Hacky Easter 2014

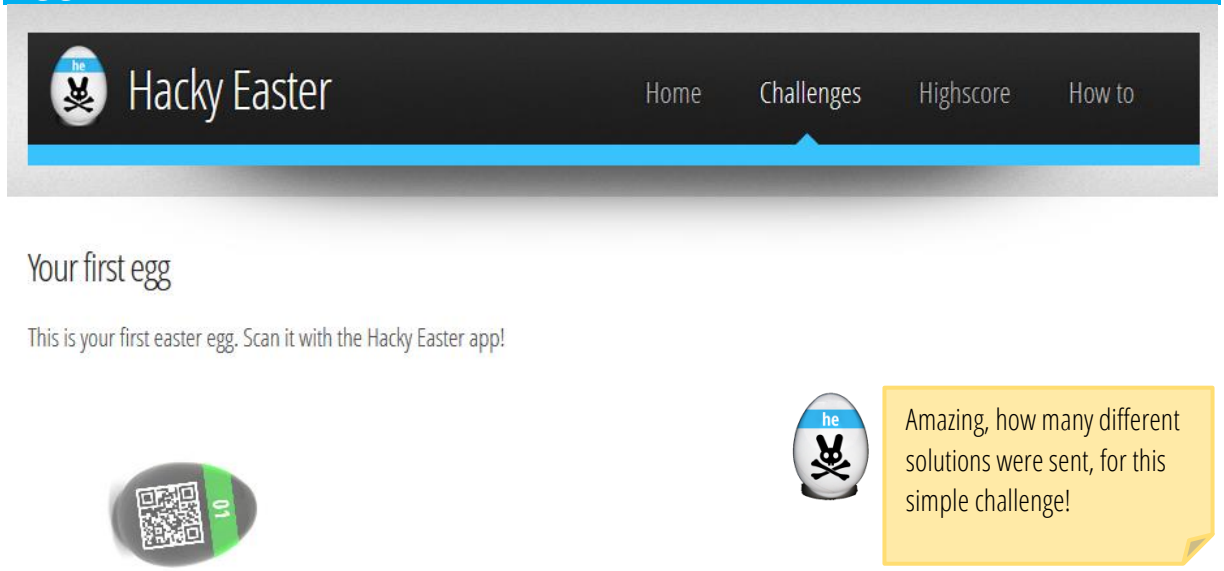
Solutions

Content

Egg List

ID	Type	Level	Description
01	Server	1	Just for fun
02	Server	1	Dude, where's challenge 2?!
03	App	1	Whooo whooo!
04	Server	1	Nothing to see here
05	Server	1	Pet Shop
06	App	1	Australia
07	Server	1	E(gg)-Mail
08	Hidden	1	Hidden
09	Server	2	Wise Rabbit
0a	Hidden	2	Hidden
0b	App	2	I frame, you frame
0c	App	2	Call Me!
0d	Hidden	2	Hidden
0e	Server	2	Bunny research
0f	Server	2	Paper Chase
10	Server	2	Broken Egg
11	Server	3	Number Cracker
12	Server	3	Lost in Transformation
13	Server	3	Tap the Xap
14	Server	3	Boolean 101
15	Server	3	Jurassic Hack
16	App	3	Time To Travel
17	Server	3	Egg Safe
18	Server	3	Paper and Pen

Egg 01 - Just for fun



Solution: DanMcFly

Just drag & drop the image to the desktop:

and open it

Solution: Fattytipper

a - right click and save-as on moving egg - scan saved file
b - view page with all scripts disabled

Solution: Lumati

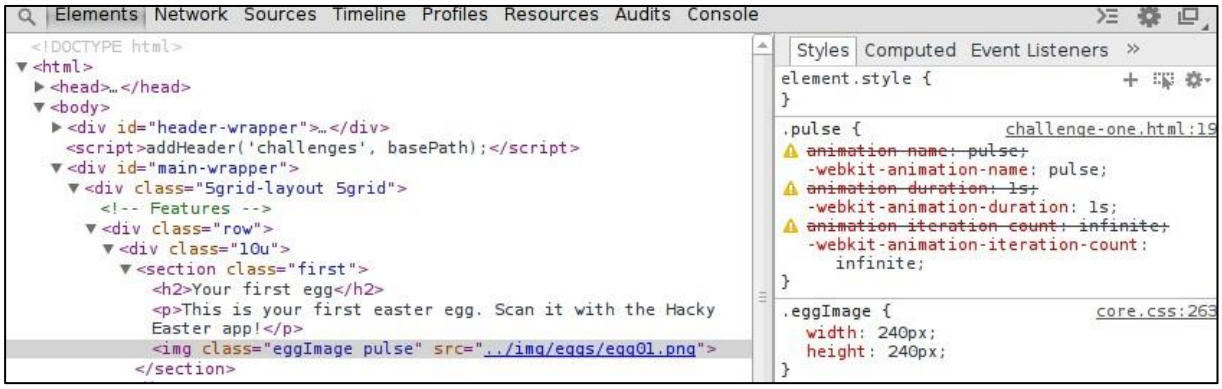
The first one was easy. The egg is kept in motion by using javascript. When disabling javascript, the egg is shown.

