PicoCTF 2018 Writeup: Forensics

Oct 13, 2018 08:56 · 1346 words · 7 minute read

CTF CYBER-SECURITY WRITE-UP PICOCTF FORENSICS

Forensics Warmup 1

Problem

Can you unzip this file for me and retreive the flag?

Solution

Just unzip the file.

flag: `picoCTF{welcome_to_forensics}`

Forensics Warmup 2

Problem

Hmm for some reason I can't open this PNG? Any ideas?

Solution

Using the `file` command, you can see that the image is, in fact, in `jpeg` format not `png`:

```
❯ file flag.png
flag.png: JPEG image data, JFIF standard 1.01
```

Open the image as a `jpeg` file to get the file.

flag: `picoCTF{extensions_are_a_lie}`

Desrouleaux

Problem

Our network administrator is having some trouble handling the tickets for all of of our incidents. Can you help him out by answering all the questions? Connect with `nc 2018shell2.picoctf.com 10493`. incidents.json

Solution

Here is the solution script:

```
from sets import Set
from pwn import *
import json

sh = remote('2018shell2.picoctf.com', 10493)

with open('./incidents.json') as f:
  data = json.loads(f.read())

# question 1
src = {}

for each in data[u'tickets']:
  src_ip = each[u'src_ip']
  if src_ip in src:
    src[src_ip] += 1
  else:
    src[src_ip] = 1

print sh.recvuntil('ones.\n')
sh.sendline(max(src, key=src.get))

# question 2
```

```
target = sh.recvuntil('?\n').split(' ')[-1][:-2]
target_ls = {}
count = 0
for each in data[u'tickets']:
  if each[u'src_ip'] == target and each[u'dst_ip'] not in target_ls:
    target_ls[each[u'dst_ip']] = True
    count += 1

sh.sendline(str(count))

# question 3
hashes = {}
for each in data[u'tickets']:
  hash = each[u'file_hash']
  if hash not in hashes:
    hashes[hash] = Set()
  hashes[hash].add(each[u'dst_ip'])

avg = 0
for each in hashes:
  e = hashes[each]
  avg += len(e)
avg = (avg * 1.0) / len(hashes)

print sh.recvuntil('.\n')
sh.sendline(str(avg))

sh.interactive()
```

flag: `picoCTF{J4y_s0n_d3rUUUULo_a062e5f8}`

Reading Between the Eyes

Problem

Stego-Saurus hid a message for you in this image, can you retreive it?

Solution

This problem is about using the Least Significant Bit algorithm for image steganography.
It can be solved using an online decoder.

