Hacky Easter 2019 Solution

h3k

h3k hackyeaster 19 solution

This is my partial writeup of HE19. Thanks to all the creators of the challenges!

Teaser

Use virtualdub to convert to frames [1].

Possibly LSB encoded

Colours

```
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (183, 182, 219))]
[(10000, (183, 182, 219))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (182, 177, 222))]
[(10000, (182, 177, 222))]
[(10000, (187, 189, 222))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (172, 138, 206))]
[(10000, (172, 138, 206))]
[(10000, (195, 200, 228))]
[(10000, (0, 0, 0))]
[(10000, (0, 0, 0))]
[(10000, (183, 181, 221))]
[(10000, (0, 0, 0))]
[(10000, (177, 179, 206))]
[(10000, (0, 0, 0))]
[(10000, (245, 248, 245))]
[(10000, (181, 170, 222))]
[(10000, (65, 60, 60))]
[(10000, (182, 177, 222))]
[(10000, (182, 177, 222))]
[(10000, (196, 201, 229))]
[(10000, (0, 0, 0))]
[(10000, (221, 210, 237))]
[(10000, (170, 134, 203))]
[(10000, (65, 60, 60))]
[(10000, (65, 60, 60))]
[(10000, (0, 0, 0))]
[(10000, (51, 54, 50))]
[(10000, (51, 54, 50))]
[(10000, (191, 193, 224))]
[(10000, (182, 175, 218))]
[(10000, (0, 0, 0))]
[(10000, (184, 184, 223))]
[(10000, (0, 0, 0))]
[(10000, (191, 193, 224))]
[(10000, (148, 136, 181))]
[(10000, (245, 248, 245))]
[(10000, (245, 248, 245))]
[(10000, (0, 0, 0))]
[(10000, (252, 255, 252))]
[(10000, (252, 255, 252))]
[(10000, (49, 53, 56))]
[(10000, (178, 164, 219))]
[(10000, (185, 180, 223))]
[(10000, (215, 218, 215))]
[(10000, (0, 0, 0))]
[(10000, (252, 255, 252))]
[(10000, (252, 255, 252))]
[(10000, (182, 180, 219))]
[(10000, (168, 130, 199))]
```

```
[(10000, (170, 136, 204))]
[(10000, (170, 136, 204))]
[(10000, (0, 0, 0))]
[(10000, (42, 44, 41))]
[(10000, (137, 106, 159))]
[(10000, (172, 136, 205))]
[(10000, (49, 53, 56))]
[(10000, (0, 0, 0))]
[(10000, (252, 255, 252))]
[(10000, (0, 0, 0))]
[(10000, (94, 97, 93))]
[(10000, (141, 126, 177))]
[(10000, (179, 168, 219))]
[(10000, (252, 255, 252))]
[(10000, (0, 0, 0))]
[(10000, (42, 44, 41))]
[(10000, (0, 0, 0))]
[(10000, (252, 255, 252))]
[(10000, (252, 255, 252))]
[(10000, (177, 161, 217))]
[(10000, (49, 53, 56))]
[(10000, (49, 53, 56))]
[(10000, (252, 255, 252))]
[(10000, (252, 255, 252))]
[(10000, (169, 128, 196))]
[(10000, (138, 122, 175))]
[(10000, (65, 60, 60))]
[(10000, (136, 102, 157))]
[(10000, (136, 102, 157))]
[(10000, (173, 153, 217))]
[(10000, (173, 153, 217))]
[(10000, (174, 155, 218))]
[(10000, (135, 111, 158))]
[(10000, (135, 111, 158))]
[(10000, (141, 126, 179))]
[(10000, (0, 0, 0))]
[(10000, (140, 124, 177))]
[(10000, (0, 0, 0))]
[(10000, (138, 122, 175))]
[(10000, (138, 122, 175))]
[(10000, (0, 0, 0))]
[(10000, (138, 120, 176))]
[(10000, (0, 0, 0))]
[(10000, (136, 102, 157))]
[(10000, (135, 104, 155))]
[(10000, (0, 0, 0))]
[(10000, (65, 60, 60))]
[(10000, (0, 0, 0))]
[(10000, (65, 60, 60))]
```

We seem to be missing some frames with virtualdub.

```
Format                                   : MPEG-4
Format profile                           : Base Media
Codec ID                                 : isom (isom/iso2/mp41)
File size                                : 9.48 MiB
Duration                                 : 5 s 0 ms
Overall bit rate mode                    : Variable
Overall bit rate                         : 15.9 Mb/s
Writing application                      : Lavf58.17.101

Video
ID                                       : 1
Format                                   : MPEG-4 Visual
Format profile                           : Simple@L1
Format settings, BVOP                    : No
Format settings, QPel                    : No
Format settings, GMC                     : No warppoints
Format settings, Matrix                  : Default (H.263)
Codec ID                                 : mp4v-20
Duration                                 : 5 s 0 ms
Bit rate mode                            : Variable
Bit rate                                 : 14.2 Mb/s
Maximum bit rate                         : 922 Mb/s
Width                                    : 100 pixels
Height                                   : 100 pixels
Display aspect ratio                     : 1.000
Frame rate mode                          : Constant
Frame rate                               : 46 080.000 FPS
Color space                              : YUV
Chroma subsampling                       : 4:2:0
Bit depth                                : 8 bits
Scan type                                : Progressive
Compression mode                         : Lossy
Bits/(Pixel*Frame)                       : 0.031
Stream size                              : 8.46 MiB (89%)
Writing library                          : Lavc58.21.104
```

Unterschied bei Extraction mit oder ohne -vsync 2 (andere Bilder sind “falsch”). Falsche Bilder haben nicht immer die gleiche Farbe. 9600 eine und 400 Pixel die andere Farbe. Nur zusammenhängende Bereiche. Total 46080 FPS * 5s = 230400 Bilder 2254 verschiedene Farben

http://download.tsi.telecom-paristech.fr/gpac/mp4box.js/filereader.html https://github.com/adrianlopezroche/fdupes https://imageio.readthedocs.io/en/stable/index.html https://imageio.readthedocs.io/en/stable/examples.html https://stackoverflow.com/questions/29718238/how-to-read-mp4-video-to-be-processed-by-scikit-image

ffmpeg -i he2019_teaser.mp4 ffframe-%08d.png ffmpeg -skip_frame nokey -i he2019_teaser.mp4 ffframe_nokey%06d.png

Pink Noise -> Could be music

[1] https://www.raymond.cc/blog/extract-video-frames-to-images-using-vlc-media-player/

01 - Twisted

Use gimp whirl and pinch tool. Set whirl to 116, leave pinch at zero and radius at one.

