HackyEaster 2019 - Solutions

inik, 16.4.2019

01 - Twisted

Task

As usual, the first one is very easy - just a little twisted, maybe.

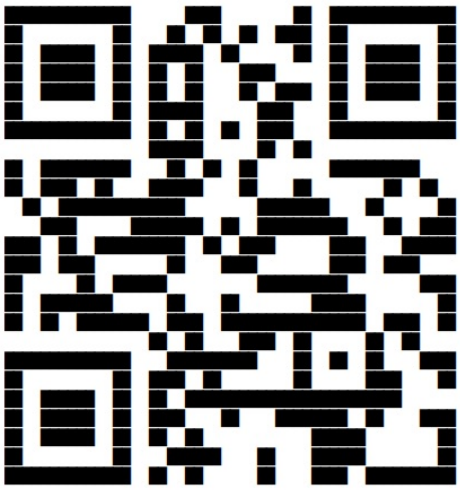
Solution

OK, that's straight forward. Unfortunately the first attempt using Gimp didn't work for me.

I decided to write i down as "1" and " " and convert it to a QR code in Java

```java
package egg01;

public class PrintQR {

    public static void main(String[] args) {
        String s = "";
        s += "\n";

        s += "1111111   11    1 1111111\n";
        s += "1     1  1 1 1 11 1     1\n";
        s += "1 111 1 111 1   1 1 111 1\n";
        s += "1 111 1 1 1  11   1 111 1\n";
        s += "1 111 1 1 1  1 11 1 111 1\n";
        s += "1     1 111   11  1     1\n";
        s += "1111111 1 1 1 11  1111111\n";
        s += "        111   1 1\n";
        s += "1 11111  1 1 1    11111\n";
        s += "1 11 1  111  1  1    1 1\n";
        s += " 1 1111111    1111  1   1\n";
        s += "  11   1111 1 11111  1 11\n";
        s += "11 1 11111 1 11   11  111\n";
        s += "11 11    1  11    11 11 \n";
        s += "1   1 111     11111    11\n";
        s += "1 1111   1  1    111    1\n";
        s += "1 1 1111 11 11111111 11 1\n";
        s += "        1 1  11 1   1 1  \n";
        s += "1111111  11111  1 1 1  11\n";
        s += "1     1 11 1    1   11  1\n";
        s += "1 111 1 1 11111111111\n";
        s += "1 111 1 11  1  1111 1  11\n";
        s += "1 111 1 1 1      111    1\n";
        s += "1     1  111  1  111    1\n";
        s += "1111111 1 1   1  11 11111\n";

        s = s.replace("1", "\u2588\u2588").replace(" ", "  ");
        System.out.println("RES:\n" + s);
    }
}
```

```
<terminated> PrintQR [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Apr 18, 201
RES:
```



Side note: The flag was he19-Eihb-UUVw-nObm-lxaW. But this was not readable with the Hacky Easter App, but with other QR Readers, so obviously it still contained some wrong blocks, but it was good enough...

02 - Just Watch

Task

Just watch and read the password.

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

Solution

It's the ASL (American Sign Language Alphabet), see https://de.wikipedia.org/wiki/Datei:Asl_alphabet_gallaudet.png.

Take the gif into gimp. There you can look at each frame and translate it.

The password is **givemeasign**

03 - Sloppy Encryption

Task

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

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

```
K7sAYzGlYx0kZyXIIPrXxK22DkU4Q+rTGfUk9i9vA60C/ZcQOSWNfJLTu4RpIBy/27yK5CBW+UrBhm0=
```

sloppy.rb

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

Solution

Ruby is rather a cryptic language. Because I never learned Ruby I tried to understand step by step what's happen by printing each intermediate result:

```
require"base64"
puts"write some text and hit enter:"
input=gets.chomp
h=input.unpack('C'*input.length).collect{|x|x.to_s(16)}.join
puts"h text:""#{h}"
ox='%#X'%h.to_i(16)
puts"ox text:""#{ox}"

x=ox.to_i(16)
puts"1 x text:""#{x}"
# x1=ox.to_i(16)*['5']
# puts"2 x text:""#{x1}"
x=['5'].cycle(101)
puts"3 x text:""#{x}"
x=['5'].cycle(101).to_a
puts"4 x text:""#{x}"
x=['5'].cycle(101).to_a.join
puts"5 x text:""#{x}"
x=['5'].cycle(101).to_a.join.to_i
puts"6 x text:""#{x}"
c=x.to_s(16)
puts"1 c text:""#{c}"
c=x.to_s(16).scan(/../)
puts"2 c text:""#{c}"
c=x.to_s(16).scan(/../).map(&:hex)
puts"3 c text:""#{c}"
c=x.to_s(16).scan(/../).map(&:hex).map(&:chr)
puts"4 c text:""#{c}"
c=x.to_s(16).scan(/../).map(&:hex).map(&:chr).join
puts"5 c text:""#{c}"preamble
b=Base64.encode64(c)
puts"encrypted text:""#{b}"
```

