porpeeranut / CTF



porpeeranut fix seccon writeup Latest commit 77772b3 on Jan 25

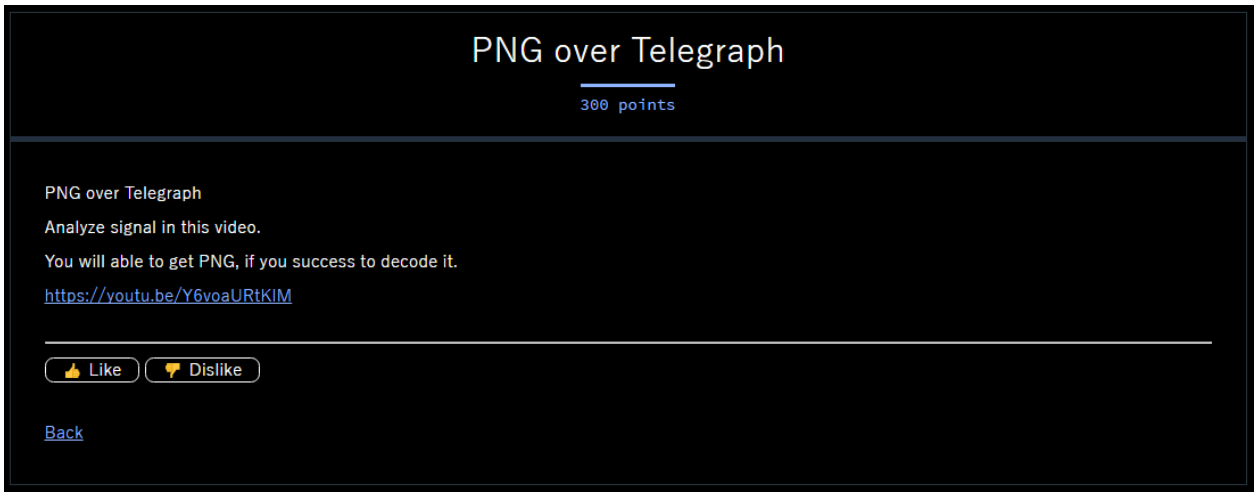
..

| | | |
|---|---|---|
| image | add seccon | 4 months ago |
| Crypto - PNG over Telegraph 300 pts.py | add seccon | 4 months ago |
| README.md | fix seccon writeup | 4 months ago |

