 5.0K Sep 29 2021 sbin
drwxr-xr-x 2 root root 1.0K Sep 29 2021 srv
drwxr-xr-x 2 root root 1.0K Sep 29 2021 swap
drwxr-xr-x 2 root root 1.0K Sep 29 2021 sys
drwxrwxrwt 4 root root 1.0K Sep 29 2021 tmp
drwxr-xr-x 8 root root 1.0K Sep 29 2021 usr
drwxr-xr-x 11 root root 1.0K Sep 29 2021 var
```

Wah, biasanya flag disembunyikan di directory root, coba kita cek

```
ls root
my_folder
```

Ada my_folder coba kita cek

```
ls root/my_folder
flag.uni.txt
```

Wah ada flag.txt, coba kita baca

```
cat root/my_folder/flag.uni.txt
picoCTF{by73_5urf3r_2f22df38}
```

picoCTF{by73_5urf3r_2f22df38}



Redaction Gone Wrong

Now you DON'T see me. This report has some critical data in it, some of which have been redacted correctly, while some were not. Can you find an important key that was not redacted properly?

Attachment: file.pdf

Kita diberi file pdf, pada soal memberi petunjuk redaction coba ktia buka file pdf tersebut

Financial Report for ABC Labs, Kigali, Rwanda for the year 2021.

[REDACTED] - Just painted over in MS word.

[REDACTED]

Cost Benefit Analysis

Credit Debit

[REDACTED]

Expenses from the [REDACTED]

[REDACTED]

Redacted document.

Hmm ada bagian yang ditutupi sama kotak hitam, coba kita select kotak tersebut

Financial Report for ABC Labs, Kigali, Rwanda for the year 2021.

Breakdown - Just painted over in MS word.

[REDACTED]

Cost Benefit Analysis

Credit Debit

This is not the flag, keep looking

Expenses from the [REDACTED]

picoCTF{C4n_Y0u_S33_m3_fully}