Output:

```
write some text and hit enter:
TEST
h text:54455354
ox text:0X54455354
1 x text:1413829460
3 x text:#<Enumerator:0x0000557c087a71f8>
```

```
4 x text:["5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5",
"5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5",
"5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5",
"5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5",
"5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5",
"5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5", "5",
"5", "5", "5", "5", "5", "5", "5"]
5 x
text:555555555555555555555555555555555555555555555555555555555555555555555555
555555555555555555555555
6 x
text:555555555555555555555555555555555555555555555555555555555555555555555555
555555555555555555555555
1 c
text:659950261e0527c476db8c455c426574eb9f876293265efac9be5102a9038e38e38e38e38e
38e38e3
2 c text:["65", "99", "50", "26", "1e", "05", "27", "c4", "76", "db", "8c", "45",
"5c", "42", "65", "74", "eb", "9f", "87", "62", "93", "26", "5e", "fa", "c9",
"be", "51", "02", "a9", "03", "8e", "38", "e3", "8e", "38", "e3", "8e", "38",
"e3", "8e", "38", "e3"]
3 c text:[101, 153, 80, 38, 30, 5, 39, 196, 118, 219, 140, 69, 92, 66, 101, 116,
235, 159, 135, 98, 147, 38, 94, 250, 201, 190, 81, 2, 169, 3, 142, 56, 227, 142,
56, 227, 142, 56, 227, 142, 56, 227]
4 c text:["e", "\x99", "P", "&", "\x1E", "\x05", "'", "\xC4", "v", "\xDB", "\x8C",
"E", "\\", "B", "e", "t", "\xEB", "\x9F", "\x87", "b", "\x93", "&", "^", "\xFA",
"\xC9", "\xBE", "Q", "\x02", "\xA9", "\x03", "\x8E", "8", "\xE3", "\x8E", "8",
"\xE3", "\x8E", "8", "\xE3", "\x8E", "8", "\xE3"]
5 c text:e▯P&'▯v؟E\Bet랏b▯&^▯rQ▯▯8▯▯8▯▯8▯▯
encrypted text:ZZlQJh4FJ8R224xFXEJldOufh2KTJl76yb5RAqkDjjjjjjjjjjjjjjjj
```

Ahh, now that's easy! I just have to revert the logic by:

1. base64 decode
2. convert this to a number (base 16)
3. divide it by a number consisting of 101 chars of '5'
4. Number is base 16, convert it to text

As usual, I solve this *not with ruby*:

```
package egg03;
import java.math.BigInteger;
import util.DecodeUtil;

public class RevertRuby {
        public static void main(String[] args) {
                String c =
"K7sAYzGlYx0kZyXIIPrXxK22DkU4Q+rTGfUk9i9vA60C/ZcQOSWNfJLTu4RpIBy/27yK5CBW+UrBhm0="
;
                String ds =
"555555555555555555555555555555555555555555555555555555555555555555555555555
5555555555555555555";
                byte[] cb = DecodeUtil.base64ToByteArray(c);
                BigInteger b = new BigInteger(cb);
                System.out.println("b: " + b);
                BigInteger d = new BigInteger(ds, 10);
                System.out.println("d: " + b);
                b = b.divide(d);
                System.out.println("p: " + new String(b.toByteArray()));
        }
```

```
}
```

Result:

```
b:
208306191836603693312477778177601749521683333333333333333333333333333333333333
333333333333333333312502714149672964002085555515573158381165
d:
208306191836603693312477778177601749521683333333333333333333333333333333333333
333333333333333333312502714149672964002085555515573158381165
p: n00b_style_crypto
```

04 - Disco 2

Task

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

Solution

My challenge!

It's trivial to solve:

- either change js code (remove grey sphere and all mirrors not on sphere (calculated by distance from point 0,0,0)), align the remaining mirrors and scan it
- or only look at the data and do it programmically

05 - Call for Papers

Task

Please read and review my CFP document, for the upcoming IAPLI Symposium. I didn't write it myself, but used some artificial intelligence.

What do you think about it?

Solution

Since docx files are zip files containing multiple files this has to do something with those files (metadata).

But what?