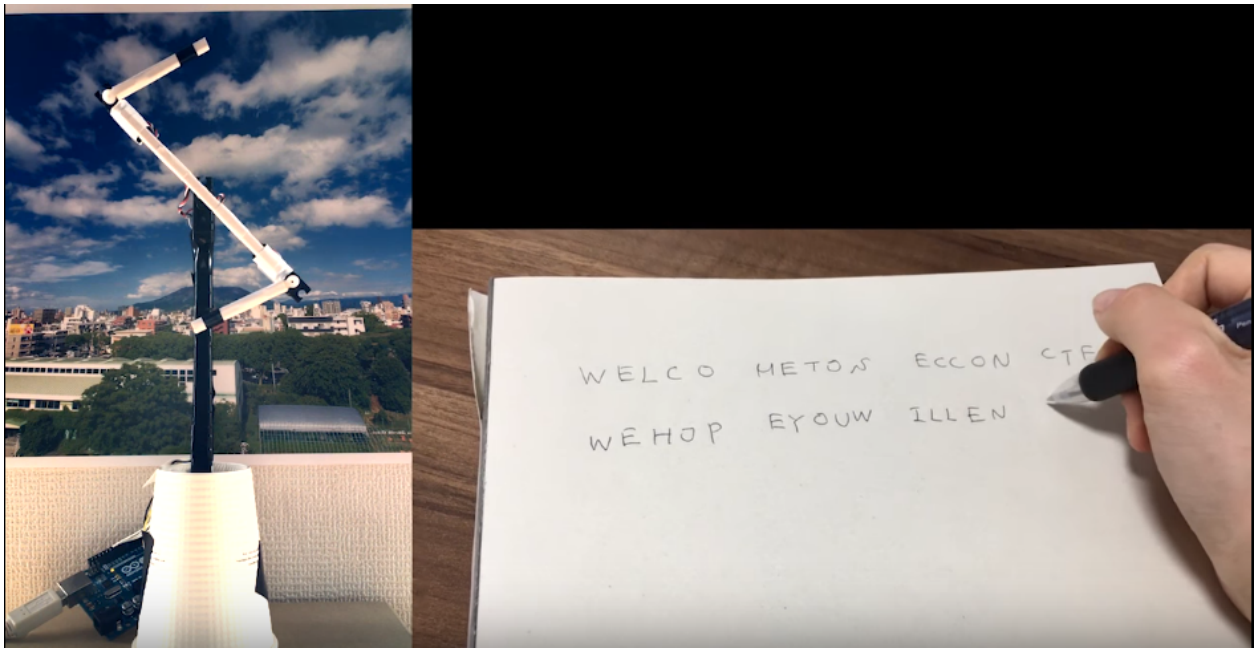
README.md

# SECCON 2016 Online CTF

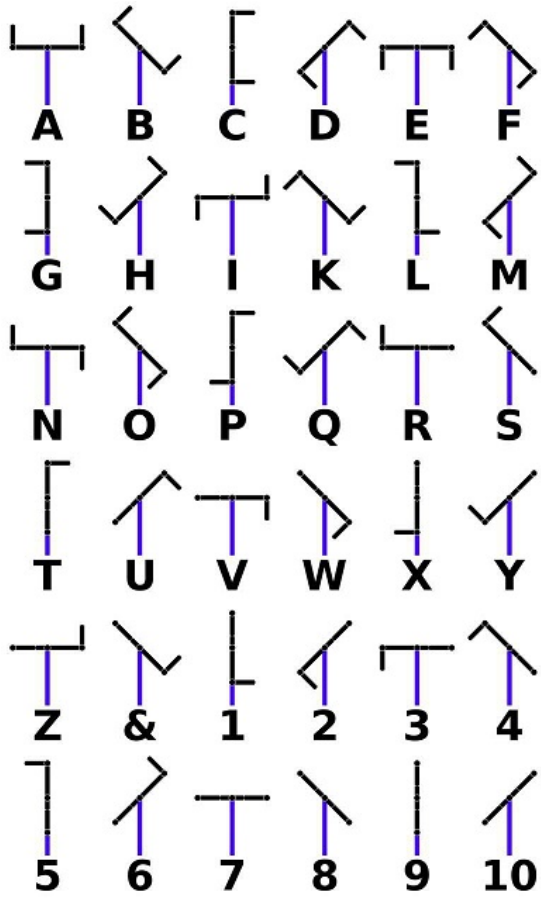
##Description **Category:** Crypto **Points:** 300 **Solved:** 14




##Solution To solve this challenge, we need to know how to decode tower's posture in video into message. First, I clicked the link from the challenge to see an interesting video. At the beginning of the video we'll see someone show how to decode signal tower to message.




He'll decode just some character and we have to find out what kind of communication system for conveying a message. After awhile, my team mate did google and found it's a 'Sémaphore'.




Next step, we have to download vdo and try to extract frames. I used 'ffmpeg' to extract frames every 1 second. After trying several times, I found that 00:01:15.790 was the best start time for the first frame.

