Brian Leung UofM ’22

matryoshka

b01lers_ctf 2020

CATEGORY: Images

PTS: 200

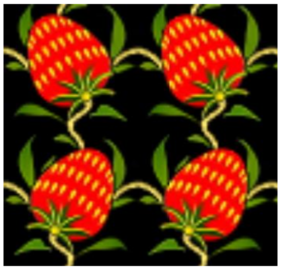
DESC: We are given a large image consisting of many strawberries. Through image processing and QR code shenanigans, we can get the flag.

TOOLS: MATLAB, 7zip, Snipping Tool, https://online-barcode-reader.inliteresearch.com/, CyberChef, Google

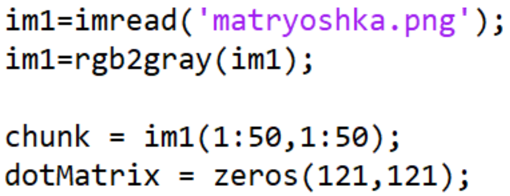
DETAILED APPROACH (Solution Summary at end):

We are given a big image named ‘matryoshka.png’, which consists of an ungodly amount of strawberries against a black background. If we look closer, we can see that certain strawberries are turned to the left and certain strawberries are turned to the right. Hmm. They must mean something.

Strawberries.

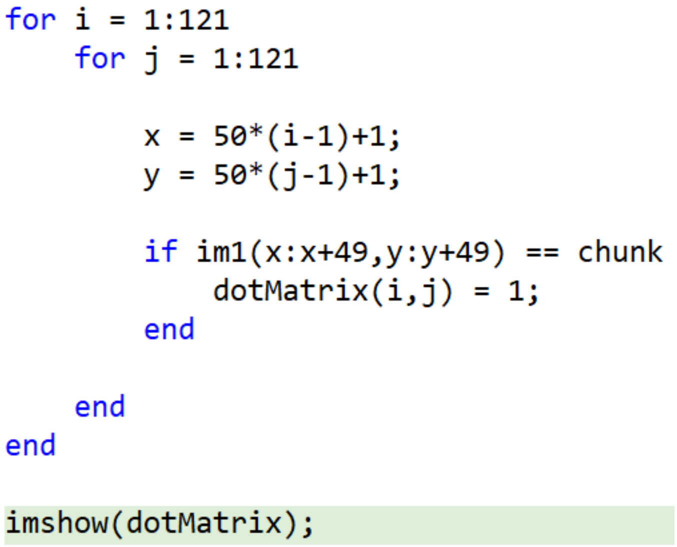


I decide to throw this entire image into MATLAB and make a matrix, setting each “left strawberry” as a 1 and each “right strawberry” as a 0, just to see what comes out. I first load the image into MATLAB and convert the RGB values to black and white, to make my life easier.



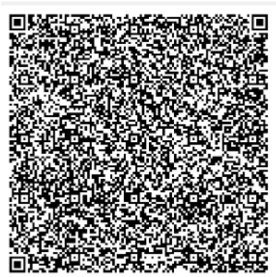
```
im1=imread('matryoshka.png');
im1=rgb2gray(im1);

chunk = im1(1:50,1:50);
dotMatrix = zeros(121,121);
```

By inspecting the ‘im1’ matrix in the variable viewer I discover that each berry is 50x50 pixels. By dividing the dimensions of the entire image by 50, I figure that we are given a 121x121 matrix of strawberries. The ‘chunk’ variable is a snip of the first strawberry, which is left facing. ‘dotMatrix’ is what we’re going to store all the 1s and 0s in.



```
for i = 1:121
    for j = 1:121

        x = 50*(i-1)+1;
        y = 50*(j-1)+1;

        if im1(x:x+49,y:y+49) == chunk
            dotMatrix(i,j) = 1;
        end

    end
end

imshow(dotMatrix);
```

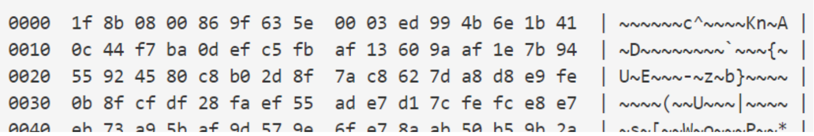
This block of code goes through the entire ‘im1’ matrix and sets all left strawberries to 1 and all right strawberries to 0. Lo and behold, imshow() gives us a 121x121 QR code:



QR1

We're not exactly out of the woods yet. When read by a QR code reader, it gives us gibberish. However, in a hex dump…