02 JustWatch

convert -coalesce justWatch.gif out%05d.gif

0: g 1: i 2: k / 2 3: e 4: m 5: e 6: a 7: s 8: i 9: g 10: n

gikemeasign -> givemeasign

he19-DwWd-aUU2-yVhE-SbaG

03 - Sloppy Encryption

The easterbunny is not advanced at doing math and also really sloppy.

He lost the encryption script while hiding your challenge. Can you decrypt it?

```
K7sAYzGlYxokZyXIIPrXxK22DkU4Q+rTGfUk9i9vA60C/ZcQOSWNfJLTu4RpIBy/27yK5CBW+UrBhm0=
```

```
require"base64"
puts"write some text and hit enter:"
input=gets.chomp
h=input.unpack('C'*input.length).collect{|x|x.to_s(16)}.join
ox='%#X'%h.to_i(16)
x=ox.to_i(16)*['5'].cycle(101).to_a.join.to_i
c=x.to_s(16).scan(/../).map(&:hex).map(&:chr).join
b=Base64.encode64(c)
puts"encrypted text:""#{b}"
```

Solution

Unsloppify ruby code.

```
from base64 import b64decode
import binascii
input = " K7sAYzGlYx0kZyXIIPrXxK22DkU4Q+rTGfUk9i9vA60C/ZcQOSWNfJLTu4RpIBy/27yK5CBW+UrBhm0="

decoded = b64decode(input)
print(decoded)
print(decoded.hex())
intrepr = int(decoded.hex(), 16)
print(intrepr)
baseint = intrepr // int(''.join(['5'] * 101))
print(hex(baseint))
print(hex(baseint)[2:])
superbase = hex(baseint)[2:]
if len(superbase) % 2 != 0:
    superbase += '0'
print(binascii.unhexlify(superbase))
```

```
b'n00b_style_crypto'
```

Flag: he19-YPkZ-ZZpf-nbYt-6ZyD

04 - Disco 2

This year, we dance outside, yeaahh! See here.

Solution

Some kind of modem? Mp3stego didn't work. View it in sonic visualizer

05 - Call for Papers

Please read and review my CFP document, for the upcoming IAPLI Symposium.

I didn't write it myself, but used some artificial intelligence.

What do you think about it?

Solution

Look at document in word. Author SCIpher. Use decoder from https://pdos.csail.mit.edu/archive/scigen/scipher.html (https://pdos.csail.mit.edu/archive/scigen/scipher.html)

https://hackyeaster.hacking-lab.com/hackyeaster/images/eggs/5e171aa074f390965a12fdc240.png

06 - Dots

Uncover the dots' secret!

Then enter the password in the egg-o-matic below. Uppercase only, and no spaces!

Solution

By just looking at the black points I can decipher HELL - O - UC - ???

HELLOUC

07 - Shell we Argument

Let's see if you have the right arguments to get the egg.

Solution

Use sh -x to trace function output.

```
/bin/sh -x eggi.sh
[...]
```

Use last part to reconstruct URL:

```
if [[ $match -eq 5 ]]; then
t="$Az$Bz$Cz$Dz$Ez$Fz$Gz$Hz$Iz$Jz$Ez$Fz$Kz$Lz$Mz$Nz$Oz$Pz$Ez$Fz$Gz$Hz$Iz$Qz$Rz$Sz$Tz$Uz$Vz$Wz$Xz$Yz$Zz$az$bz$cz$dz$e
echo "Great, that are the perfect arguments. It took some time, but I'm glad, you see it now, too!"
sleep 2
if b x-www-browser ; then
x-www-browser $t
else
echo "Find your egg at $t"
fi
else
echo "I'm not really happy with your arguments. I'm still not convinced that those are reasonable statements..."
echo "low: $low, matched $match, high: $high"
fi
+ z=

+ Cz=s:
+ qz=.p
+ fz=8a
+ az=e9
+ Oz=co
+ Xz=a6
+ hz=7e
+ Rz=im
+ Bz=tp
+ lz=62
+ Kz=in
+ Wz=s/
+ rz=ng
+ Yz=1e
+ Jz=r.
+ Iz=te
+ Tz=es
+ Zz=f3
+ kz=15
+ Az=ht
+ Fz=ck
+ Uz=/e
+ Sz=ag
+ Lz=g-
+ Ez=ha
+ Vz=gg
+ Pz=m/
+ pz=8c
+ Gz=ye
+ Dz=//
+ iz=cd
+ Hz=as
+ Mz=la
+ Nz=b.
+ nz=c7
+ Qz=r/
+ ez=d8
+ cz=ac
+ gz=12
+ bz=75
+ oz=4a
+ mz=42
+ jz=6e
+ dz=b7
```

Set the variables and run the url generation:

```
Cz=s:
qz=.p
fz=8a
az=e9
Oz=co
Xz=a6
hz=7e
Rz=im
Bz=tp
lz=62
Kz=in
Wz=s/
rz=ng
Yz=1e
Jz=r.
Iz=te
Tz=es
Zz=f3
kz=15
Az=ht
Fz=ck
Uz=/e
Sz=ag
Lz=g-
Ez=ha
Vz=gg
Pz=m/
pz=8c
Gz=ye
Dz=//
iz=cd
Hz=as
Mz=la
Nz=b.
nz=c7
Qz=r/
ez=d8
cz=ac
gz=12
bz=75
oz=4a
mz=42
jz=6e
dz=b7
echo
  $Az$Bz$Cz$Dz$Ez$Fz$Gz$Hz$Iz$Jz$Ez$Fz$Kz$Lz$Mz$Nz$Oz$Pz$Ez$Fz$Gz$Hz$Iz$Qz$Rz$Sz$Tz$Uz$Vz$Wz$Xz$Yz$Zz$az$bz$cz$dz$ez$
> https://hackyeaster.hacking-lab.com/hackyeaster/images/eggs/a61ef3e975acb7d88a127ecd6e156242c74af38c.png
```

It seems however that the challenge can't be solved with `/bin/sh`. However with bash it works.

```
bash eggi.sh -R 465 -a 333 -b 911 -I 112 -t 007
```

Flag: he19-Bxvs-Vno1-9l9D-49gX

08 - Modern Art

Do you like modern art?