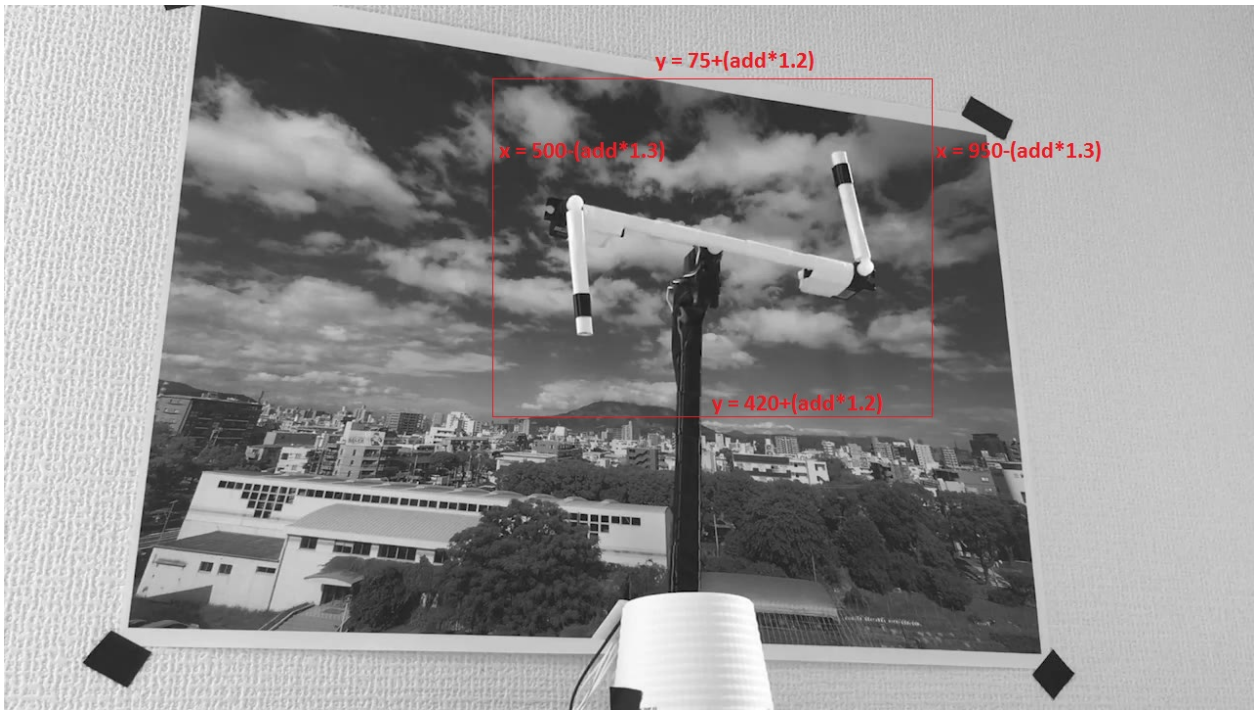
```
ffmpeg -i seccontower.mp4 -ss 00:01:15.790 -r 1 out/img%04d.jpg
```

Then I got all frames(2992 frames).

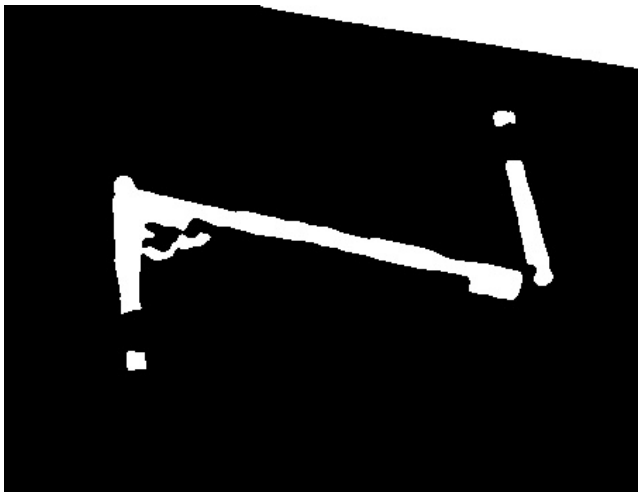

After that, I selected image which would be a training set for each posture (3 images in range 300, 1300, 2300 per posture) and tried to decode image by comparing hash value with library 'imagehash' in python.

Unfortunately, there're too many error in result. I replayed the vdo with very fast speed and noticed that the camera was panning up and light had a bit changed when recording.

So I had to do preprocess for better result by croping image as below.