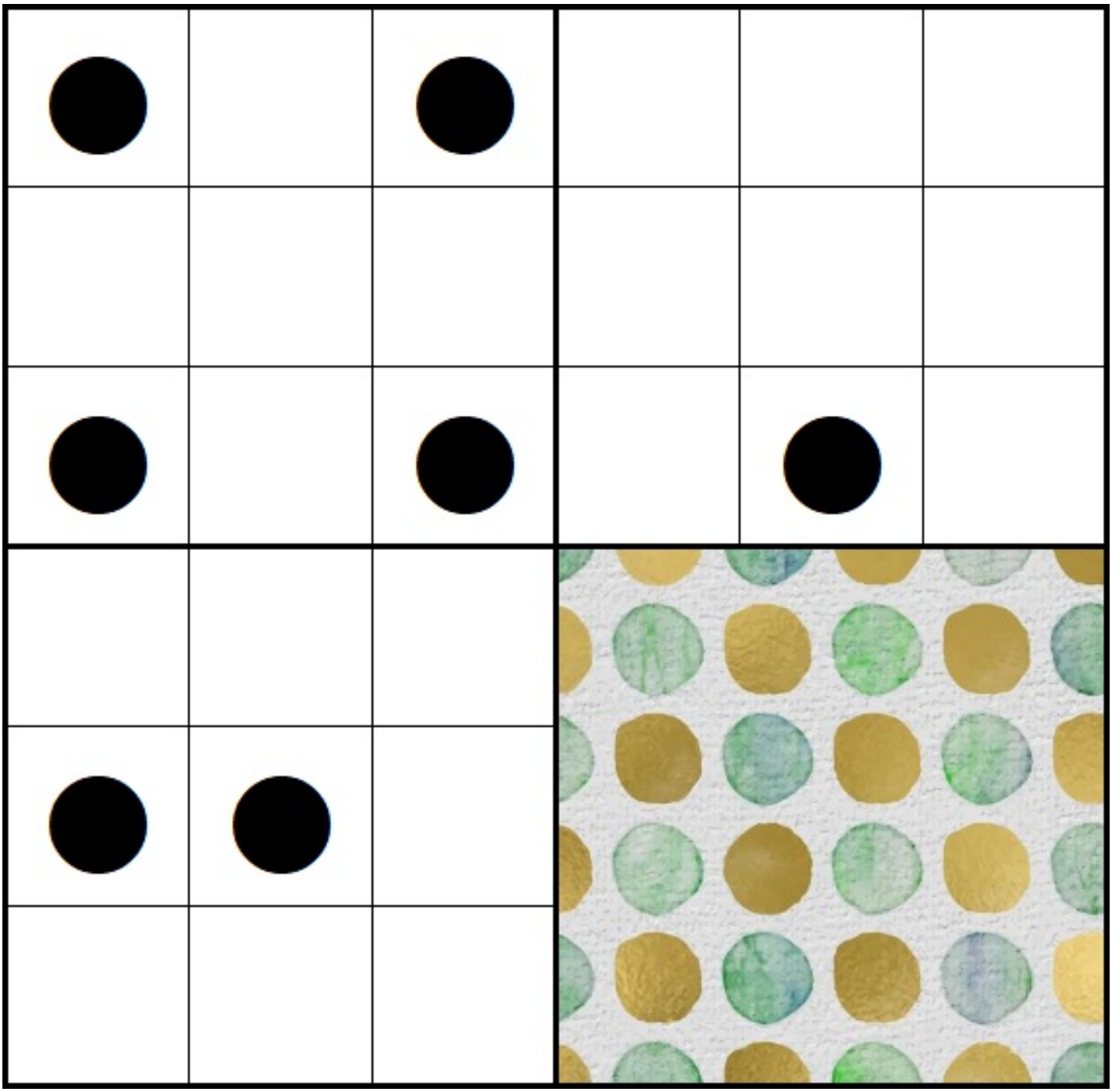
06 - Dots

Task

Uncover the dots' secret!

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

H	C	E	H	T	O
R	C	H	E	D	I
L	S	L	O	O	L
P	W	A	H	B	I
U	C	A	T	S	K
S	E	W	T	O	E



Solution

If we assume that each dot in the quadrant has to be used once, then the missing letters in the 4 quadrant are "B" and "K". Reading this sequentially this will give "HELLOBUCK", which is not the solution.

Another idea is the dots font, see https://fontmeme.com/schriftarten/3x3-dots-schriftart/. If reverse the dots this will read +m?g, for ? there is no letter. Since it's also saying that't the password is only uppercase letters, this is not the solution.

07 - Shell we Argument

Task

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

Solution

My challenge!

This is trivial challenge. You could either:

- replace eval with echo to output the bash code
- bash -x eggi.sh to debug the shell
- do it manually with much patience (5 * 13 + 5 * logn(1000)) = 130 (average)

Anyway, the right arguments are

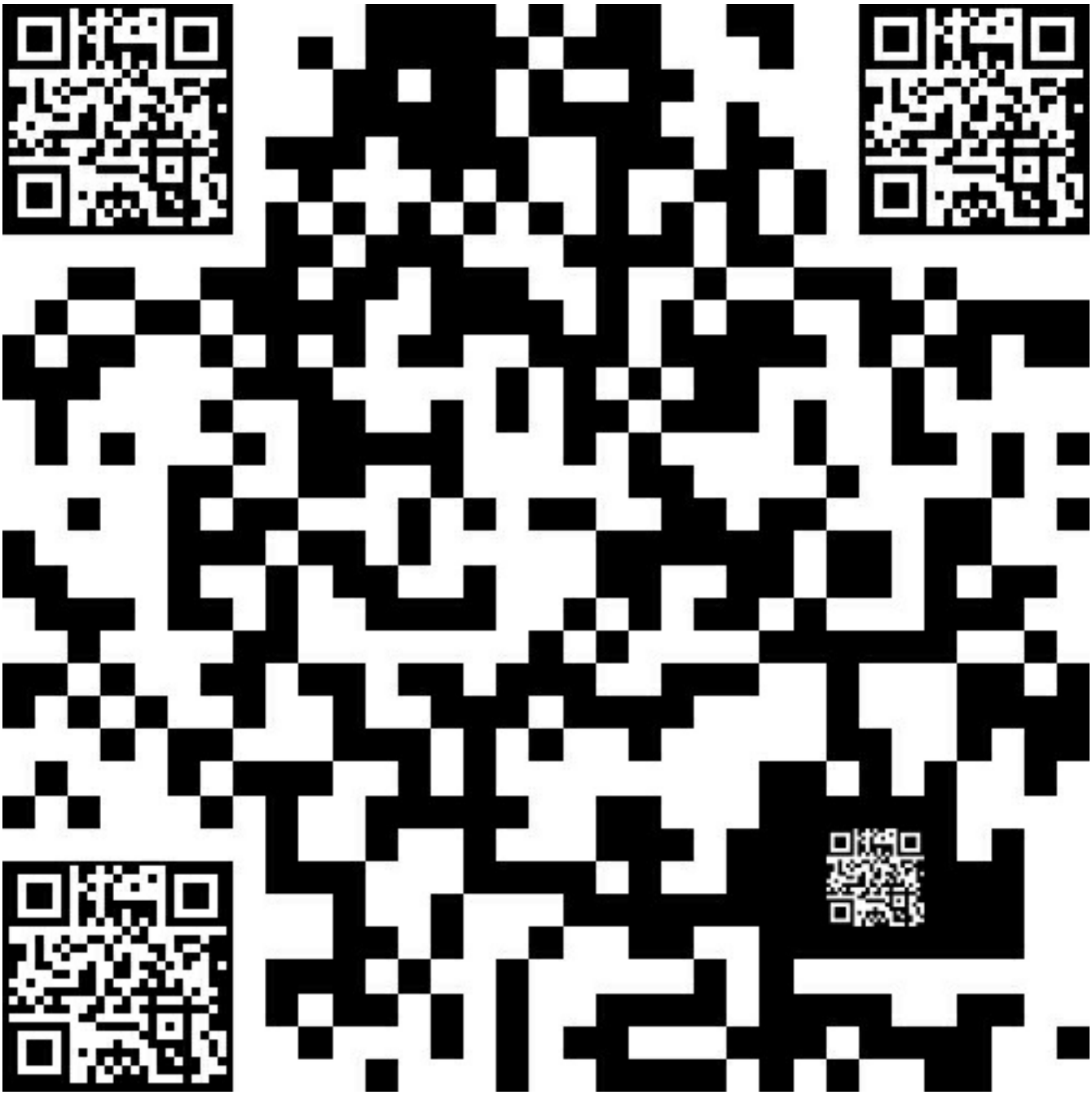
```
./eggi.sh -R 465 -a 333 -b 911 -I 112 -t 7
```