Solution

Replace the little qr codes with the correct part of a qr code. (Edge ones say “remove me”)

```
Isn't that a bit too easy?
```

At the end of the jpg file there is an UTF-8 string resembling another QR-Code “AES-128”. Also it has two jpg end tags, the data in between seems almost random. However it contains things such as:

```
(E7EF085CEBFCE8ED93410ACF169B226A)
(KEY=1857304593749584)
```

Try to decrypt AES-128, use KEY*2 as it is too short

09 - rorriM rorriM

Mirror, mirror, on the wall, who’s the fairest of them all?

Solution

evihcra.piz

evihcra.piz -> archive.zip Reverse whole file to get content.

90gge.gnp

Just reverse GNP to PNG, now we can read the file and scan the QR code in it. (If your QR decoder can’t scan it, apply negative and flip vertical)

he19-VFTD-kVos-DeL1-lATA

10 - Stackunderflow

Check out this new Q&A site. They must be hiding something but we don’t know where to search.

http://whale.hacking-lab.com:3371/ (http://whale.hacking-lab.com:3371/)

Solution

According to one question, this page uses a nosql database. While looking at the links I see this pattern:

```
questions/5cb581c877c89400158f585e
questions/5cb581c877c89400158f585f
questions/5cb581c877c89400158f5863
questions/5cb581c877c89400158f5860
questions/5cb581c877c89400158f5861
questions/5cb581c877c89400158f5862
```

Only the last two characters change in our env.

They seem to be NoSQL object ids https://docs.mongodb.com/manual/reference/method/ObjectId/ (https://docs.mongodb.com/manual/reference/method/ObjectId/)

We have 96 bits (12 bytes) to brute force. However this seems unreasonable. The incrementing part is only 3 bytes (in the end).

However it is still 24 bits to search. The title leads to underflow, which reminds me of the unix timestamp overflow that will happen in 2038.

11 - Memeory 2.0

We improved Memeory 1.0 and added an insane serverside component. So, no more CSS-tricks. Muahahaha.

Flagbounty for everyone who can solve 10 successive rounds. Time per round is 30 seconds and only 3 missclicks are allowed.

Good game.

Solution

We can still see the images, however we have to automate the clicks due to time constraints.

```
$(".boxBack").remove()
```

- SessionID defines game state
- Solution is sent by POST to /solve with params first:id, second:id. These IDs start with zero, pictures start with 1.
    - This seems to be an error in the code.

I first checked if the server sends the same bytes for the same image. Due to this fact I build a dictionary with the hash of the image as key and the image ids as value.

```
import requests
import shutil
from hashlib import md5

baseurl = 'http://whale.hacking-lab.com:1111/'

picpart = 'pic/'

s = requests.session()

def getSession():
    return s.get(baseurl)

def getPictureUrl(n):
    return baseurl + picpart + str(n)

def savePicturesToDisk():
    for i in range (1,98 + 1):
        #print(i, getPictureUrl(i))
        res = s.get(getPictureUrl(i), stream=True)
        with open('img' + str(i) + '.png', 'wb') as out_file:
            shutil.copyfileobj(res.raw, out_file)
        #print(i, res.status_code)

def addImageToDict(image : bytes, index: int, myDict : dict):
    x = md5()
    x.update(image)
    if x.hexdigest() in myDict:
        myDict[x.hexdigest()].append(index)
    else:
        myDict[x.hexdigest()] = [index]

def sendSolution(a: int, b: int):
    res = s.post(baseurl + 'solve', data={'first': a, 'second': b})
    print(res.content)
    return res

def main():
    for _ in range(10):
        getSession()
        memoryDict = {}
        for i in range(1, 98 + 1):
            res = s.get(getPictureUrl(i), stream=True)
            addImageToDict(res.content, i, memoryDict)
        print(memoryDict)
        for _,(a,b) in memoryDict.items():
            sendSolution(a,b)

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

Solution: b'ok, here is your flag: 1-m3m3-4-d4y-k33p5-7h3-d0c70r-4w4y'

Flag: he19-jaQ9-0NIr-Ladc-brOT

12 - Decryptor

Crack the might Decryptor and make it write a text with a flag.

No Easter egg here. Enter the flag directly on the flag page.

Solution

ELF executable file. Aks for password, gives output. IDA shows a hash function that is run. One constant seems to be 0x10325476, which is also used in SHA1. Calls fgets with maximum 16 chars (including \0 byte) but only allocates 4 bytes for the destination. Possible that we have to bufferoverflow for the solution. It seems that the hash constants just get added and subtracted again. Using the same characters repeatedly doesn't change the output. The output only changes at the key-input position. This smells like XOR.

To verfiy XOR-hypothesis I XOR the first data with my input `aaaa` and verify that I get the same result as with the program. Data is at 0000000000601060. Extct data using IDA, fill missing zeroes and swap endianness.

XOR Decrypt try one with https://github.com/Alamot/code-snippets/blob/master/crypto/xorknown.py (https://github.com/Alamot/code-snippets/blob/master/crypto/xorknown.py)

```
hacker@kali:~/Documents/easter19/12$ python xorknown.py encrypted.bin he19-
Searching XOR-encrypted encrypted.bin for string 'he19-' (max_key_length = 20)
Key length: 13
Partial Key: ������1th_n���
Plaintext: ������,
co�����������ou fo�����������hidde�����������he19-�����������J-3dy�����������

'Th�����������erato�����������remel����������  as a�����������nt in�����������mplex�����������. By
  ����������using�����������ant r�����������  key,�����������e XOR�����������can t�����������  be
  b�����������ing f�����������  anal�����������  the ����������of an�����������e can�����������sed
  o�����������ise k�����������n the�����������  be r�����������'
(ht�����������.wiki�����������g/wik�����������pher)�����������R gat�����������t can�����������
  from�����������ND ga�����������fact,�����������ND an�����������tes a�����������lled ����������al
  ga�����������  any ����������funct�����������be co�����������d fro�����������  NAND�����������r
  NOR�����������lone.�����������four ����������es ar�����������ed by�����������es, t�����������lts
  i�����������R gat�����������  can ����������rted ����������R gat�����������ertin�����������tput
  ����������f the�����������(e.g.�����������fifth�����������e).'
  ����������//en.�����������a.org�����������R_gat�������
```

This looks intersting. Use https://github.com/ThomasHabets/xor-analyze (https://github.com/ThomasHabets/xor-analyze) with default frequency table to give a best guess.