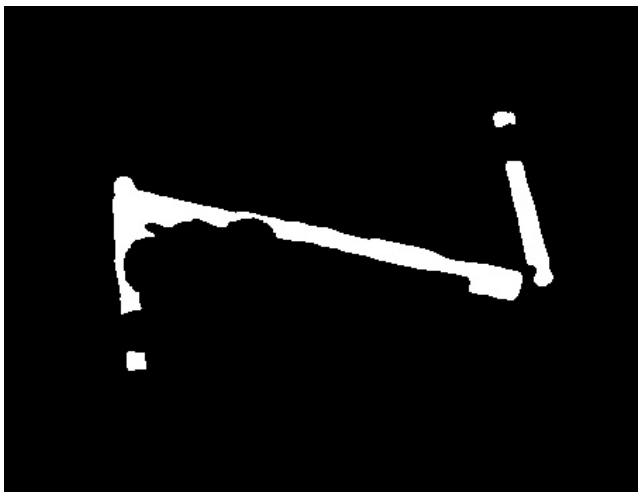



'add' was a variable which will be added by 1 every 200 frames. Then I did a Gaussian Blur and made it to binary image by threshold = 185-add*1.6



I created a below image.

Then combined with previous image and made it to binary image again by threshold = 255.



It looks pretty good.

Then I tried to decode again and grouped each image into folder to check error more easily.

Actually, it could be done here. We can decode result with Base32, write it to png file and open it on Windows(Linux does not work). We will get flag in QR code image. Or encode with Base64, put it in Image tag, save it to html file and open it with Chrome(Firefox does not work).

It seems like Windows and Chrome try to show QR code image even 'Check value' for zlib or CRC for IDAT chunk are incorrect.

I checked png file with pngcheck and found that CRC for IDAT chunk is incorrect (Assume that CRC data(0xD912FF2C) after IDAT chunk data is correct).

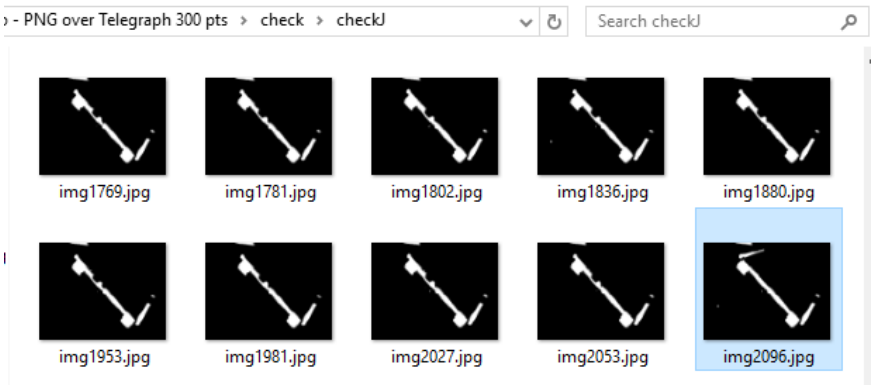
```
root@kali:/tmp/seccon# pngcheck -fv flag.png
File: flag.png (1862 bytes)
  chunk IHDR at offset 0x0000c, length 13
    730 x 730 image, 1-bit palette, non-interlaced
  chunk PLTE at offset 0x00025, length 6: 2 palette entries
  chunk tRNS at offset 0x00037, length 2: 2 transparency entries
  chunk pHYs at offset 0x00045, length 9: 2834x2834 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0005a, length 1752
    zlib: deflated, 32K window, default compression
  CRC error in chunk IDAT (computed 9ca0183e, expected d912ff2c)
  chunk IEND at offset 0x0073e, length 0
ERRORS DETECTED in flag.png
root@kali:/tmp/seccon#
```