This will start a browser or issue an URL which will show the egg.

08 - Modern Art

Task

Do you like modern art?



Solution

The smaller QR codes all contains the text "remove me". This is easy done by taking the sync squares from an original QR code and pasting it into it.

This will give the text

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

Well, indeed!

Looking if there is some stego, found nothing. Looking if the smaller and larger QR codes should be xored, but no (different size).

Looking into JPEG. The file is missing a file header (FF E0) but otherwise no special marks.

09 - rorriM rorriM

Task

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

Solution

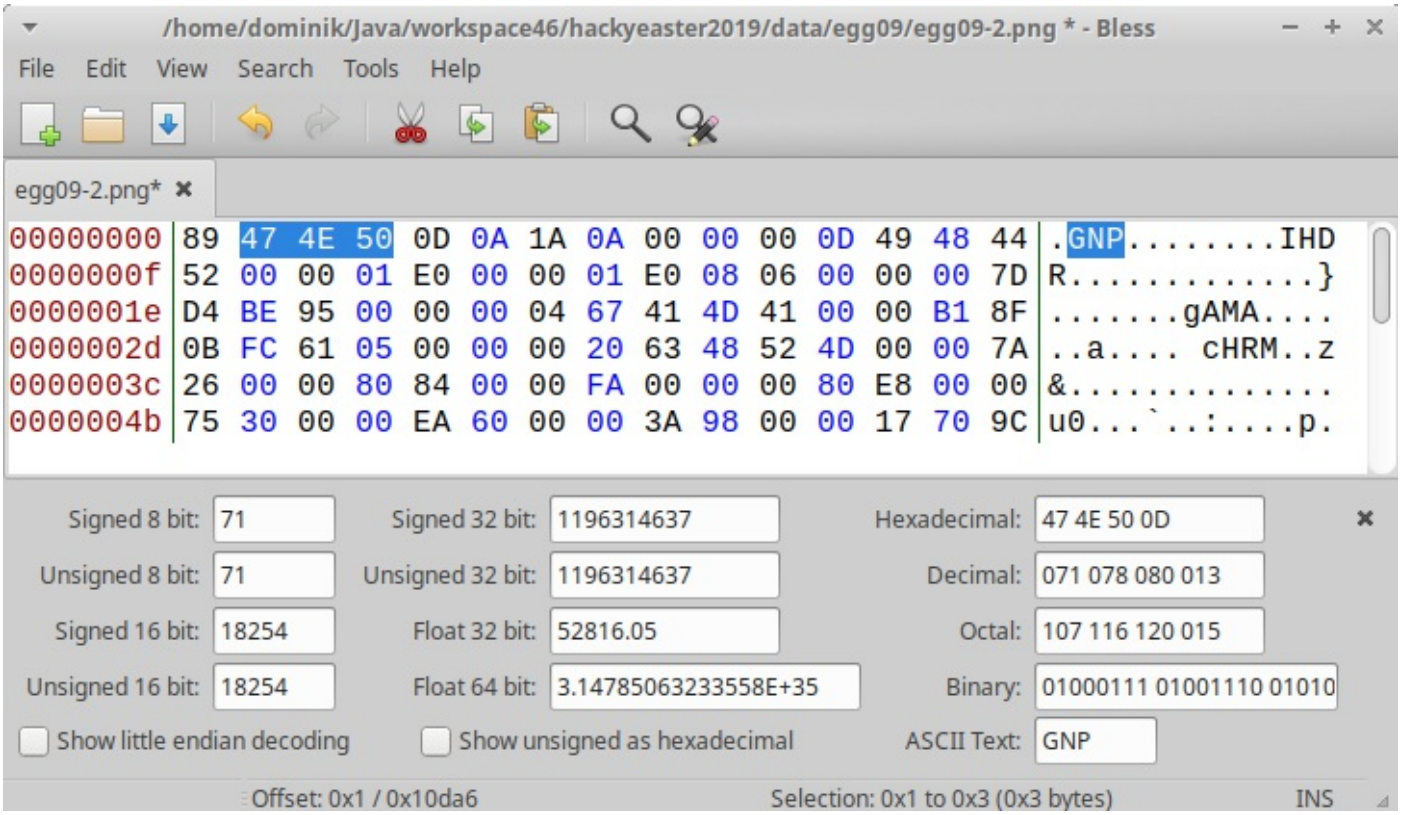
OK, seems that the file is inverted, not the first time this happens, so I have a utility method for this.

```
package egg09;

import java.io.IOException;
import util.FileUtil;

public class Reverse {
        public static void main(String[] args) throws IOException {
                FileUtil.binaryReverse("data/egg09/evihcra.piz",
"data/egg09/archive.zip");
        }
}
```

When I unarchive this, I get a new file

```
dominik@oslo:~/Java/workspace46/hackyeaster2019/data/egg09$ unzip archive.zip
Archive:  archive.zip
  inflating: 90gge.gnp
```

But this file is not completly inverted. Only the file header had to be corrected. I used bless and changed the marked bytes to "PNG":



But still, it's not readable.