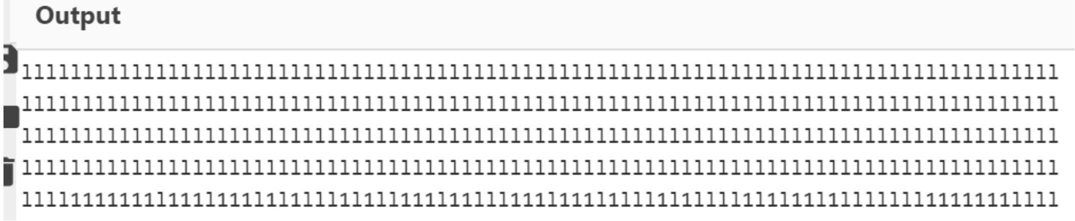
Snippet:



```
0000  1f 8b 08 00 86 9f 63 5e  00 03 ed 99 4b 6e 1b 41  | ~~~~~~c^~~~~Kn~A |
0010  0c 44 f7 ba 0d ef c5 fb  af 13 60 9a af 1e 7b 94  | ~D~~~~~~~~`~~~{~ |
0020  55 92 45 80 c8 b0 2d 8f  7a c8 62 7d a8 d8 e9 fe  | U~E~~~-~z~b}~~~~ |
0030  0b 8f cf df 28 fa ef 55  ad e7 d1 7c fe fc e8 e7  | ~~~~(~~U~~~|~~~~ |
0040  eb 73 a9 5b af 9d 57 9e  6f e7 8a ab 50 b5 9b 2a  | ~s~[~~W~O~~~P~~* |
```

I admit the hex dump still looks like gibberish, but the first few hex values indicate that it's in the gz format. Copying and pasting this into CyberChef yields:

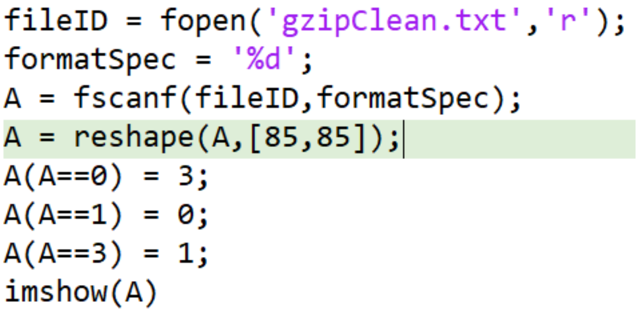
Hex dump to gz:



Output

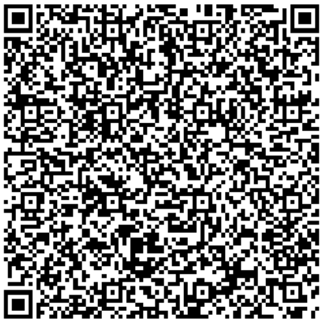
```
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

Now we're talking. They were being cheeky and made a dump full of 1s and lowercase Ls. I replace each L with a 0 and put a space between every character (Use your method of choice). I suspect that this is another QR code and, counting the characters, determine that this one is 85x85. I throw this cleaned data back into MATLAB and view it through imshow().



```
fileID = fopen('gzipClean.txt','r');
formatSpec = '%d';
A = fscanf(fileID,formatSpec);
A = reshape(A,[85,85]);
A(A==0) = 3;
A(A==1) = 0;
A(A==3) = 1;
imshow(A)
```

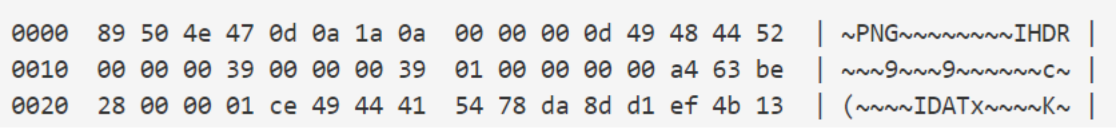
True to the chall’s name, the image is another QR code.



QR2

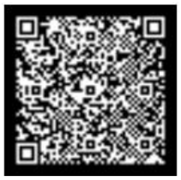
Knowing the drill now, I check the hex dump. The first few hex values indicate that it’s a PNG.

Snippet:

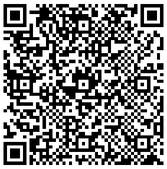