flag: `picoCTF{r34d1ng_b37w33n_7h3_by73s}`

Recovering From the Snap

Problem

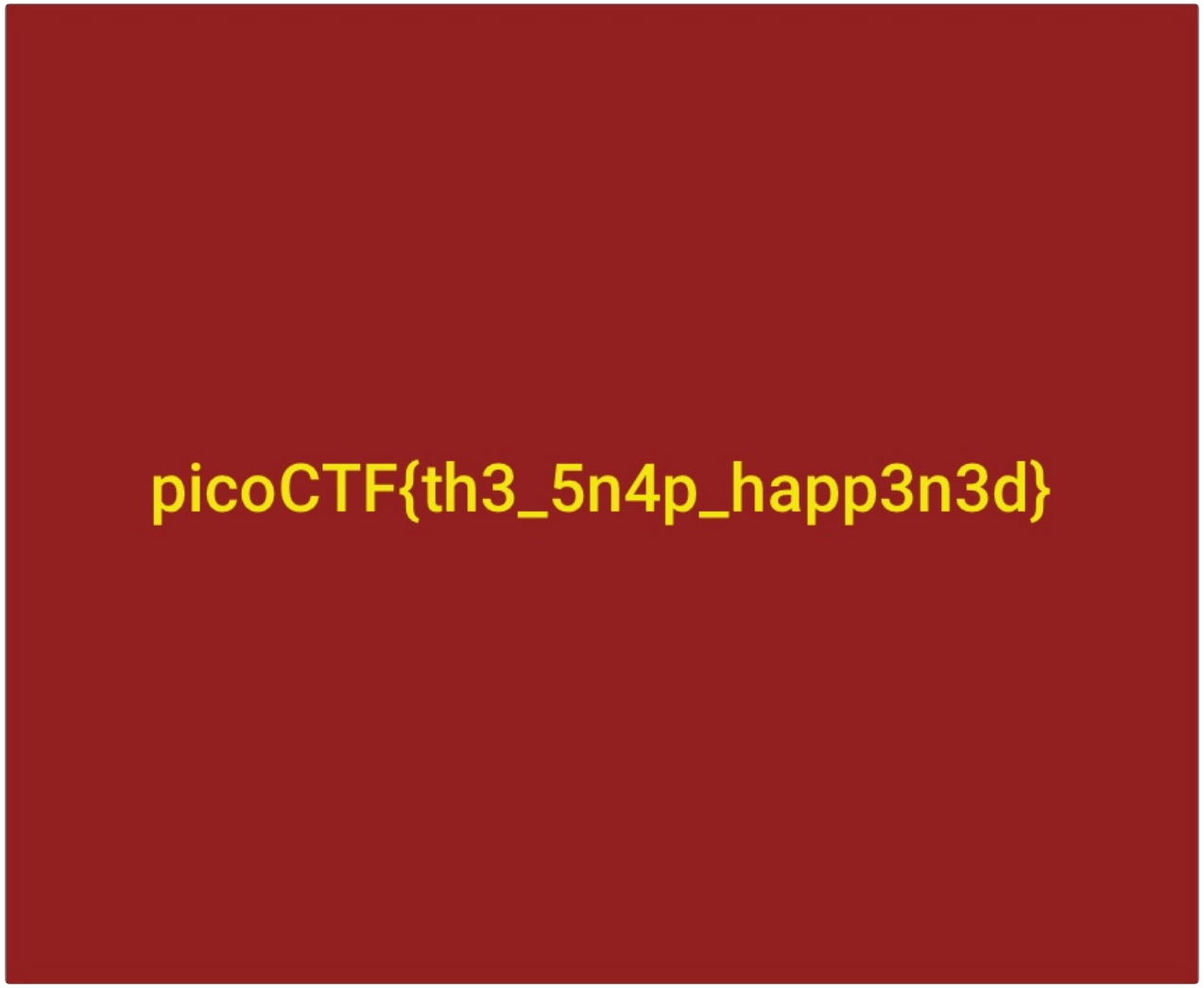
There used to be a bunch of animals here, what did Dr. Xernon do to them?

Solution

This problem is about recovering files from a FAT filesystem. It can be done using TestDisk, a powerful free data recovery software.

You can follow this guide to recover the `theflag.jpg` file.