Redacted document.

Wah ternyata itu adalah text yang ditutupi sama kotak hitam

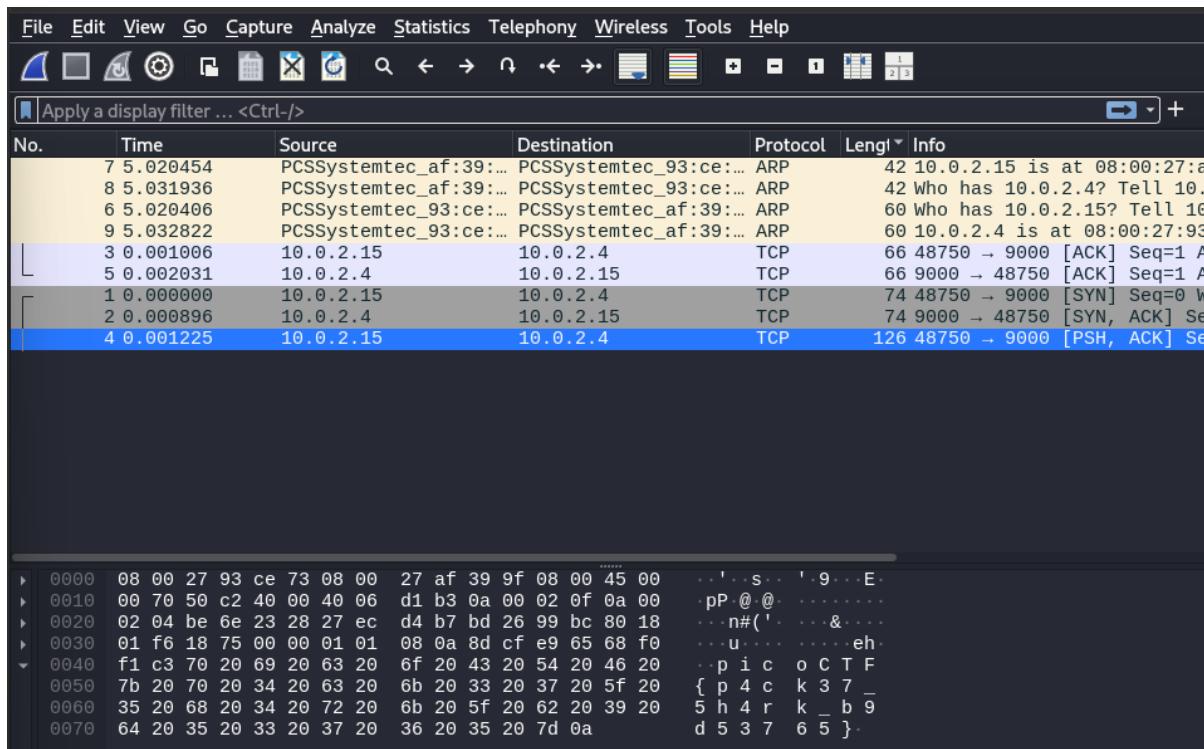
picoCTF{c4n_y0u_S33_m3_fully}

Redaction Gone Wrong

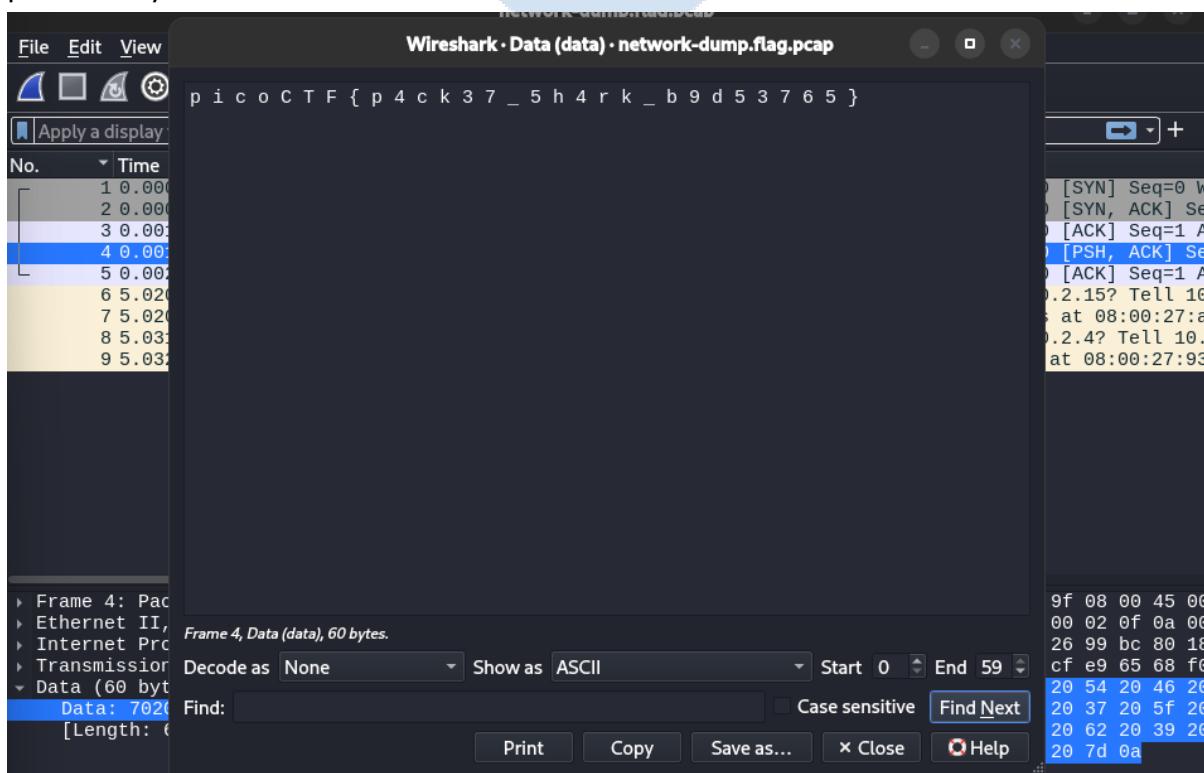
Now you DON'T see me. This report has some critical data in it, some of which have been redacted correctly, while some were not. Can you find an important key that was not redacted properly?

Attachment: file.pcap

Disini kita diberikan file pcap, coba kita lihat menggunakan wireshark apa isi dari file tersebut



Sepertinya pada packet ke 4 terdapat sebuah flag coba kita show packet bytes



picoCTF{p4ck37_5h4rk_b9d53765}

Operation Orchid

Download this disk image and find the flag. Note: if you are using the webshell, download and extract the disk image into /tmp not your home directory.

Attachment: disk.flag.img

kita diberi file disk image coba kita cek menggunakan sleuthkit begini command-nya

```
~$ ./mmls disk.flag.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start          End          Length       Description
000: Meta      0000000000  0000000000  0000000001 Primary Table (#0)
001: -----    0000000000  0000002047  0000002048 Unallocated
002: 000:000   0000002048  0000206847  0000204800 Linux (0x83)
003: 000:001   0000206848  0000411647  0000204800 Linux Swap / Solaris x86 (0x82)
004: 000:002   0000411648  0000819199  0000407552 Linux (0x83)
```

Terdapat 3 partisi tapi partisi yang kedua adalah swap, coba kita mount partisi kesatu