For the next steps I used Gimp, doing the following:

- inverted the color of the image
- mirrored the image

This got me the egg:

10 -

Task

Solution

11 - Memeory 2.0

Task

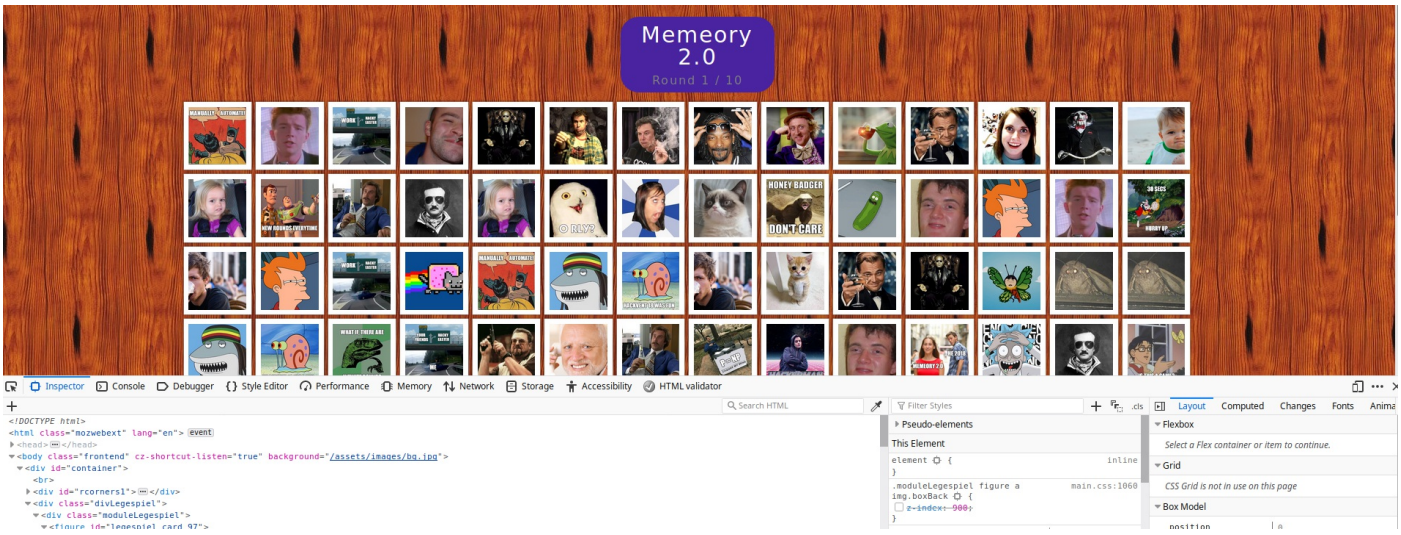
We improved Memeory 1.0 and added an insane serverside component. So, no more CSS-tricks. Muahahaha.

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

Good game.

Solution

OK, I analyzed first, whats happen. Time is very limited, there are only 3 failures and CSS tricks are still possible (disable z-index = 900).



But the time restriction stays and for this we have to automate this.

- A new game starts with: http://whale.hacking-lab.com:1111/
- Then all the images are loaded: http://whale.hacking-lab.com:1111/pic/{nr} with nr = 1..98
- A guess is issued to the server by Form POST at http://whale.hacking-lab.com:1111/solve. Form values: first={x}&second={y}

So all we need to do is to match all the files and issue the correct solve forms.

Java Code:

```
        package egg11;

        import java.io.IOException;
        import java.util.ArrayList;
        import java.util.Arrays;
        import java.util.List;

        import org.apache.http.HttpResponse;
        import org.apache.http.NameValuePair;
        import org.apache.http.client.ClientProtocolException;
        import org.apache.http.client.HttpClient;
        import org.apache.http.client.entity.UrlEncodedFormEntity;
        import org.apache.http.client.methods.HttpGet;
        import org.apache.http.client.methods.HttpPost;
        import org.apache.http.impl.client.BasicCookieStore;
        import org.apache.http.impl.client.HttpClientBuilder;
        import org.apache.http.message.BasicNameValuePair;
        import org.apache.http.util.EntityUtils;

        import util.FileUtil;

        public class PlayMemory {
                static private byte[] EMPTY = { 0 };

                public static void main(String[] args) throws
ClientProtocolException, IOException {
                        // empty array
                        BasicCookieStore cookieStore = new BasicCookieStore();
                        HttpClient client =
HttpClientBuilder.create().setDefaultCookieStore(cookieStore).build();

                        HttpGet get = new HttpGet("http://whale.hacking-
lab.com:1111/");
                        HttpPost post = new HttpPost("http://whale.hacking-
lab.com:1111/solve");

                        for (int x = 0; x < 11; x++) {
                                // get page
                                HttpResponse response = client.execute(get);
                                byte[] res2 =
EntityUtils.toByteArray(response.getEntity());
                                FileUtil.writeFileBytes("data/egg11/res_" + x,
res2);
                                if (x == 11) {
                                        break;
                                }

                                List<byte[]> files = new ArrayList<>();
                                for (int i = 1; i <= 98; i++) {
                                        HttpGet pic = new
HttpGet("http://whale.hacking-lab.com:1111/pic/" + i);
                                        response = client.execute(pic);

files.add(EntityUtils.toByteArray(response.getEntity()));
                                        System.out.println("READ pic " + i);
                                }

                                // match pics
                                for (int i = 0; i < 98; i++) {
```

```
                                        if (!(Arrays.equals(EMPTY, files.get(i))))
{
                                                // if not empty --> search for
copy
                                                int k = -1;
                                                for (int j = i + 1; j < 98; j++) {
                                                        if (!(Arrays.equals(EMPTY,
files.get(j))) && Arrays.equals(files.get(i), files.get(j))) {
                                                                // OK, found

List<NameValuePair> nvps = new ArrayList<NameValuePair>();
                                                                nvps.add(new
BasicNameValuePair("first", "" + (i + 1)));
                                                                nvps.add(new
BasicNameValuePair("second", "" + (j + 1)));
                                                                post.setEntity(new
UrlEncodedFormEntity(nvps));

                                                                HttpResponse
postResponse = client.execute(post);
                                                                String res =
EntityUtils.toString(postResponse.getEntity());

System.out.println("SET: " + i + "/" + j + ", RES: " + res);

                                                                // mark used tiles
                                                                files.set(i,
EMPTY);
                                                                files.set(j,
EMPTY);
                                                                break;
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
```

