Hackyeaster 2019 Writeup

01 – Twisted

This was a lot harder than it looked like. I first tried various transformation tools and distortion filters in gimp (especially the warp one) but none lead to a scannable result. Until I found the **whirl and pinch** filter, which is most likely the one which was used in the first place to distort the egg. With that one, it was very easy to straighten the qr code.

02 – Just watch

Well this one was a breeze, just download the file and open it in gimp, where you can have a look at each frame individually. Then use some table from wherever to translate the text (for example this one: http://www.skylarrobbins.com/wp-content/uploads/2015/02/fingerspelling.jpg).

03 – Sloppy encryption

I solved this one but already deleted the file *facepalm*

04 – Disco 2

This is a cool one! First, I had to realize that you can move the camera in this 3D environment and then I realized, that there was a QR-Code inside the disco-ball made up of mirrors. To get the code to be scannable however, there were some adjustment to the rendering code to be done. For this to work, its easiest to download the website and work on a local copy and edit the js there. So here are the modifications I made:

1. On line 117 set the sphere radius to -400 instead of 400. This has the effect, that we can now see the inside of the sphere but not the outside. Also change the colour of the sphere material to FFFFFF (white) for better contrast on line 122.
2. Comment out the line 142 so the mirrors don't face the centre of the sphere.
3. Add a conditional to line 143 so the mirror only gets added to the scene if the distance to the centre of the sphere is less than 390. Use this formula:
   m[0]*m[0]+m[1]*m[1]+m[2]*m[2] < 390*390 (3D-Phytagoras)
4. Reload the page and volà, you got the qr code.

05 – Call for papers

This one clearly is a steganography challenge, but it took me some time to find out what technique was used. The crucial hint can be found in the document properties: The Author of the document is "SCIpher". When you google that name, you find this website: https://pdos.csail.mit.edu/archive/scigen/scipher.html where you can simply put in the text of the document and it gives you your flag.

06 – Dots

This one was driving me nuts and when I finally figured it out, I was so angry at myself for not figuring it out earlier because I the technique was very popular in my early school days to hide messages. It's simple, you just take a piece of paper and punch some holes in it where the dots are, then you overlay the letters with it, write the visible letters down (regular reading direction) and then – and this is the clue – you rotate the paper 90 degrees and repeat the process. Of course, you have that one region which is covered by those stupid golden and green dots, so you'll be missing some letters. It is, however, quite easy to guess them after you've written the ones you know. You'll get the following string: "Hello Buck the password is WHITECHOCOLATE"

07 – Shell we argument

This one is quite simple as well – if you've got to grips with bash that is… So, you can just put an "echo" before the last line. If you then execute the script, instead of doing what it should do, it prints out a readable version of its own code. And in there you can very quickly and easily find the necessary arguments. Then remove the echo, execute it again, but this time with the arguments you found, and it will print your flag.

08 – Modern art

This is a sneaky one. You have to "cat" the image file, then you get a QR-code in ascii-art. When you scan that code, you get the string "AES-128". This alone isn't worth anything of course. When you then use the "strings" command on the image, you find two strings which are encapsulated in parenthesis. One is labelled with "KEY". So, all you got to do is decrypt the other string with the key using AES-128.

09 – rorriM rorriM

This is a very cool one. As the title suggests, some stuff here is mirrored. The first stage is to mirror a zip file. Just write a little script which reads the file byte by byte and writes them out to a new file in reverse order. Then open the archive and extract a PNG file. Here, only the magic bytes are reversed, so you can just swap them with a regular hex editor. And finally, invert the image itself in gimp.

11 – Memeory 2.0

This one was cool to solve once I figured out what the bug on the website was. I first tried to match some cards manually and looked at the network traffic. It's all simple post requests with the two card numbers you clicked on and the response is either "ok" or "not ok" with the number of mis-clicks since the start. However, even when I modified the css to see the pictures and then matched two pictures correctly, it still gave me a "not ok". This struck me as odd since I had no idea on how to find the pairs otherwise. But then by fiddling around a bit, I found something very interesting: when I clicked the first card (bottom right), then I could click on as many others as I wanted and didn't receive the "you are too slow or too bad hobo" message. I investigated and found out that the server was crashing whenever I clicked on that first card. Hmm interesting, I thought and looked at that card in detail. The problem was simple: the cards have a 0-based index whereas the images have a 1-based index… After that I could match some pictures manually, by editing the css and then clicking on a card one further than the one I actually wanted to click on. So now I knew that I could be sure that a pair of pictures would match a pair on the server. So, I wrote a script which downloads all the pictures, calculates the md5 sums of them all, searches for pairs and then makes a post request with the two numbers. It's important to pass the session cookie of course, so you stay in the game and then just repeat it ten times until the response is no longer just "ok" but the flag.

13 – Symphony in Hex

Well this one was waaaaaay too easy to be in the medium section. Just do as the hint tells: count the quavers in each beat and write down the number, and read the absolute pitch of the semibreves as a letter (C,D,E,F,G,H,A) and write them down as well. As there is no H in hex-notation, treat it as a B and then use a hex converter to convert the hex-string to a human readable string.

14 – Every-Thing

Not to hard to solve either: write a script which pulls all entries out of the table with equal “pid” values and writes them to a file in the order given by “ord”. Don’t forget to first convert the base64 encoded data to raw bytes. When you do that, you get a dozen png images and one of them is the egg.

18 – Egg storage

This was a hassle to solve. There is some Web assembly code you must reverse engineer. The main problem was, that all the disassemblers I used gave me wrong code. I was still able to get all the conditions that the string must meet though and wrote a python script, which just generates strings character by character and for each character checks whether the conditions are met. By doing this, I came up with a dozen of passwords which meet all the conditions. So, I put them in one by one until the page gave me the egg.

25 – Hidden egg 1

The hint is: “I like hiding Eggs in baskets”. Let’s search the page for baskets. Oh, look at that: on the submission page there is an image of a basket with egg. Let’s download it and have a look at it! After running the strings command on it I was successful: he19-xzCc-xElf-qJ4H-jay8 is the flag.