Recorded with asciinema



theflag.jpg

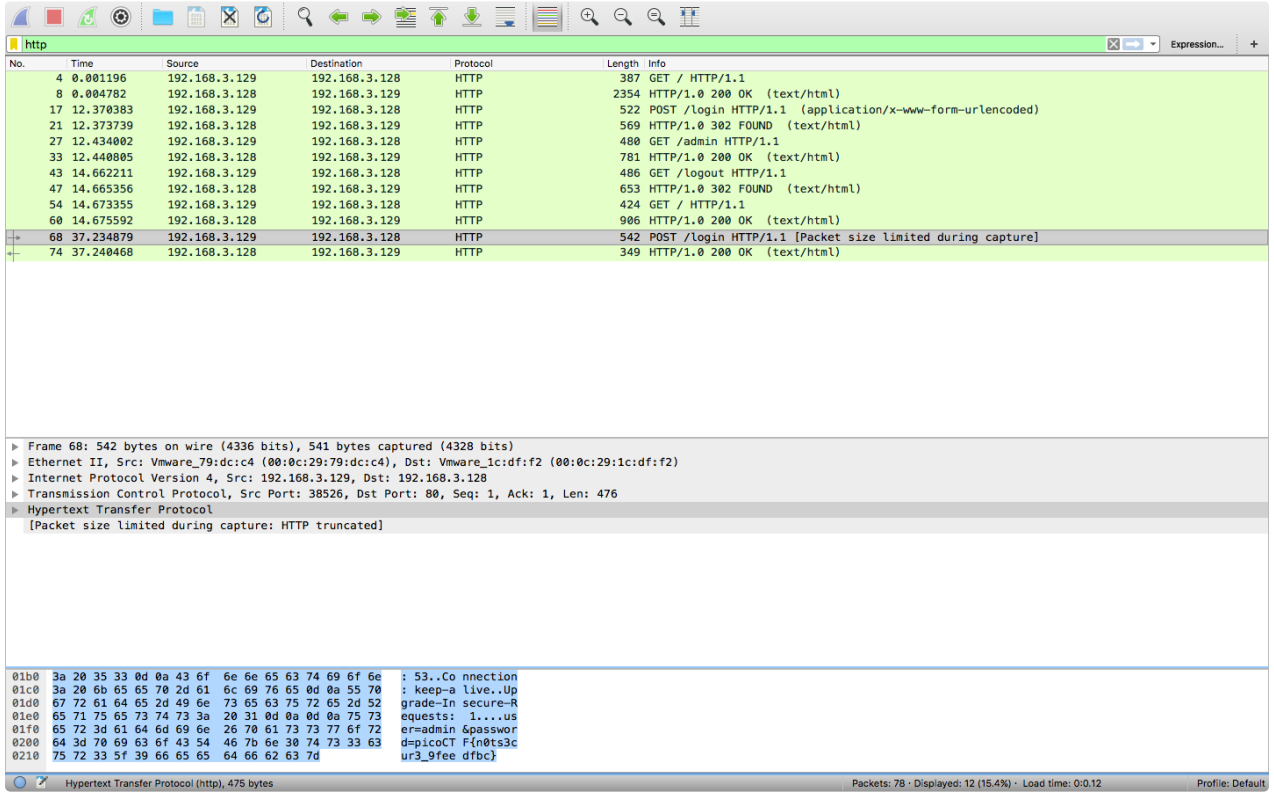
flag: `picoCTF{th3_5n4p_happ3n3d}`

admin panel

Problem

We captured some traffic logging into the admin panel, can you find the password?

Solution



If you look for `http` requests, you will see two login attempts, and the second request contains the flag:

```
POST /login HTTP/1.1
Host: 192.168.3.128
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.3.128/
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Connection: keep-alive
Upgrade-Insecure-Requests: 1

user=admin&password=picoCTF{n0ts3cur3_9feedfbc}
```

flag: `picoCTF{n0ts3cur3_9feedfbc}`

hex editor

Problem

This cat has a secret to teach you. You can also find the file in /problems/hex-editor_4_0a7282b29fa47d68c3e2917a5a0d726b on the shell server.

Solution

You can get the flag by looking at the hex hump of the image or just print out all the readable parts of the file:

```
❯ strings hex_editor.jpg | grep pico
Your flag is: "picoCTF{and_thats_how_u_edit_hex_kittos_dF817ec5}"
```

flag: `picoCTF{and_thats_how_u_edit_hex_kittos_dF817ec5}`

Truly an Artist

Problem

Can you help us find the flag in this Meta-Material? You can also find the file in /problems/truly-an-artist_3_066d6319e350c1d579e5cf32e326ba02.

Solution

The flag is in the EXIF meta-data of the image:

```
❯ exiftool 2018.png
ExifTool Version Number         : 11.01
File Name                       : 2018.png
Directory                       : .
File Size                       : 13 kB
File Modification Date/Time     : 2018:10:09 23:34:05+08:00
File Access Date/Time           : 2018:10:10 09:15:07+08:00
File Inode Change Date/Time     : 2018:10:09 23:34:06+08:00
File Permissions                : rw-r--r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
```

```
Image Width                     : 1200
Image Height                    : 630
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Artist                          : picoCTF{look_in_image_eeea129e}
Image Size                      : 1200x630
Megapixels                      : 0.756
```