Output:

```
        READ pic 1
        READ pic 2
        READ pic 3
        ...
        SET: 70/92, RES: ok
        SET: 74/87, RES: ok
        SET: 77/79, RES: ok, here is your flag: 1-m3m3-4-d4y-k33p5-7h3-d0c70r-4w4y
```

12 -

Task

Solution

13 - Symphony in HEX

Task

A lost symphony of the genius has reappeared.

Hint: count quavers, read semibreves

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

Solution

14 -

Task

Solution

15 - Seen in Steem

Task

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

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

Solution

First I looked at the API explorer here https://steem.esteem.ws/ to see if there is a fitting method to retrieve / search for terms. But unfortunately there is no such API. With "manual binary search" I found that the blocks 21170000 - 21200000 covers the 1.4.2018, which is the Easter date.

There is also asksteem which had provided better search. Also asksteem was embedded in DuckDuckGo ("!steem asksteem search something"). But both are not running anymore.

Then I tried with https://steemlookup.com/, so that darkstar has an account there (darkstar-42). Maybe he has other accounts? Looking at https://steemit.com/@darkstar-42.

Since I found no better, I took the bruteforce way:

```
package egg15;

import java.io.IOException;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.BasicCookieStore;
import org.apache.http.impl.client.HttpClientBuilder;
```

```
import org.apache.http.util.EntityUtils;

public class ScanSteem {

        public static void main(String[] args) throws ClientProtocolException,
IOException {
                BasicCookieStore cookieStore = new BasicCookieStore();
                HttpClient client =
HttpClientBuilder.create().setDefaultCookieStore(cookieStore).build();
                for (int i = 21185000; i < 21193000; i++) {
                        HttpGet get = new
HttpGet("https://api.steemjs.com/get_block?blockNum=" + i);
                        HttpResponse response = client.execute(get);
                        String res2 = EntityUtils.toString(response.getEntity());
                        int pos = res2.indexOf("timestamp");
                        if (pos > 0) {
                                System.out.println("Trying block " + i + " / " +
res2.substring(pos, pos + 32));
                        } else {
                                System.out.println("Trying block " + i);
                        }
                        if (res2.toLowerCase().contains("hacky")) {
                                System.out.println("Found: " + res2);
                                if (!res2.toLowerCase().contains("fardelynhacky"))
{
                                        System.exit(0);
                                }
                        }
                }
        }
}
```

After a while I got a hit at block 21187964, where I found the following transaction (from darkstar-42):

```
{
        "ref_block_num": 19815,
        "ref_block_prefix": 2055191833,
        "expiration": "2018-04-01T14:49:36",
        "operations": [
                [
                        "transfer",
                        {
                                "from": "darkstar-42",
                                "to": "ctf",
                                "amount": "0.001 SBD",
                                "memo": "Hacky Easter 2019 takes place between
April and May 2019. Take a note: nomoneynobunny"
                        }
                ]
        ],
        "extensions": [],
        "signatures": [

"1f2463aa3600559e8dcb9a4712516f853723cba450f87c59d41db175dba4fe243219dfecd981ec4b2
8d92bc5b141152413c0d75afb65148611d32ebb5cf50c6580"
        ],
        "transaction_id": "94130b733fdfd6fd068b1879d4a23bca4bc78f12",
        "block_num": 21187964,
        "transaction_num": 67
}
```

So the eggomatic passphrase is **nomoneynobunny**

16 - Every-Thing

Task

After the brilliant idea from here.

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

Solution

There are at least 3 possible solutions for this challenge. You could solve it

- Interpret the dump directly and find the fitting base64 snipplets (pain in the ass)
- programmatically by accessing the DB with any programming language and loop through the hierarchy
- Using (My-)SQL directly (SQL queries and / or procedures)

This solution shows the 3nd way.

- Install MySql on your device
- Configure MySql to allow export of file (by setting `secure_file_priv="/tmp"` in `/etc/mysql/mysql.conf.d/mysqld.cnf`)
- Restart MySql
- Import the dump
- Understand the structure of the table (see **1**)
- Issue the following query:

  set group_concat_max_len = 1000000; select to_base64(group_concat(from_base64(value) separator '')) from (select ord as ord1, 0 as ord2, 0 as ord3, id, type, value, pid from Thing where type='png' and ord = 6 union all select t1.ord as ord1, t2.ord as ord2, 0 as ord3, t2.id, t2.type, t2.value, t2.pid from Thing t2 join Thing t1 on t2.pid = t1.id where t1.type='png' and t1.ord=6 union all select t1.ord as ord1, t2.ord as ord2, t3.ord as ord3, t3.id, t3.type, t3.value, t3.pid from Thing t3 join Thing t2 on t3.pid = t2.id join Thing t1 on t2.pid = t1.id where t1.type='png' and t1.ord=6 order by ord1, ord2, ord3) t where length(value) > 10 && type <> 'png' into outfile '/tmp/test.png';