```
~$ sudo mount -o loop,ro,offset=$((512*2048)) disk.flag.img mnt
[sudo] password for kali:
```

Coba kita cek ada apa didalam

```
~$ ll
total 16M
lrwxrwxrwx 1 root root 1 Oct  6 2021 boot -> .
-rw-r--r-- 1 root root 121K Aug 27 2021 config-virt
-rw-r--r-- 1 root root 398 Oct  6 2021 extlinux.conf
-rw----- 1 root root 5.5M Oct  6 2021 initramfs-virt
-r--r--r-- 1 root root 114K Oct  6 2021 ldlinux.c32
-r--r--r-- 1 root root 68K Oct  6 2021 ldlinux.sys
-rw-r--r-- 1 root root 176K Oct  6 2021 libcom32.c32
-rw-r--r-- 1 root root 23K Oct  6 2021 libutil.c32
drwx----- 2 root root 12K Oct  6 2021 lost+found
-rw-r--r-- 1 root root 12K Oct  6 2021 mboot.c32
-rw-r--r-- 1 root root 26K Oct  6 2021 menu.c32
-rw-r--r-- 1 root root 3.5M Aug 27 2021 System.map-virt
-rw-r--r-- 1 root root 27K Oct  6 2021 vesamenu.c32
-rw-r--r-- 1 root root 6.0M Aug 27 2021 vmlinuz-virt
```

Sepertinya tidak ada yang menarik, coba sekarang kita mount partisi yang ketiga

```
~$ sudo mount -o loop,ro,offset=$((512*411648)) disk.flag.img mnt
```

```
~ /a/pico-ctf/forensic/medium/operation-orchid/mnt 05:28:38
) ll
total 39K
drwxr-xr-x 2 root root 3.0K Oct  6 2021 bin
drwxr-xr-x 2 root root 1.0K Oct  6 2021 boot
drwxr-xr-x 2 root root 1.0K Oct  6 2021 dev
drwxr-xr-x 27 root root 3.0K Oct  6 2021 etc
drwxr-xr-x 2 root root 1.0K Oct  6 2021 home
drwxr-xr-x 9 root root 1.0K Oct  6 2021 lib
drwx----- 2 root root 12K Oct  6 2021 lost+found
drwxr-xr-x 5 root root 1.0K Oct  6 2021 media
drwxr-xr-x 2 root root 1.0K Oct  6 2021 mnt
drwxr-xr-x 2 root root 1.0K Oct  6 2021 opt
drwxr-xr-x 2 root root 1.0K Oct  6 2021 proc
drwx----- 2 root root 1.0K Oct  6 2021 root
drwxr-xr-x 2 root root 1.0K Oct  6 2021 run
drwxr-xr-x 2 root root 5.0K Oct  6 2021 sbin
drwxr-xr-x 2 root root 1.0K Oct  6 2021 srv
drwxr-xr-x 2 root root 1.0K Oct  6 2021 swap
drwxr-xr-x 2 root root 1.0K Oct  6 2021 sys
drwxrwxrwt 4 root root 1.0K Oct  6 2021 tmp
drwxr-xr-x 8 root root 1.0K Oct  6 2021 usr
drwxr-xr-x 11 root root 1.0K Oct  6 2021 var
```

Ada yang menarik coba kita cari dengan kata kunci flag

```
~/.flag.txt.enc
```

Wah ternyata ada file bernama flag coba kita lihat

```
↳ sudo cat ./root/flag.txt.enc  
Salted__QE&$jMGeE1Z7$%
```

Sepertinya file tersebut terenkripsi, coba kita cari passwordnya

```
~ /a/pico-ctf/forensic/medium/operation-orchid/mnt 05:44:30
❯ sudo ls -la ./root/
total 4
drwx----- 2 root root 1024 Oct  6 2021 .
drwxr-xr-x 22 root root 1024 Oct  6 2021 ..
-rw-----  1 root root   202 Oct  6 2021 .ash_history
-rw-r--r--  1 root root    64 Oct  6 2021 flag.txt.enc
```

wah ada history, coba kita cek ada apa disana

```
8 ~ /a/pico-ctf/forensic/medium/operation-orchid/mnt 05:45:22
) sudo cat ./root/.ash_history
touch flag.txt
nano flag.txt
apk get nano
apk --help
apk add nano
nano flag.txt
openssl
openssl aes256 -salt -in flag.txt -out flag.txt.enc -k unbreakablepassword1234567
shred -u flag.txt
ls -al
halt
```

Bingo sepertinya file flag tersebut di enkripsi menggunakan password tersebut coba kita decrypt, begini caranya