flag: `picoCTF{look_in_image_eeea129e}`

now you don't

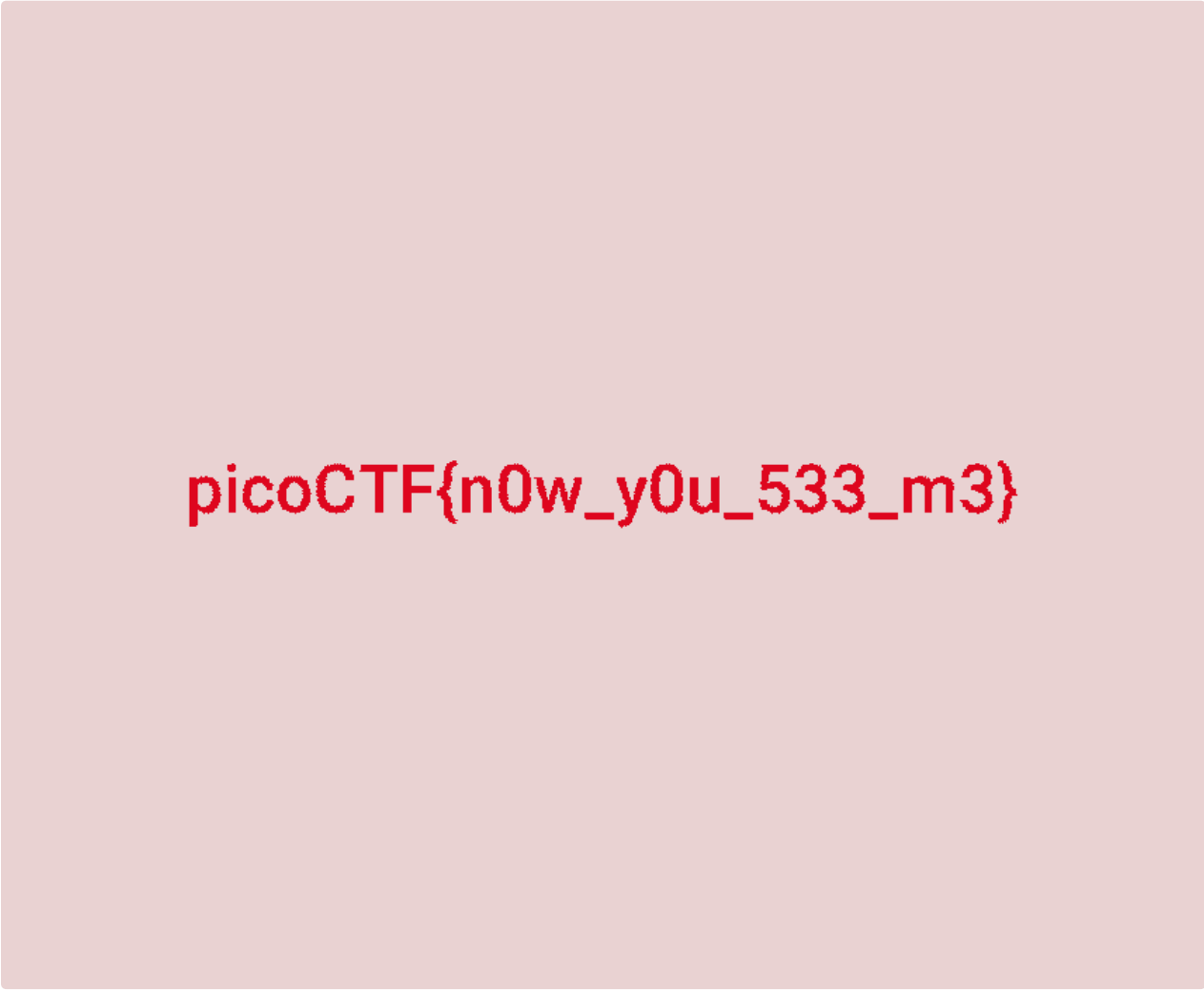
Problem

We heard that there is something hidden in this picture. Can you find it?

Solution

You can create another image with only one shade of red and diff that image with the one provided to get the flag:

```
❯ convert -size 857x703 canvas:"#912020" pure.png
❯ compare nowYouDont.png pure.png diff.png
```



diff.png

flag: `picoCTF{n0w_y0u_533_m3}`

Ext Super Magic

Problem

We salvaged a ruined Ext SuperMagic II-class mech recently and pulled the filesystem out of the black box. It looks a bit corrupted, but maybe there's something interesting in there. You can also find it in /problems/ext-super-magic_4_f196e59a80c3fdac37cc2f331692ef13 on the shell server.

Solution

You are given a ext3 file image that is broken. To fix the image, you have to correct the magic number of the file. You can read more about the ext3 file format over here.

Here is the script that writes the magic number `0xEF53` into the file:

```
# flag: picoCTF{a7DB29eCf7dB9960f0A19Fdde9d00Af0}nc 2018shell2.picoctf.com 2651

from pwn import *

with open('./ext-super-magic.img', 'rb') as f:
  data = f.read()

print enhex(data[1024:1024+82])
print enhex(data[1024+56:1024+56+2])

data = data[:1024+56] + p16(0xEF53) + data[1024+56+2:]

with open('fixed.img', 'wb') as f:
  f.write(data)
```

flag: `picoCTF{a7DB29eCf7dB9960f0A19Fdde9d00Af0}`

Lying Out

Problem

Some odd traffic has been detected on the network, can you identify it? More info here. Connect with `nc 2018shell2.picoctf.com 27108` to help us answer some questions.