```
hacker@kali:~/Documents/easter19/12/xor-analyze$ ./xor-analyze -k 26 ../encrypted.bin freq/linux-2.2.14-int-
  m0.freq
xor-analyze version 0.4 by Thomas Habets <thomas@habets.pp.se>
Finding key based on byte frequency... 26 / 26
Checking redundancy... 100.00 %
Probable key: "r0rYw1thYn4nni0rUw1thXn4nd"
```

Tune the output by looking at the key and fixing some chars manually. Also the guessed keylength was wrong. XorKey: x0r_w1th_n4nd

```
Hello,
congrats you found the hidden flag: he19-Ehvs-yuyJ-3dyS-bN8U.
[...]
```

Flag: he19-Ehvs-yuyJ-3dyS-bN8U

13 - Symphony in HEX

A lost symphony of the genius has reappeared.

symphony-image

Hint: count quavers, read semibreves

Once you found the solution, enter it in the egg-o-matic below. Uppercase only, and no spaces!

Solution

Quaver: ♪ (also rest) Semibreve: 𝅝 (also rest)

Number of quavers: 27

8R 2R 0R 0R 1R 15

14 - White Box

Do you know the mighty WhiteBox encryption tool? Decrypt the following cipher text!

9771a6a9aea773a93edc1b9e82b745030b770f8f992d0e45d7404f1d6533f9df348dbccd71034aff88afd188007df4a5c844969584b5ffd6ed2eb92aa419914e

Solution

WhiteBox ELF executable. Input plaintext, output ciphertext. Cipertext is 128 characters long -> Input should be between 3*16 and 4*16-1 characters long (48 - 63)

WhiteBox Cryptography means that the crypto primitives are hidden in the program. In theory I shouldn't be able to find the key of the cipher in ram. It looks like a 16 byte block cipher (repeating plaintext gives repeating ciphertext). It also seems to use padding in the last block.

Table 1: .data:0000000000603060 Table 2: .data:0000000000602060

Use solution based on https://github.com/ResultsMayVary/ctf/tree/master/RHME3/whitebox (https://github.com/ResultsMayVary/ctf/tree/master/RHME3/whitebox)

```
(.venv) hacker@kali:~/Documents/easter19/14$ python3 dfait.py
Press Ctrl+C to interrupt
Send SIGUSR1 to dump intermediate results file: $ kill -SIGUSR1 12245
Lvl 011 [0x0001F144-0x0001F145[ xor 0x0F 74657374746573747465737474657374 -> 8CDDDF1521A954FD6161E2212F0237D4
  GoodEncFault Column:0 Logged
Lvl 011 [0x0001F144-0x0001F145[ xor 0x1C 74657374746573747465737474657374 -> E8DDDF1521A954E76161D4212F1A37D4
  GoodEncFault Column:0 Logged
Lvl 011 [0x0001F144-0x0001F145[ xor 0x14 74657374746573747465737474657374 -> 12DDDF1521A95499616169212FC237D4
  GoodEncFault Column:0 Logged
Lvl 011 [0x0001F144-0x0001F145[ xor 0x1B 74657374746573747465737474657374 -> 2BDDDF1521A9545961618A212FA237D4
  GoodEncFault Column:0 Logged
Lvl 011 [0x0001F145-0x0001F146[ xor 0x57 74657374746573747465737474657374 -> 8DDDDF6E21A92D5561FACA21C6D237D4
  GoodEncFault Column:3 Logged
Lvl 011 [0x0001F145-0x0001F146[ xor 0x65 74657374746573747465737474657374 -> 8DDDDF6C21A9E455617ECA215ED237D4
  GoodEncFault Column:3 Logged
Lvl 011 [0x0001F145-0x0001F146[ xor 0xF8 74657374746573747465737474657374 -> 8DDDDF1F21A9BF5561D3CA219BD237D4
  GoodEncFault Column:3 Logged
Lvl 011 [0x0001F145-0x0001F146[ xor 0x8D 74657374746573747465737474657374 -> 8DDDDF2E21A9C155613BCA2108D237D4
  GoodEncFault Column:3 Logged
Lvl 011 [0x0001F146-0x0001F147[ xor 0xBF 74657374746573747465737474657374 -> 8DDD7815213754551961CA212FD23799
  GoodEncFault Column:2 Logged
Lvl 011 [0x0001F146-0x0001F147[ xor 0x24 74657374746573747465737474657374 -> 8DDDEF15214554553661CA212FD237FB
  GoodEncFault Column:2 Logged
Lvl 011 [0x0001F146-0x0001F147[ xor 0xA1 74657374746573747465737474657374 -> 8DDD811521835455EA61CA212FD23714
  GoodEncFault Column:2 Logged
Lvl 011 [0x0001F146-0x0001F147[ xor 0x49 74657374746573747465737474657374 -> 8DDD221521D854559A61CA212FD237F9
  GoodEncFault Column:2 Logged
Lvl 011 [0x0001F147-0x0001F148[ xor 0x67 74657374746573747465737474657374 -> 8D28DF1560A954556161CAF62FD285D4
  GoodEncFault Column:1 Logged
Lvl 011 [0x0001F147-0x0001F148[ xor 0xA6 74657374746573747465737474657374 -> 8D6CDF15FEA954556161CAA82FD266D4
  GoodEncFault Column:1 Logged
Lvl 011 [0x0001F147-0x0001F148[ xor 0x95 74657374746573747465737474657374 -> 8D59DF1541A954556161CA222FD2F5D4
  GoodEncFault Column:1 Logged
Lvl 011 [0x0001F147-0x0001F148[ xor 0xDA 74657374746573747465737474657374 -> 8D79DF158BA954556161CA262FD2E5D4
  GoodEncFault Column:1 Logged
Saving 17 traces in dfa_enc_20190422_205937-205951_17.txt
Last round key #N found:
FD83DB41AC158393CC291088B76F201A
```

Tool: https://github.com/ResultsMayVary/ctf/raw/master/RHME3/whitebox/inverse_aes.py (https://github.com/ResultsMayVary/ctf/raw/master/RHME3/whitebox/inverse_aes.py)