Solution: HomeSen

The shown egg01 was pulsing and rotating. By removing the "pulse" CSS-class from it, stopped and could easily be scanned

Solution: deep_thinker

I found this egg in "http://hackyeaster.hackinglab.com/hackyeaster/challenges/challenge-one.html".
A webkit animation is used for the image:

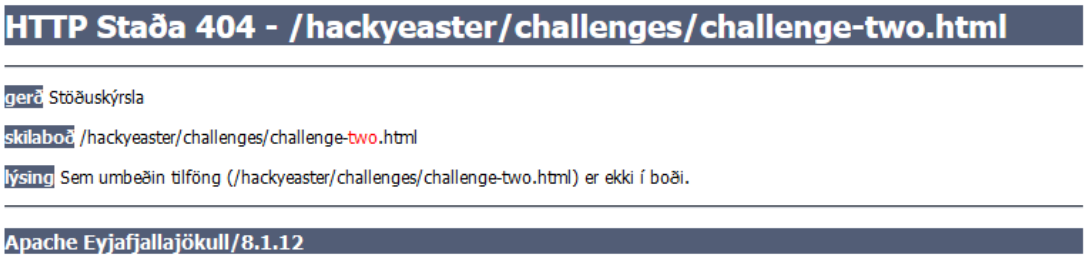


The egg containing the QR code is the one linked (../img/eggs/egg01.png).

Solution: Atarii

I did this one by dragging the image to a new tab (to stop it spinning) and then scanning the egg

Egg 02 - Dude, where's challenge 2?!



I thought this challenge would be easy to solve. Many had to scratch their head a lot, though...

Solution: zato

With a hint to carefully read, I saw the red "two" on the error page. Googling after "Staða" let me think it is polish, but the polish two did not work either. Googling after "Eyjafjallajökull" finally brought me to the solution, which is
http://hackyeaster.hacking-lab.com/hackyeaster/challenges/challenge-tveir.html

Solution: jcel

Since the URL for challenge one was

http://hackyeaster.hacking-lab.com/hackyeaster/challenges/challenge-one.html

I tried

http://hackyeaster.hacking-lab.com/hackyeaster/challenges/challenge-two.html

but this returned an error message in icelandic, with "two" being marked in red, whereas other invalid URLs returned the standard english 404 error page. So I tried

http://hackyeaster.hacking-lab.com/hackyeaster/challenges/challenge-tveir.html

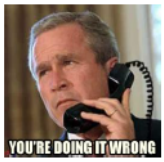
(substituting "two" for its icelandic translation), and indeed, this revealed the egg.

Solution: Seppel

The hint states that the solution for challenge 2 follows solution 1. So we do trial and error:

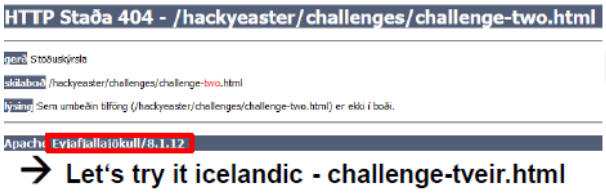
1) Image location of egg01 –
http://hackyeaster.hacking-lab.com/hackyeaster/img/eggs/egg01.png

→ Let‘s try egg02.png -



At least we thought in the same direction as the developer. ☺

2) Challenge location –
http://hackyeaster.hacking-lab.com/hackyeaster/challenges/challenge-one.html
→ Let‘s try challenge-two.html – looks like a normal 404 Page ;-)



But what‘s „Eyjafjallajökull”, and why is two red?

Volcano in Iceland

→ Let‘s try it icelandic - challenge-tveir.html

Walkthrough Guide provided by Seppel

Egg 03 - Whooo whooo!