```
0000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52  | ~PNG~~~~~~~~IHDR |
0010  00 00 00 39 00 00 00 39  01 00 00 00 00 a4 63 be  | ~~~9~~~9~~~~~~c~ |
0020  28 00 00 01 ce 49 44 41  54 78 da 8d d1 ef 4b 13  | (~~~~IDATx~~~~K~ |
```

I threw it into CyberChef. This QR code came out inverted, but it can simply be fixed with a bit of photo editing.



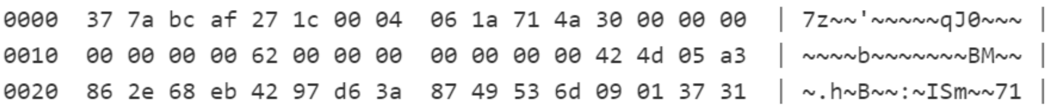
to



QR3

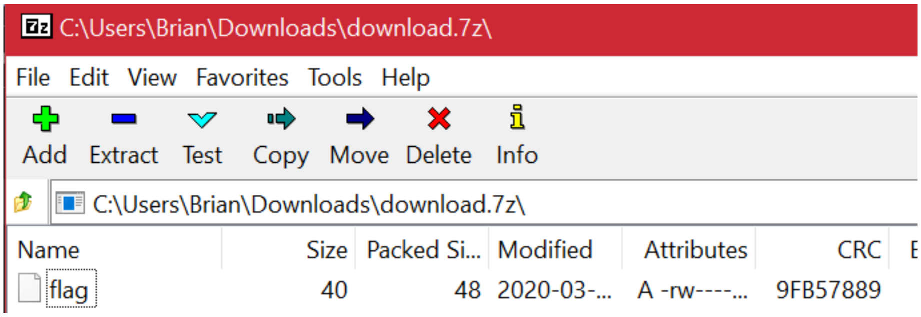
One more time. This time around, we get a .7z file:

Snippet:



```
0000  37 7a bc af 27 1c 00 04  06 1a 71 4a 30 00 00 00  | 7z~~'~~~~~qJ0~~~ |
0010  00 00 00 00 62 00 00 00  00 00 00 00 42 4d 05 a3  | ~~~~b~~~~~~~BM~~ |
0020  86 2e 68 eb 42 97 d6 3a  87 49 53 6d 09 01 37 31  | ~.h~B~~:~ISm~~71 |
```

Saving the output to a .7z and looking inside the archive reveals our target:



C:\Users\Brian\Downloads\download.7z\

C:\Users\Brian\Downloads\download.7z\

Name	Size	Packed Si...	Modified	Attributes	CRC	E
flag	40	48	2020-03-...	A -rw----...	9FB57889	

However, the file is password protected. What could this password be? The chall description is in Russian, but translated, it says that the "super secret" password is 1234. Awesome. The password works and opening the flag file reveals:

pctf{dolls_do_get_boring_after_a_while}

SOLUTION SUMMARY:

Convert the initial strawberry field into a QR(1) by replacing each left strawberry with black and each right strawberry with white.

Convert the hexdump to gz. Replace the Ls in the converted data with 0s and display another QR(2).

Convert the hexdump of this new QR(2) to PNG. The PNG reveals a new QR(3). Convert this third hex dump into a 7z file.

Unpackage the 7z file with the password given in the chall description and the flag file is revealed.

P.S.: Why MATLAB?
For manipulating images as matrices, MATLAB is an incredibly strong tool. Python is not bad, but MATLAB's development environment is a one-stop shop for viewing images, inspecting individual values of cells, and handling big matrices in general (after all it was built with that purpose in mind).

RELEVANT CODE:

Strawberry to QR (MATLAB)

```
close all
clear all

im1=imread('matryoshka.png');
im1=rgb2gray(im1);

chunk = im1(1:50,1:50);
dotMatrix = zeros(121,121);
for i = 1:121
    for j = 1:121

        x = 50*(i-1)+1;
        y = 50*(j-1)+1;

        if im1(x:x+49,y:y+49) == chunk
            dotMatrix(i,j) = 1;
        end

    end
end

imshow(dotMatrix);
```

Ls and 1s to a clean GZ (Python)

```
fin = open('gzipData.txt','r')
data = fin.readlines()
for i in range(0,len(data)):
    data[i] = data[i].replace("l","0")
    data[i] = data[i].replace("\r","")
    data[i] = data[i].replace(""," ")

fout = open('gzipClean.txt','w')
fout.writelines(data)

fout.close()
fin.close()
```

A clean GZ to QR (MATLAB)

```
fileID = fopen('gzipClean.txt','r');
formatSpec = '%d';
A = fscanf(fileID,formatSpec);
A = reshape(A,[85,85]);
A(A==0) = 3;
A(A==1) = 0;
A(A==3) = 1;
imshow(A)
```