```
hacker@kali:~/Documents/easter19/14$ python inverse_aes.py FD83DB41AC158393CC291088B76F201A
Inverse expanded keys = [
    fd83db41ac158393cc291088b76f201a
    91879460519658d2603c931b7b463092
    508d33cfc011ccb231aacbc91b7aa389
    a0c83a2a909cff7df1bb077b2ad06840
    9f60d8933054c5576127f806db6b6f3b
    96e8ff67af341dc451733d51ba4c973d
    f344af8e39dce2a3fe472095eb3faa6c
    473a36d7ca984d2dc79bc23615788af9
    5268bc628da27bfa0d038f1bd2e348cf
    b1aef4fcdfcac79880a1f4e1dfe0c7d4
    336d62336e6433645f6b33795f413335
]
Cipher key: 336d62336e6433645f6b33795f413335
As string: '3mb3nd3d_k3y_A35'
```

```
openssl aes-128-ecb -in secret.dat -out dec_secret.dat -d -K 336d62336e6433645f6b33795f413335
cat ./dec_secret.dat
Congrats! Enter whiteboxblackhat into the Egg-o-Matic!
```

Flag: he19-fPHI-HUKJ-u15q-Lvwz

15 - Seen in Steem

An unknown person placed a secret note about Hacky Easter 2019 in the Steem blockchain. It happend during Easter 2018.

Go find the note, and enter it in the egg-o-matic below. Lowercase only, and no spaces!

Solution

Steem is a blockchain with multiple dApps. In a blockchain I should be able to get all the contents. Then I will search by date.

It is not clear to me what the differences are between steemit and esteem. After installing the app, it seems that this is just another electron app that shows the same stuff as the webif.

Ways to access this blockchain:

- SteemSQL (Subscription only)
- Blockchain Explorer (no free text seaerch) https://steemblockexplorer.com/ (https://steemblockexplorer.com/)
- SteemDB has search by date, but only shows top 100 posts https://steemdb.com (https://steemdb.com)
- Direct API https://api.steemit.com/ (https://api.steemit.com/)

Easter Date 2018: Sunday (01/04/2018). The description says during easter, so it could be between 30/03/2018 and 02/04/2018. The free APIs seem to lack full text search capabilities. Maybe I need to download the “full” blockchain, at least the part in easter 2018. Or query the API for all blocks during that timeframe.

For full-node: https://github.com/steemit/hivemind (https://github.com/steemit/hivemind)

16 - Every-Thing

After the brilliant idea from here.

The data model is stable and you can really store Every-Thing.

Solution

Receive SQLdump for mysql. Every thing can reference another thing (DAG). While grepping through the data I see that there are base64 encoded png headers.

```
CREATE TABLE `Thing` (
  `id` binary(16) NOT NULL,
  `ord` int(11) NOT NULL,
  `type` varchar(255) NOT NULL,
  `value` varchar(1024) DEFAULT NULL,
  `pid` binary(16) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FKfaem61vklu1cjw9ckunvpicgi` (`pid`),
  CONSTRAINT `FKfaem61vklu1cjw9ckunvpicgi` FOREIGN KEY (`pid`) REFERENCES `Thing` (`id`)
)
```

After exploring the data with SQL-cli I see that there are three main types of data: Posts, Contacts and Images. I'll start with the images as they are usually QR codes and because I couldn't find a string resembling a flag with grep.

Pure SQL solution

```
select t1.value, t2.value, t3.value, t4.type, t4.ord, t4.value, t5.ord, t5.value FROM `Thing` t1
LEFT JOIN Thing t2 ON t1.id = t2.pid
LEFT JOIN Thing t3 ON t2.id = t3.pid
LEFT JOIN Thing t4 ON t3.id = t4.pid
LEFT JOIN Thing t5 on t4.id = t5.pid
WHERE t1.type like 'root' AND t2.type like 'galery'
ORDER by t1.ord, t2.ord, t3.ord, t4.ord, t5.ord
```

This gives us all the values in the right order. However it still needs some stiching in excel and a conversion.

```
# having pic1.text with each base64 value line by line
with open('pic1.text', 'r') as fsrc:
    with open('pic1.png', 'wb') as fdst:
        for line in fsrc:
            fdst.write(base64.b64decode(line))
```

Python

As I didn't want to do too much in excel I programmed my way out of it in python.

```
import pymysql.cursors
from base64 import b64decode

connection = pymysql.connect(host='1.2.3.4',
                             user='asdf',
                             password='asdf',
                             db='he19_16',
                             charset='utf8mb4',
                             cursorclass=pymysql.cursors.DictCursor)

def getChildrenRecursive(cursor, id):
    sql = 'SELECT * from Thing where pid = %s ORDER BY ord'
    cursor.execute(sql, (id,))
    output = []
    for c in cursor.fetchall():
        children = getChildrenRecursive(cursor, c['id'])
        if len(children) is 0:
            output.append(c)
        else:
            output.append(children)

    return output

def flattenArray(myArray):
    output = []
    for e in myArray:
        if type(e) is dict:
            output.append(e['value'])
        elif type(e) is list:
            output += flattenArray(e)
        else:
            raise Exception('Fuuu')
    return output

def createImageFromList(myList, filename):
    with open(filename, 'wb') as f:
        for entry in myList:
            f.write(b64decode(entry))

def getAlbums(cursor):
    sql = "SELECT * from Thing where pid = (SELECT id from Thing where type like 'root') AND type='galery'"
    cursor.execute(sql)
    album_master = cursor.fetchone()
    return getChildrenRecursive(cursor, album_master['id'])

try:
    with connection.cursor() as cursor:
        albums = getAlbums(cursor)
        for i, album in enumerate(albums):
            img = flattenArray(album)
            createImageFromList(img, 'image_{}.png'.format(i))

finally:
    connection.close()
```

I want to thank my algorithm and data structures teacher :-)

Flag: he19-qKaG-VHmv-Mm26-0mwy

17 - New Egg Design

Thumper is looking for a new design for his eggs. He tried several filters with his graphics program, but unfortunately the QR codes got unreadable. Can you help him?!

Solution

Not sure if I have to use the eggdesign.png or the challange pic.

18 - Egg Storage

Last year someone stole some eggs from Thumper.

This year he decided to use cutting edge technology to protect his eggs.

https://hackyeaster.hacking-lab.com/hackyeaster/challenges/eggstorage/index.html (https://hackyeaster.hacking-lab.com/hackyeaster/challenges/eggstorage/index.html)

Solution

Passwordfield using Webassembly to validate password. However the egg is stored on the server itself.

I used Google Chrome and Firefox to analyze the WebAssembly. By trying out different inputs I traced the exit conditions. These conditions I wrote down using a constraint solver https://labix.org/python-constraint (https://labix.org/python-constraint). Before that I tried it manually but it was too cumbersome.