```
~$ /abyan/pico-ctf/forensic/medium/operation-orchid 05:49:39 ⓘ
$ sudo openssl aes-256-cbc -d -salt -in flag.txt.enc -out flag.txt -k unbreakablepassword12345
67
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
40374796E27F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad decrypt:../providers/implementations/ciphers/ciphercommon_block.c:107:
~$ cat flag.txt
picoCTF{h4un71ng_p457_5113beab}%
```

Yeay dapat flag nya

picoCTF{h4un71ng_p457_5113beab}



Operation Oni

Download this disk image, find the key and log into the remote machine. Note: if you are using the webshell, download and extract the disk image into /tmp not your home directory.

- Download disk image
- Remote machine: ssh -i key_file -p 60042 ctf-player@saturn.picoctf.net

Attachment: disk.img

kita diberikan file disk image, coba kita lihat menggunakan the sleuth kit

```
~$ ./mmls disk.img
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot      Start        End        Length       Description
000: Meta    0000000000  0000000000  0000000001  Primary Table (#0)
001: -----  0000000000  0000002047  0000002048  Unallocated
002: 000:000  0000002048  0000206847  0000204800  Linux (0x83)
003: 000:001  0000206848  0000471039  0000264192  Linux (0x83)
```

Terdapat 3 partisi, langsung saja coba kita mount partisi yang ketiga

```
~$ ls
bin  dev  home  lost+found  mnt  proc  run  srv  tmp  var
boot etc  lib   media     opt  root  sbin sys  usr
```

Coba kita lihat-lihat apakah ada yang menarik

```
~$ sudo find . -type f -iname "*flag*"
~$ sudo find . -type f -iname "*pico*"
```

Sepertinya tidak ada file yang memiliki kata kunci flag ataupun pico coba kita cek pada directory root

```
~$ sudo ls -la root
total 4
drwx----- 3 root root 1024 Oct  6  2021 .
drwxr-xr-x 21 root root 1024 Oct  6  2021 ..
-rw-----  1 root root   36 Oct  6  2021 .ash_history
drwx----- 2 root root 1024 Oct  6  2021 .ssh
```

Wah ada history, coba kita cek

```
~$ sudo cat ./root/.ash_history
ssh-keygen -t ed25519
ls .ssh/
halt
```

Sepertinya ada history generate ssh keys, coba kita cek di directory .ssh

```
~ /abyan/pico-ctf/forensic/medium/operation-oni/mnt 06:12:20
) sudo ls -la ./root/.ssh
total 4
drwx----- 2 root root 1024 Oct  6 2021 .
drwx----- 3 root root 1024 Oct  6 2021 ..
-rw----- 1 root root  411 Oct  6 2021 id_ed25519
-rw-r--r-- 1 root root   96 Oct  6 2021 id_ed25519.pub
```

Id_ed25519 sepertinya menarik karena di history terdapat history membuat key ssh

```
~ /abyan/pico-ctf/forensic/medium/operation-oni/mnt 06:12:30
) sudo cat ./root/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlnZaC1rZXktdjEAAAABG5vbmUAAAAEb9uZQAAAAAAABAAAAAMwAAAAtzc2gtZW
QyNTUx0QAAACBgrXe4bKNh0zkCLW0mk4zDMimW9RVZngX51Y8h3BmKLAJgxpYKDMAWC
gwAAAAtzc2gtZWQyNTUx0QAAACBgrXe4bKNh0zkCLW0mk4zDMimW9RVZngX51Y8h3BmKLA
AAAECItu0F8DIjWxTp+KeMDvX1lQwYtUvP2SfSV0fMOChxYGCTd7hs02E70QItY6aTjMMY
KZb1FVmeBfnVjyHcGYosAAAAdnJvb3RAbG9jYWxob3N0AQIDBAUGBw==
-----END OPENSSH PRIVATE KEY-----
```

Horee dapat key untuk ssh-nya, langsung saja kita gunakan key tersebut

```
~/abyan/pico-ctf/forensic/medium/operation-oni 06:22:01
) ssh -i key_file -p 6486 ctf-player@saturn.picoctf.net
ctf-player@challenge:~$ ls
flag.txt
ctf-player@challenge:~$ cat flag
cat: flag: No such file or directory
ctf-player@challenge:~$ cat flag.tx
cat: flag.tx: No such file or directory
ctf-player@challenge:~$ cat flag.txt
picoCTF{k3y_5l3u7h_af277f77}ctf-player@challenge:~$
```

Yeay sudah ketemu flagnya

picoCTF{k3y_5l3u7h_af277f77}