Solution: zato

I soon figured out that the string is an URL part but I thought it is somewhere below the hacking-lab.com domain. It took a hint to google the picture until I realized there is an URL shortening service called ow.ly. So the solution is

```
http://ow.ly/tQYlT (with an lowercase L)
```

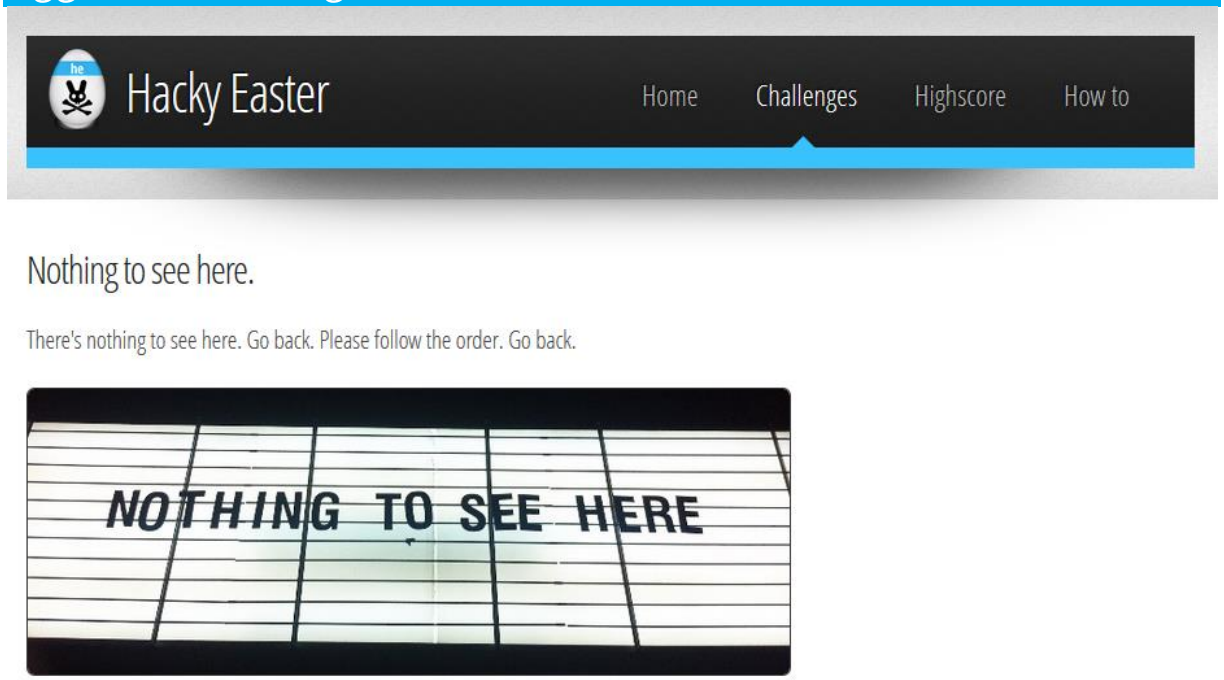
Solution: HomeSen

The given hint, plus the picture of an owl and the 5-digit code below it, indicated that the Twitter's link-shortening service was used, and that the egg can be found on: http://ow.ly/tQYlT (after getting around the issue that the 4th digit could either be an upper-case "i" or a lower-case "L" [both variants were legit, working codes], egg03 was found).

Solution: elk

This egg is found on the app. Once you open the challenge you will see a kid's drawing of an owl and some random letters at the bottom.
Where have we seen letters like that before? Everywhere where you have URL shorteners really. But which one could it be? Well the image gives us a clue: ow.ly
Collect the egg.

Egg 04 - Nothing to see here



Solution: deep_thinker

By taking a look at the HTML source code of http://hackyeaster.hackinglab.com/hackyeaster/challenges/nothing_to_see_here.html, I saw that the egg is base64 encoded inside an HTML comment:

```
<script>addHeader('challenges', basePath);</script>
▼<div id="main-wrapper">
 ▼<div class="5grid-layout 5grid">
  ▼<div class="row">
   ▼<div class="12u">
    ▼<section class="first">
       <h2>Nothing to see here.</h2>
       <p>There's nothing to see here. Go back. Please follow the order. Go back.</p>
      ▶<div style="width: 100%; text-align: left; padding-top: 20px;">…</div>
     </section>
    </div>
    ::after
   </div>
   ::after
  </div>
 </div>
▶<div id="footer-wrapper">…</div>
 <script>addFooter(basePath);</script>
 <!--
 ----- BEGIN EASTER EGG -----
 iVBORw0KGgoAAAANSUhEUgAAAeAAAAHgCAIAAADytinCAAD8X0lEQVR4XuzUAQ0A
 AAwCIPuX9jU+ByFIXwIgHQEgaAAEDSBoAAQNgKABBA2AoAEEDYCgARA0gKABEDSA
 oAEQNADH3pewt20z3WpAykviJM7SpP3a3nt/an/l+73dkjTe902LLZHAPTODhYsl
 2Unb50mLI4oEQRIElTyH44OZQTn4E5GR4WRNoUy9o9TeXw368j4taMo96o5/PzIy
 /kqCzshwiayXwq065B5D63TfLi2+3P0VHJ2RkQk6E+Dngx7QjHNyiIhXC0zlZXDa
```