```
from constraint import *
from string import ascii_letters, digits

all_chars = ascii_letters + digits
all_domain_chars = list(map(ord, list(all_chars)))
#all_domain_chars = list(range(32,126))
print(all_domain_chars)

dest_length = 24

limited_chars = "01345HLXcdfr"
limited_domain_chars = list(map(ord, list(limited_chars)))

def xor_constraint(*args):
    state = 0
    for x in args:
        state = state ^ x
    return state == 44

problem = Problem()

# first four characters are clear
problem.addVariable(0, [ord('T')]) # T
problem.addVariable(1, [ord('h')]) # h
problem.addVariable(2, [ord('3')]) # 3
problem.addVariable(3, [ord('P')]) # P

# only some acceptable chars
problem.addVariable(4, limited_domain_chars)

# add all other characters
problem.addVariables(list(range(5,24)), all_domain_chars)

problem.addConstraint(lambda a, b: a == b, [23, 17])
problem.addConstraint(lambda a, b: a == b, [12, 16])
problem.addConstraint(lambda a, b: a == b, [22, 15])
problem.addConstraint(lambda a, b: a - b == 14, [5, 7])
problem.addConstraint(lambda a, b: a + 1 == b, [14, 15])
problem.addConstraint(lambda a, b: a % b == 40, [9, 8])
problem.addConstraint(lambda a, b, c: a - b + c == 79, [5, 9, 19])
problem.addConstraint(lambda a, b, c: a - b == c, [7, 14, 20])
problem.addConstraint(lambda a, b, c: (a % b) * 2 == c, [9, 4, 13])
problem.addConstraint(lambda a, b: (a % b)  == 20, [13, 6])
problem.addConstraint(lambda a, b, c: (a % b) == c - 46, [11, 13, 21])
problem.addConstraint(lambda a, b, c: (a % b) == c, [7, 6, 10])
problem.addConstraint(lambda a, b: (a % b) == 2, [23, 22])
problem.addConstraint(ExactSumConstraint(1352), [x for x in range(4,24)])
problem.addConstraint(xor_constraint, [x for x in range(4,24)])

def extractSolution(mySolDict):
    return  [mySolDict[x] for x in range(24)]

def prettyPrintSolution(mySolDict):
    sol = extractSolution(mySolDict)
    chars = list(map(chr, sol))
    return ''.join(chars)

xor_sol = [102, 81, 1, 105, 80, 19, 87, 80, 3, 106, 6, 7, 7, 123, 5, 4, 80, 11, 6, 7, 87, 122, 80, 4]
```

```
def xor_solution(s, t):
    return [(a^b) for a, b in zip(s, t)]

def possible_solution(sol):
    return all(map(lambda x: x in all_domain_chars + [ord('-')], sol))

for i, solution in enumerate(problem.getSolutionIter()):
    extracted = extractSolution(solution)
    xor_temp = xor_solution(extracted, xor_sol)
    if possible_solution(xor_temp):
        print(i, prettyPrintSolution(solution), ''.join(list(map(chr, xor_temp))))
```

This gave about 5000 possiblities. In the end of the wasm it xors the input with a static key. I therefore applied this key to all possibilites and only had 21 possiblities left after sorting out all unprintable solutions.

These 21 possiblites I just “brute-forced”.

Password: Th3P4r4d0X0fcH01c3154L13

Flag: he19-DJXj-nL5q-BrfK-7z1x

19 - CoUmpact DiAsc

Today the new eggs for HackyEaster 2019 were delivered, but unfortunately the password was partly destroyed by a water damage.

Solution

All uppercase letters are CUDA.

```
Enter Password: asdf
Cuda error: CUDA driver version is insufficient for CUDA runtime version
```

This seems to be more tricky. Have to use https://docs.nvidia.com/cuda/cuda-binary-utilities/index.html (https://docs.nvidia.com/cuda/cuda-binary-utilities/index.html) to investigate further.

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin>nvdisasm.exe C:\Users\admin\Documents\easter19
\19\coumpactdiasc.elf
nvdisasm fatal   : C:\Users\admin\Documents\easter19\19\coumpactdiasc.elf is not a supported Elf file
```

```
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin>cuobjdump.exe C:\Users\admin\Documents\easter19
\19\coumpactdiasc.elf

Fatbin elf code:
================
arch = sm_30
code version = [1,7]
producer = <unknown>
host = linux
compile_size = 64bit

Fatbin elf code:
================
arch = sm_30
code version = [1,7]
producer = cuda
host = linux
compile_size = 64bit

Fatbin ptx code:
================
arch = sm_30
code version = [6,3]
producer = cuda
host = linux
compile_size = 64bit
compressed
```

https://github.com/hyqneuron/asfermi/wiki (https://github.com/hyqneuron/asfermi/wiki)

Extract cuda-elf > cuobjdump -xelf all coumpactdiasc.elf

Disassemble directly > cuobjdump -ptx -sass coumpactdiasc.elf > asm-with-ptx.txt > cuobjdump -xptx all coumpactdiasc.elf

Disassemble with better annotations > nvdisasm coumpactdiasc.2.sm_30.cubin > 2.asm

Generate CFG > nvdisasm -cfg coumpactdiasc.2.sm_30.cubin > 2.cfg

20 - Scrambled Egg

This Easter egg image is a little distorted...

Can you restore it?

Solution

Picture with stripes in different colors, has transparent pixels between some stipres. Transparent pixels also have a color, but only in one channel.

```
hacker@kali:~/Documents/easter19/20$ file egg.png
egg.png: PNG image data, 259 x 256, 8-bit/color RGBA, non-interlaced
```

By manually looking at the lines I see three transparent pixels per line. Verified with python that every line has exactly three of them, all have the same max value but for other colours.

```
(0, 107, 0, 0)(107, 0, 0, 0)(0, 0, 107, 0)
(240, 0, 0, 0)(0, 240, 0, 0)(0, 0, 240, 0)
```

Maybe I need to drop the three transparent pixels.

21 - The Hunt: Misty Jungle

Welcome to the longest scavenger hunt of the world!

The hunt is divided into two parts, each of which will give you an Easter egg. Part 1 is the Misty Jungle.

To get the Easter egg, you have to fight your way through a maze. On your journey, find and solve 8 mini challenges, then go to the exit. Make sure to check your carrot supply! Wrong submissions cost one carrot each.