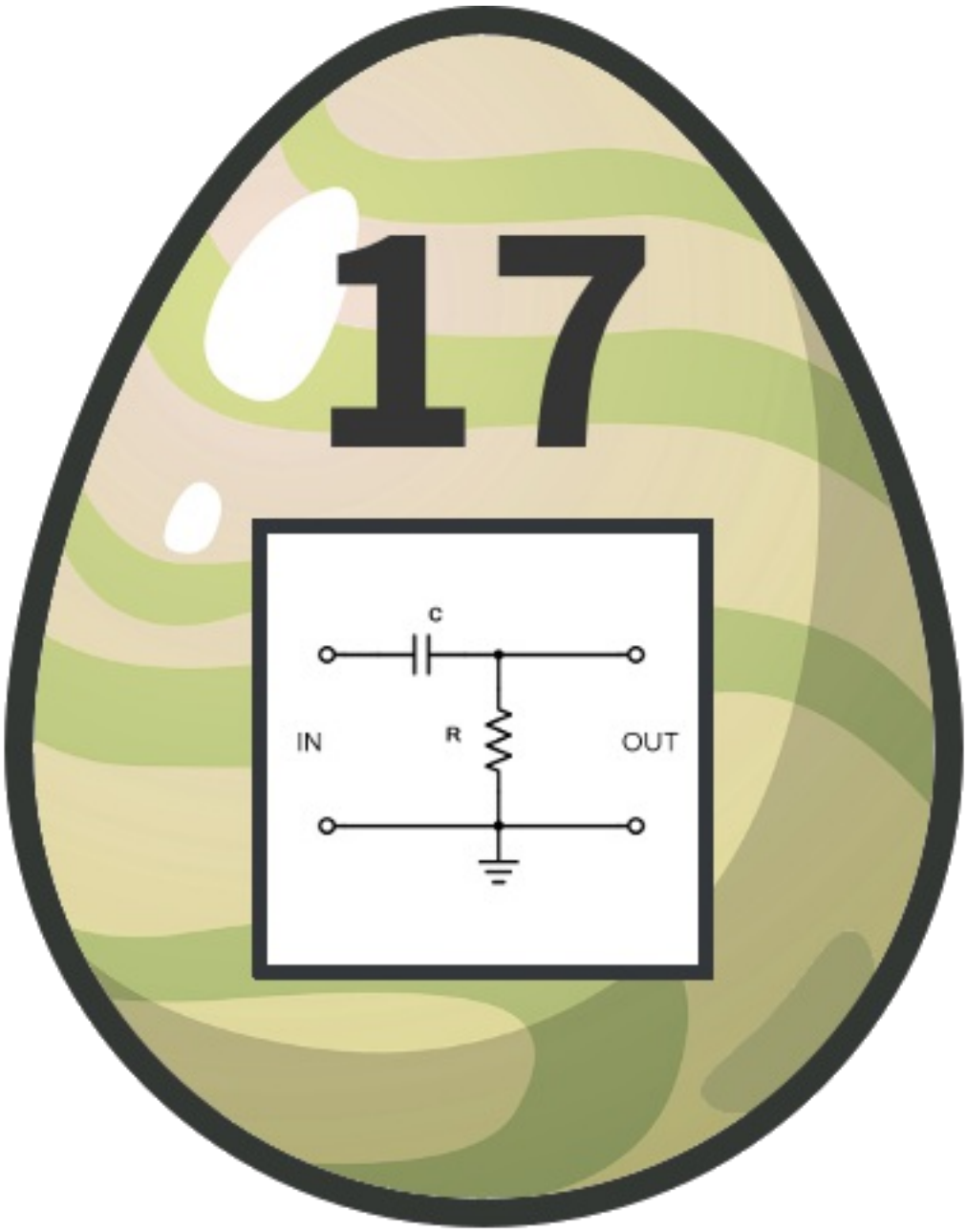
- This will export the base64 of the original png
- To convert do `cat /tmp/test.png | sed 's/\\//g' | base64 -d > test2.png`
- Scan the QR Code of test2.png

1 It's important to understand the content and structure of the table. Once you got it, you see that there's a galery with 10 PNGs (type='png'). Now the dedicated rows must be reconstructed to the original PNG. By decoding a single basd64 value you find out, that the base64 values are complete (they contain also the preamble length in front of the type and the final CRC code of the segments), so it's just a concatination job.

17 - New Egg Design

Task

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



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

Solution

A hint was issued for this challenge, saying "filter". But then, this was clear, because the circuit shows a high pass filter.

Already tried with Gimp, but no success.

18 - Egg Storage

Task

Last year someone stole some eggs from Thumper.

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

Solution

This is a web assembly task. The web assembly is a an Javascript array. First i converted this to a binary wasm file.