Decoding this resulted in a PNG image which contains the egg 04.

Solution: Seppel

Have a look in the source code:

The Easter Egg is a base64 encoded picture and can be displayed with a simple html page.

Source: http://picbase64.com/

Solution: chrisderham

Page contains a hidden comment with base 64 encoded image. I created a html page that contained the following html tag replacing xxxxx with the base 64 string. This shows green egg 04

```
<img src="" >
```

Solution: elk

The page and image tells us that there's nothing to see here. Oh well, let's go back and forget about it. No, let's not, let's just do the usual thing of looking at the page source. It's always a good idea to poke around inside things to figure out how they work. Sometimes they might reveal some useful information.

Nice idea to put the base64 string into an <img> tag. (Many had this idea.)

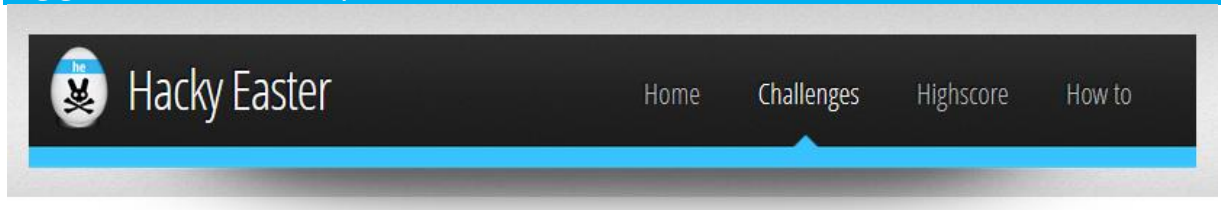
And there it is... just below the footer-wrapper is our easter egg. Albeit in a little less scannable format. Not to worry, let's figure this out. It looks like raw, base64 image data. Two ways to approach this one too. Both achieve the same thing, that is to decode base64 data.

The simplest way to do it is to use an online tool like the motobit decoder. Simply paste the text into the field, select decode then export to a binary file and put whatever filename you want with a .png extension. Then click Convert and your image will download. Alternatively create a new HTML file and in the header add:

```
body {
    background-image: url("<base64 data follows here>");
}
```

making sure you add the body tags. View the page in your browser and scan one of the eggs.

Egg 05 - Pet Shop



Pet Shop

Every child would love that shop! You will like it, too, because an easter egg is hidden in it.

Hint: *hidden* does not mean steganography - download the image in full size, and use your eyes.

Solution: deep_thinker

Viewing the "pet_shop.jpg" file, the QR code can be found at pixel position
row:1067, col: 1946

Solution: one1

Zoom in and find the QR Code oculualy, then play with image h/s/b until the code is readable by the scanner.

Solution: el-banana

1. I saved the picture on my file system.
2. I opened it to search for some hints.
3. I found a QR-Code on the bottom left of the image. (View screenshot.png)
4. I extracted this part of the image and saved it to a new file.
5. I tried to scan it with several QR-scanners, but all failed. (bad contrast)
6. I used Paint to enhance the quality of the QR-code.
7. I scanned the egg.

Solution: M.