Solution

Mistery Jungle is at path 1804161a0dabfdcd26f7370136e0f766. Server sends new session cookie for every request.

How to navigate in this thing? There doesn't seem to be a real clicky gui.

```
<script type='text/javascript'>
    // add all this variables later
    let youCanTouchThis = "";
    let youCantTouchThis = "";
    let randomNumber = undefined;

    for (let i = 0; i < youCantTouchThis.length; i++) {
        if (youCantTouchThis.charCodeAt(i) === 28) {
            youCanTouchThis += '&';
        } else if (youCantTouchThis.charCodeAt(i) === 23) {
            youCanTouchThis += '!';
        } else {
            youCanTouchThis += String.fromCharCode(youCantTouchThis.charCodeAt(i) - randomNumber);
        }
    }
    // document.write(m);
</script>
```

What is this?

Statement in walking.

```
``bqq`vsm``0npwf0y0z
```

Don't know what it means.

Giving bad feedback:

```
WHY ARE YOU UNHAPPY?!

OMG! Tell me what's your problem? We are doing nothing to make our customers happy. So why aren't you happy,
too?! WHAT'S YOUR PROBLEM?!

Wait ...
Please don't tell me.
I don't care. Bye!

Flag: he19-YouT-u.be-TKN7-IvhY-H2M

Please, don't feel free to visit us again!
```

Unfortunately not a hidden flag, just some "random" youtube video.

Feedback for path 3:

> Sorry we don't accept feedback for path 3 yet. If you are a beta contributor you already got the link to the route via mail. It's very similar to the links of path 1 and path 2. If you lost it, just recover it on your own.

23 - The Maze

Can you beat the maze? This one is tricky - simply finding the exit, isn't enough!

```
nc whale.hacking-lab.com 7331
```

Solution

```
To navigate through the maze use the following commands:
  - go <direction>  (north, south, west, east)
  - search
  - pick up
  - open
  - exit

Press enter to resume to the menue.
```

Maze gets dynamically generated. Probably need to write an automatic maze traverser.

```
Please enter the key:
> c70909aa5941a0fc2dd3d28e136bab26
.[H.[JCongratulation, you solved the maze. Here is your reward:
                  *****
               ****   ****
            ***         ***
          ***             ***
        ***                 ***
      ***     ****   ****     ***
     **       ** *** **  **       **
    **            **   ***.        **
   **          .***  **  **         **
  **          ******  ****           **
 **                                   **
**         +------------------+         **
*          | +--+ *  *  +--+ |          *
*          | |  |  ** * |  | |          *
*          | +--+ ** ** +--+ |          *
*          |  * **  ** *** * |          *
*          | * *  ** *** * * |          *
**         | +--+ * *  [] *  |         **
 *         | |  |  *** ** ** |         *
 **        | +--+ ** *** **  |        **
  **       +------------------+       **
   **                               **
    ***                           ***
      ***                       ***
        ****                 ****
            *****       *****
                *********
Press enter to return to the menue
```

How to extract this? `Der Weg ist das Ziel` because simply finding the exit isn't enough?