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
00000010   00 00 02 DA 00 00 02 DA 01 03 00 00 00 35 20 66  ...Ú...Ú.....5 f
00000020   70 00 00 00 06 50 4C 54 45 00 00 00 FF FF FF A5  p....PLTE...ÿÿÿ¥
00000030   D9 9F DD 00 00 00 02 74 52 4E 53 FF FF C8 B5 DF  ÙŸÝ....tRNSÿÿÈµß
00000040   C7 00 00 00 09 70 48 59 73 00 00 0B 12 00 00 0B  Ç....pHYs.......
00000050   12 01 D2 DD 7E FC 00 00 06 D8 49 44 41 54 78 9C  ..ÒÝ~ü...ØIDATxœ
00000060   ED DA 41 92 F2 3A 0C 46 51 55 65 01 2C 29 5B EF  íÚA'ò:.FQUe.,)[ï
00000070   25 B1 00 AA F4 1A 4B 9F A4 00 6F F2 E3 9E 5D 0F  %±.ªô.KŸ¤.oòã].
00000080   68 48 E2 93 4C 64 C9 4A 9B FF E1 30 70 70 70 70  hHâ"LdÉJ›ÿá0pppp
00000090   70 70 70 70 70 70 70 70 70 F0 4F B8 E5 38 DC 7F  ppppppppðO¸å8Ü.
                                ...snip...
000006D0   48 2F 28 B2 AF 8E 45 E1 5C 51 6E D5 55 8E A2 3B  H/(²¯ŽEá\QnÕUŽ¢;
000006E0   D6 00 6D 77 C1 C1 C1 B7 E1 EB 4F 14 C9 BA 20 6A  Ö.mwÁÁÁ·áëO.É° j
000006F0   E1 CB 31 BD 62 5D 67 45 9E B5 1A D4 89 97 F0 07  áË1½b]gEžµ.Ô‰—ð.
00000700   07 07 FF 0E CF 51 C1 BC BE 99 E2 FD 1C BB 5F B9  ..ÿ.ÏQÁ¼¾™âý.»_¹
00000710   B3 70 CE 02 7B 95 DA 59 6F 83 83 83 EF C3 FF 68  ³pÎ.{•ÚYoƒƒƒïÃÿh
00000720   80 83 83 83 83 83 83 83 83 83 83 83 83 83 BF 8C  €ƒƒƒƒƒƒƒƒƒƒƒƒƒ¿Œ
00000730   FF 00 F0 51 32 B9 D9 12 FF 2C 00 00 00 00 49 45  ÿ.ðQ2¹Ù.ÿ,....IE
00000740   4E 44 AE 42 60 82                              ND®B`‚

           Chunk Length Chunk Type Chunk Data zlib check value CRC
```

So I looked at each folder and found only 1 error at group 'J' (img2096.jpg) that should be decodeed to 'B'.




Then I changed code to fix this error and tried to decode again. PNG was correct now. After that I read png and decoded QR code with 'zbarimg' and got a flag. That's all the process.

```
root@kali:/tmp/seccon# pngcheck -fv flag.png
File: flag.png (1862 bytes)
  chunk IHDR at offset 0x0000c, length 13
    730 x 730 image, 1-bit palette, non-interlaced
  chunk PLTE at offset 0x00025, length 6: 2 palette entries
  chunk tRNS at offset 0x00037, length 2: 2 transparency entries
  chunk pHYs at offset 0x00045, length 9: 2834x2834 pixels/meter (72 dpi)
  chunk IDAT at offset 0x0005a, length 1752
    zlib: deflated, 32K window, default compression
  chunk IEND at offset 0x0073e, length 0
No errors detected in flag.png (6 chunks, 97.2% compression).
root@kali:/tmp/seccon#
```

```
QR-Code:Congratulations!

The flag is; SECCON{SEMAPHORE_LINE_IS_THE_1ST_TELEGRAPH_SYSTEM_IN_THE_WORLD}

Miniature model; @9sq
Base idea and software; @kikuchan98
```

Reference https://www.w3.org/TR/2003/REC-PNG-20031110/#13Decoders.Errors
http://www.libpng.org/pub/png/spec/1.2/png-1.2.pdf