Solution

Just read the graph and do this problem by hand.

flag: `picoCTF{w4y_0ut_de051415}`

What's My Name?

Problem

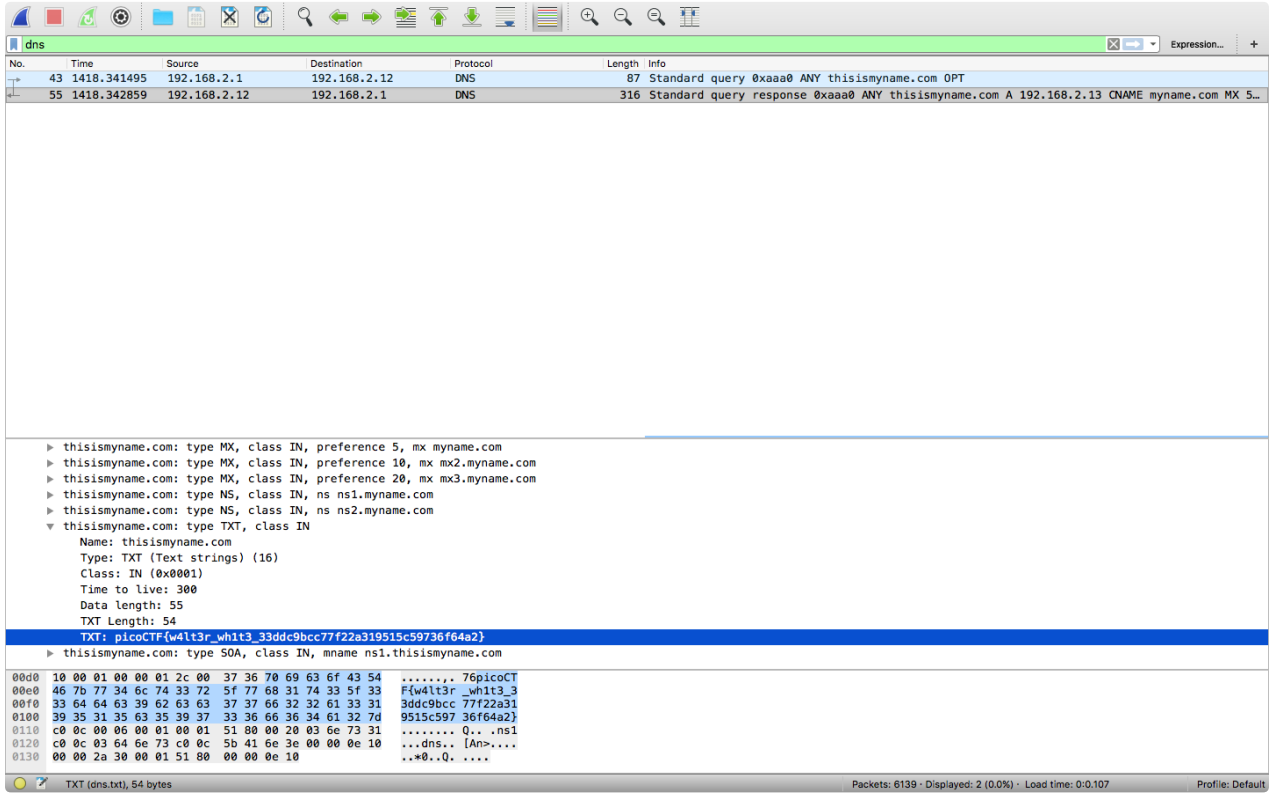
Say my name, say my name.

Solution

The hint is very helpful. It asks `If you visited a website at an IP address, how does it know the name of the domain?`.

The answer to this question is that a domain is resolved through `DNS` packets.

If we only look for `DNS` packets in wireshark, we will find the flag.



flag: `picoCTF{w4lt3r_wh1t3_33ddc9bcc77f22a319515c59736f64a2}`

core

Problem

This program was about to print the flag when it died. Maybe the flag is still in this core file that it dumped? Also available at /problems/core_1_722685357ac5a814524ee76a3dcd1521 on the shell server.