Maze Solver

```
import pyte
import random
import asyncio
from typing import Tuple, Dict
from socket import socket
from collections import deque
from dataclasses import dataclass
from enum import Enum

items_to_ignore = [
    'There is nothing interesting here.',
    "There is a message scratched into the wall 'Hacky Easter 2019, by Darkice'."
    'Ugh! There is a giant rat!',
    "You found a broken key, looks like it won't work anymore.",
    'You found an arrow stuck in the wall.',
    'Ugh! There is a skeleton!',
    'You found a map, but unfortunately someone else has already torn out a piece.',
    'Ugh! There is a spider!',
    'You found a rusty nail.',
    ''
]

interesting_items = [
    'You found a key!',
    'There is a wall',
    'You found a locked chest!',
    'Wrong option!'
]

unknown_items = []

screen_history = deque([], 100)
network_screen_history = deque([], 25)
decision_history = deque([], 100)

socket_receive_history = deque([], 100)

def printScreen(lines):
    for l in lines:
        print(l)

def isPathBlocked(lines):
    north = lines[6][12] == '-'
    west = lines[8][9] == '|'
    east = lines[8][15] == '|'
    south = lines[10][12] == '-'
    return (north, west, east, south)




def shift(l, n):
    return l[n:] + l[:n]

def getNextMove(paths, last_move):
    north, west, east, south = paths

    compass = [north, east, south, west]
    direction = [b'go north', b'go east', b'go south', b'go west']
```

```
    if last_move == None:
        return getNextRandomMove(paths)


    if last_move == b'go north':
        return getWallAlgorithm(compass, direction)
    if last_move == b'go east':
        return getWallAlgorithm(shift(compass, 1), shift(direction, 1))
    if last_move == b'go south':
        return getWallAlgorithm(shift(compass, 2), shift(direction, 2))
    if last_move == b'go west':
        return getWallAlgorithm(shift(compass, 3), shift(direction, 3))
    raise Exception('Prison!')

def getWallAlgorithm(compass, direction):
    #   If you can turn left, do it.
    #   Else (if you can't turn left), if you can continue going straight, just go straight.
    #   Else (if you can't do either of the previous steps), if you can turn right, do it.
    #   If you reached a dead end, turn back by turning around (in either direction) 180 degrees.
        if not compass[3]: # // turn left if possible left
            return direction[3]
        if not compass[0]: # go straight if possible
            return direction[0]
        if not compass[1]: # turn right if possible
            return direction[1]
        if not compass[2]:
            return direction[2]
        raise Exception('No Direction')

def getNextRandomMove(paths):
    north, west, east, south = paths
    moves = []
    if not west:
        moves.append(b'go west')
    if not north:
        moves.append(b'go north')
    if not east:
        moves.append(b'go east')
    if not south:
        moves.append(b'go south')
    return random.choice(moves)

def getNextNonLoopingMove(paths, last_move):
    top, left, right, bottom = paths

    if not top and last_move != b'go south':
        return b'go north'
    elif not left and last_move != b'go east':
        return b'go west'
    elif not right and last_move != b'go west':
        return b'go east'
    elif not bottom and last_move != b'go north':
        return b'go south'
    return getNextRandomMove(paths)

def getKey(lines):
    return lines[16].split(':')[1].strip()

def screenHasErrors(lines):
```

```
    return lines[0].strip() != 'Your position:' or lines[8][12] != 'X'

def moveToXY(move):
    if move == b'go south':
        return (0, -1)
    elif move == b'go east':
        return (+1, 0)
    elif move == b'go west':
        return (-1, 0)
    elif move == b'go north':
        return (0, +1)

class MazeConnector(asyncio.Protocol):
    def __init__(self, helper, loop):
        self.loop = loop
        self.helper = helper
        self.transport = None

        self.buffer = b''

        self.screen = pyte.Screen(80, 24)
        self.byteStream = pyte.ByteStream(self.screen)



    def connection_made(self, transport):
        # transport.write(self.message.encode())
        print('Connected!')
        self.transport = transport

    def verifyScreenSimple(self):
        for line in self.screen.display:
            if '>' in line or 'menue' in line:
                return True
        return False

    def data_received(self, data):
        #print('Data received: {!s}'.format(data.decode()))
        #print(data)
        self.byteStream.feed(data)
        network_screen_history.append(self.screen.display)
        if self.verifyScreenSimple():
            answer = self.helper.handleData(self.screen)
            if answer:
                self.transport.write(answer)
                #print('Data sent: {!r}'.format(answer.decode()))
                self.buffer = b''
            else:
                self.buffer += data
        else:
            pass
            #print('wait for more')

    def connection_lost(self, exc):
        print('The server closed the connection')
        print('Stop the event loop')
        self.loop.stop()

@dataclass
class PosInfo:
```

```
    searched : bool = False
    isKey : bool = False
    isChest : bool = False
    count : int = 0
    moves = None

class GameState(Enum):
    nameInput = 1
    mainMenu = 2
    running = 3
    waitForFlag = 4

class MazeHelper:
    def __init__(self):
        self.state : GameState = GameState.nameInput
        self.temp_screen = pyte.Screen(80, 24)
        self.temp_byteStream = pyte.ByteStream(self.temp_screen)

        self.key = None
        self.last_move = None

        self.moves = 0

        self.x = 0
        self.y = 0
        self.mazemap : Dict[Tuple[int, int], PosInfo] = {}

    def handleData(self, screen):

        if self.state == GameState.nameInput:
            if screen.display[0].strip() == 'Please enter your name:' and screen.display[1].strip() == '>':
                self.state = GameState.mainMenu
                return b'Hello world!\n'
            else:
                return None

        if self.state == GameState.mainMenu:
            if 'Choose:' in screen.display[0]:
                self.state = GameState.running
                return b'3\n'
            else:
                return None

        if self.state == GameState.running:
            if 'You pick up the key' in screen.display[16]:
                self.key = screen.display[16].split(':')[1].strip()
                return self.playTheGame(screen)
            elif 'The chest is locked' in screen.display[19] and self.key:
                self.state = GameState.waitForFlag
                print('enter chest key')
                return self.key.encode('utf-8') + b'\n'
            elif 'Your position' in screen.display[0]:
                if '' != screen.display[16].strip():
                    return self.handleSearchItem(screen)
                return self.playTheGame(screen)

        if self.state == GameState.waitForFlag:
#             print(network_screen_history)
            printScreen(screen.display)
```

```
            self.state = GameState.mainMenu
            import pickle
            import time

            with open('pickle/' + str(int(time.time())) + '.pick', 'wb') as f:
                pickle.dump(self.mazemap, f)
            return b'\n'

        print('Unknown input', self.state)
        #printScreen(screen.display)
        return None
        #raise Exception('Unknown input')

    def playTheGame(self, screen):
        if screenHasErrors(screen.display):
            #print("Data-Error!")
            return None

        screen_history.append(screen.display)

        if (self.x, self.y) not in self.mazemap:
            self.mazemap[(self.x, self.y)] = PosInfo()
            return b'search\n'

        posInfo = self.mazemap[(self.x, self.y)]

        blocked_paths = None
        if posInfo.moves:
            blocked_paths = posInfo.moves
        else:
            blocked_paths = isPathBlocked(screen.display)
            posInfo.moves = blocked_paths
        next_move = getNextMove(blocked_paths, self.last_move)
        decision_history.append({"next_move": next_move, "blocked_paths": blocked_paths})

        posInfo.count += 1

        self.moves += 1
        xdiff, ydiff = moveToXY(next_move)
        self.x += xdiff
        self.y += ydiff

        self.last_move = next_move

        if self.moves % 50 == 0:
            print('Move', self.moves, "Key", self.key != None, "Mazemap", len(self.mazemap))

        return next_move + b'\n'

    def handleSearchItem(self, screen):
        assert (self.x, self.y) in self.mazemap
        item = screen.display[16].strip()
        if item in interesting_items:
            if item == 'You found a key!':
                self.mazemap[(self.x, self.y)].isKey = True
                self.mazemap[(self.x, self.y)].searched = True
                print("Key found, going to pick up")
                return b'pick up\n'
            elif item == 'There is a wall!':
```

```
                print("Wallbreaker!")
            elif item == 'Wrong option!':
                print('Wrong option!')
            elif item == 'You found a locked chest!' and self.key:
                self.mazemap[(self.x, self.y)].isChest = True
                self.mazemap[(self.x, self.y)].searched = True
                print('chest-found and going to upen!')
                return b'open\n'
        elif item not in items_to_ignore and item not in unknown_items:
            print("Unkown thing found:", item)
            unknown_items.append(item)

        return self.playTheGame(screen)

def main():

    helper = MazeHelper()
    loop = asyncio.get_event_loop()
    coro = loop.create_connection(lambda: MazeConnector(helper, loop),
                                  'whale.hacking-lab.com', 7331)
#                                  '192.168.109.138', 7331)
    loop.run_until_complete(coro)
    loop.run_forever()
    loop.close()

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

24 - CAPTEG

CAPTEG - Completely Automated Turing test to know how many Eggs are in the Grid

CAPTEG is almost like a CAPTCHA. But here, you have to proof you are a bot that can count the eggs in the grid quickly. Bumper also wanna train his AI for finding eggs faster and faster ;)

CAPTEG page http://whale.hacking-lab.com:3555/ (http://whale.hacking-lab.com:3555/)

No Easter egg here. Enter the flag directly on the flag page.

Solution

Displays 9 images, most of them contain eggs. One has 7 seconds to count them.

Probably need to use Tensorflow Object Detection API: https://github.com/tensorflow/models/tree/master/research/object_detection (https://github.com/tensorflow/models/tree/master/research/object_detection)