This can be decompiled from wasm to wat (web assembly text) at:
https://webassembly.github.io/wabt/demo/wasm2wat/

There is a wat to c converter at https://wwwg.github.io/web-wasmdec/. unfortunately this is not working for this wat, so I tried WABT (https://github.com/WebAssembly/wabt). This provides a wasm2c decompiler

```
/home/dominik/Software/wabt/build/wasm2c code.wasm -o code.c
```

which is not much better readable than the wat:

C code:

```
i0 = p0;
i1 = 84u;
i0 = i0 != i1;
if (i0) {
  i0 = 0u;
  goto Bfunc;
}
```

corresponding wat:

```
(if $I2
   (i32.ne
     (local.get $p0)
     (i32.const 84))
   (then
     (return
       (i32.const 0))))
```

So I go on reading it from wat and if I don't get the meaning I compare with the c code. This gives me:

- Passphrase must be 24 chars
- validateRange(): The chars must be in "01345HLXcdfr", validated only after position 4
- rules
  - r1: p0-p3: "Th3P"
  - r2: p17 = p23
  - r3: p12 = p16
  - r4: p22 = p15
  - r5: p5 - p7 = 14 → X/d or d/r
  - r6: p14 + 1 = p15 → 0/1 or 3/4 or 4/5 or c/d

- r7: p9 % p8 = 40 → X/0
- r8: p5 - p9 + p19 = 79
- r9: p7 - p14 = p20 →
- r10: (p9 % p4) * 2 = p13
- r11: p13 % p6 = 20 → H/4
- r12: p7 % p6 = p10
- r13: p11 % p13 = p21 - 46
- r14: p23 % p22 = 2 → 3/1 or 5/3 or f/d or d/1
- r15: sum(p4 - p23) = 1352
- r16: xor(p4 - p23) = 44

- Store
  - 0-23: fixed string: 0x66, 0x51, 0x01, 0x69, 0x50, 0x13, 0x57, 0x50, 0x03, 0x6a, 0x06, 0x07, 0x07, 0x7b, 0x05, 0x04, 0x50, 0x0b, 0x06, 0x07, 0x57, 0x7a, 0x50, 0x04
  - 24-47: The input Passphrase

- decrypt(): Just xor of store position 0-23 to position 24-47 to get the password.

Too much to brute force (20^12) ~ 4 * 10^15, but with the rules this is reduced drastically.

```
package egg18;

public class BruteforceWasmTest2 {

	public static void main(String[] args) {
		char[] s = "01345HLXcdfr".toCharArray();
		char[] c = new char[24];
		// p0-p3: "Th3P"
		c[0] = 'T';
		c[1] = 'h';
		c[2] = '3';
		c[3] = 'P';
		// OK: * p9 % p8 = 40 --> X/0
		c[9] = 'X';
		c[8] = '0';
		// OK: * p13 % p6 = 20 --> H/4
		c[13] = 'H';
		c[6] = '4';
		// OK: * (p9 % p4) * 2 = p13 -->X/4
		c[4] = '4';

		// next two together
		// * p5 - p7 = 14 --> f/X or r/d
		// * p5 - p9 + p19 = 79
		c[5] = 'r';
		c[7] = 'd';
		c[19] = '5';

		// OK p7 % p6 = p10
		c[10] = '0';

		// * p12 = p16
		for (char p12 : s) {
			c[12] = p12;
			c[16] = p12;
			// OK * p14 + 1 = p15 --> 0/1 or 3/4 or 4/5 or c/d
			String[] tmp2 = { "01", "34", "45", "cd" };
```

```
                        for (String s14 : tmp2) {
                                c[14] = s14.charAt(0);
                                c[15] = s14.charAt(1);
                                // * p22 = p15
                                c[22] = c[15];
                                // OK: * p7 - p14 = p20 --> c/3/0 or c/0/3 or
d/0/4 or d/4/0 or d/3/1 or d/1/3 or f/1/5 or f/5/1
                                String[] tmp3 = { "d/0/4", "d/4/0", "d/3/1",
"d/1/3" };
                                for (String p7 : tmp3) {
                                        // c[7] = p7.charAt(0);
                                        if (c[14] == p7.charAt(2)) {
                                                c[20] = p7.charAt(4);
                                                // OK * p23 % p22 = 2 --> 3/1 or
5/3 or f/d or d/1
                                                String[] tmp4 = { "3/1", "5/3",
"f/d", "d/1" };
                                                for (String p23 : tmp4) {
                                                        if (c[22] ==
p23.charAt(2)) {
                                                                c[23] =
p23.charAt(0);
                                                                // OK * p17 = p23
                                                                c[17] = c[23];
                                                                // * p11 % p13 =
p21 - 46
                                                                for (char p11 : s)
{
                                                                        for (char
p21 : s) {
                                                                                if
(p11 % c[13] == p21 - 46) {
c[11] = p11;
c[21] = p21;
for (char p18 : s) {
c[18] = p18;
// * sum(p4 - p23) = 1352
if (sum(c, 1352)) {
// * xor(p4 - p23) = 44
if (xor(c, 44)) {
System.out.println("Found: " + new String(c));
break;
}
}
}
                                                                                }
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
```

```
                                        }
                                }
                        }
                }
        }

        private static boolean xor(char[] c, int i) {
                int k = 0;
                for (int j = 4; j < c.length; j++) {
                        k = k ^ (int) c[j];
                }
                if (k == i) {
                        return true;
                }
                return false;
        }

        private static boolean sum(char[] c, int i) {
                int k = 0;
                for (int j = 4; j < c.length; j++) {
                        k = k + (int) c[j];
                }
                if (k == i) {
                        return true;
                }
                return false;
        }
}
```

The passphrase showing the egg is:

```
Found: Th3P4r4d0X0fcH01c3154L13
```

19 -

Task

Solution

20 -

Task

Solution

21 -

Task

Solution

22 -

Task

Solution

23 -

Task

Solution

To navigate through the maze use the following commands:

- go (north, south, west, east)
- search
- pick up
- open
- exit

Hidden command (xor42 encoded)

```
':+6H     --> exit
2+!)b72H  --> pick up
-2',H     --> open
%-b1-76*H --> go south
%-b5'16H  --> go west
%-b,-06*H --> go north
1'#0!*H   --> search
5*-#/+    --> whoami
```

24 -

Task

Solution

25 - Hidden Egg #1

Task

Solution

26 - Hidden Egg #2

Task

Solution

27 - Hidden Egg #3

Task

Solution