Solution

Let's first take a look at the program using radare2:

```
[0x080484c0]> s sym.print_flag
[0x080487c1]> pdf
┌ (fcn) sym.print_flag 43
│   sym.print_flag ();
│           ; var int local_ch @ ebp-0xc
│           ; CALL XREF from sym.main (0x8048802)
│           0x080487c1      55              push ebp                    ; ./print_flag.c:90
│           0x080487c2      89e5            ebp = esp
│           0x080487c4      83ec18          esp -= 0x18
│           0x080487c7      c745f4390500.   dword [local_ch] = 0x539    ; ./print_flag.c:91 ;
│           0x080487ce      8b45f4          eax = dword [local_ch]      ; ./print_flag.c:92
│           0x080487d1      8b048580a004.   eax = dword [eax*4 + obj.strs] ; [0x804a080:4]=0
│           0x080487d8      83ec08          esp -= 8
│           0x080487db      50              push eax
│           0x080487dc      684c890408      push str.your_flag_is:_picoCTF__s ; 0x804894c ; "y
│           0x080487e1      e82afcffff      sym.imp.printf ()           ; int printf(const cha
│           0x080487e6      83c410          esp += 0x10
│           0x080487e9      90                                          ; ./print_flag.c:93
```

```
|           0x080487ea      c9             leave
└           0x080487eb      c3             return
```

As you can see, the flag pointer is located at `eax*4 + obj.strs` or `0x804a080+0x539*4` in memory:

```
❯ python
>>> hex(0x804a080+0x539*4)
'0x804b564'
```

Now, we can use gdb and the core file to restore the application state and extract the flag from that address:

```
$ gdb ./print_flag ./core
...
gef➢  x 0x804b564
0x804b564 <strs+5348>:  0x080610f0
gef➢  x 0x080610f0
0x80610f0:      "e52f4714963eb207ae54fd424ce3c7d4"
```

flag: `picoCTF{e52f4714963eb207ae54fd424ce3c7d4}`

Malware Shops

Problem

There has been some malware detected, can you help with the analysis? More info here.

Connect with `nc 2018shell2.picoctf.com 46168`.

Solution

Just read the graph and do this problem by hand.

```
❯ nc 2018shell2.picoctf.com 46168
You'll need to consult the file `clusters.png` to answer the following questions.


How many attackers created the malware in this dataset?
5
Correct!


In the following sample of files from the larger dataset, which file was made by the same att
       hash  jmp_count  add_count
0  3ce8eb6f       33.0       28.0
1  55489271       40.0        2.0
2  33d91680       39.0       29.0
3  ebaf5ccd        9.0       17.0
4  e9c0ac07       17.0       61.0
5  628e79cf        9.0       18.0
6  b3ae7861       41.0       10.0
7  cc251d4b       16.0       41.0
8  0c91a83b       17.0       65.0
9  97a0fc46       10.0       38.0
33d91680
Correct!


Great job. You've earned the flag: picoCTF{w4y_0ut_dea1794b}
```

flag: `picoCTF{w4y_0ut_dea1794b}`

LoadSomeBits

Problem

Can you find the flag encoded inside this image? You can also find the file in /problems/loadsomebits_2_c5bba4da53a839fcdda89e5203ac44d0 on the shell server.

Solution

Ryan Jung on our team solved this challenge. It is about looking at the least significant bit of each pixel value.

flag: `picoCTF{st0r3d_iN_th3_l345t_s1gn1f1c4nT_b1t5_2705826400}`

> Feel free to leave a comment if any of the challenges is not well explained.

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

ctflover • a year ago

hi,

i did not understand how you extracted lsb in LoadSomeBits challenge.
can you plz explain it.

• Reply • Share ›

ALSO ON GITHUB-BLOG

PicoCTF 2018 Writeup: Reversing

7 comments • a year ago

Avatar Ye Thu — thanks

PicoCTF 2018 Writeup: Binary Exploitation

8 comments • a year ago

Avatar Roee Sefi — U welcome

35c3ctf 2018 Writeup · Alan's Blog

1 comment • 10 months ago

Avatar teamrocketist — Looks like I "cheated" on the juggle challenge :O, the random was weird and allowed me guess the next

PicoCTF 2019 Writeup: Binary Exploitation

5 comments • 8 days ago

Avatar Nev River — Hello Alan, thank you very much for your write-ups, they had helped me a lot.For the NewOverFlow-1 problem,

Subscribe Add Disqus to your site Disqus' Privacy Policy DISQUS