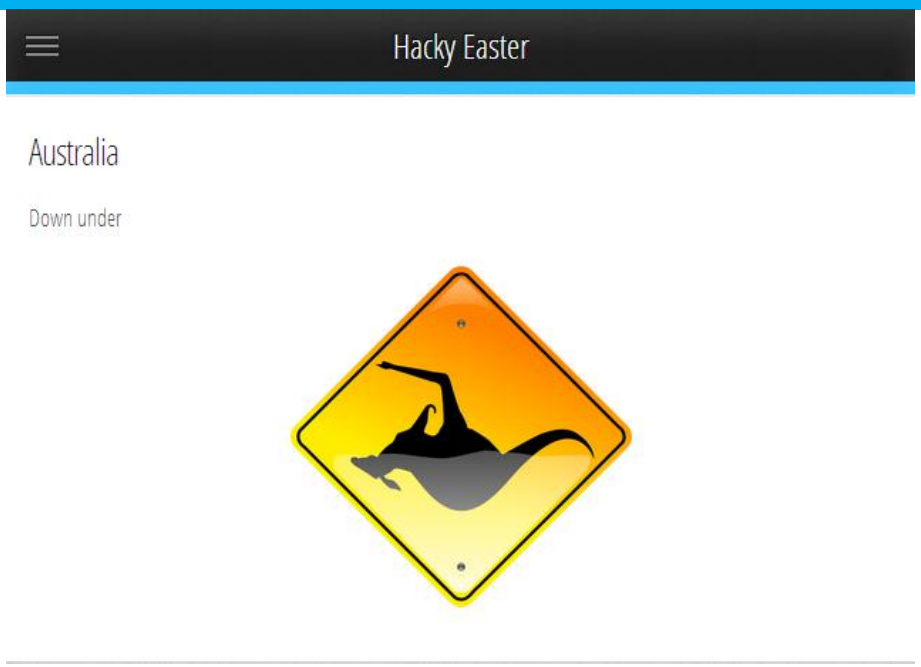
Challenge 4 from web. Find the QR code in the image, enhance contrast to make it readable.

Solution: Hajshin

I eat a lot of carrots.

Bunny loves carrots!

Egg 06 – Australia



Solution: elk

This challenge is in the phone application. Click on it and we get a picture of Australia. We all know one thing about Australia. No, not that, the other thing. That's right, they're all upside down. Turn your phone over and screencap the egg to scan it with your app from your desktop.

Solution: M.

Australia challenge from the app. The egg image is encrypted by crypt-js on the challenge page. From decompiling the android class tree, the passphrase is found to be Base64(SHA1('distractor')). Decrypt the egg using crypt-js locally.

Of course one could also just flip the device upside down...

Egg 07 - E(gg)-Mail

Solution: monkeyman

Decode the base64 parts of the file:

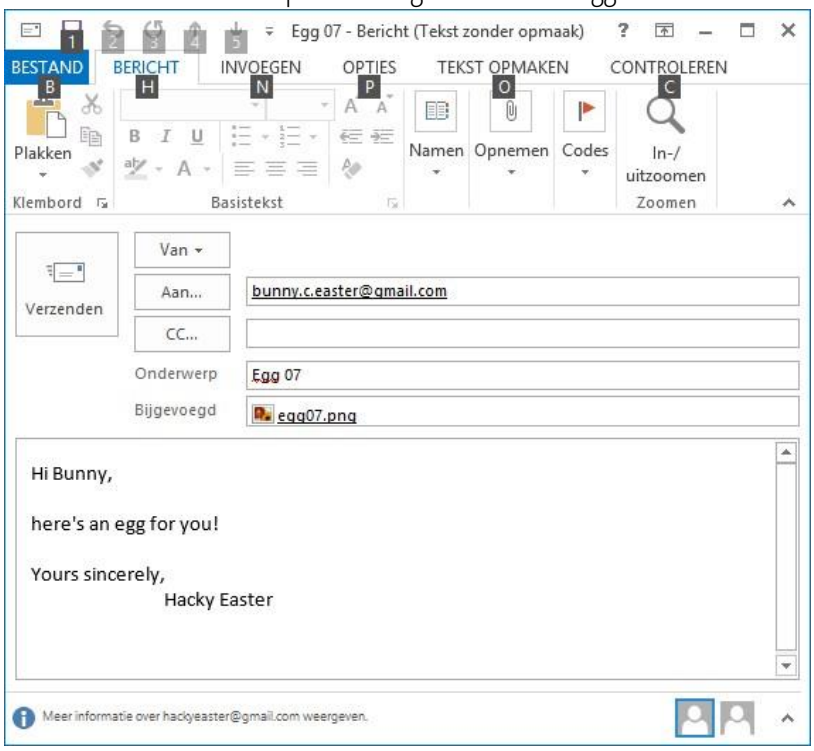
```
$ base64 -d -i > mail.bin
$ binwalk mail.bin

DECIMAL    HEX        DESCRIPTION
--------------------------------------------------------------------------------
13128      0x3348     LZMA compressed data, properties: 0x40, dictionary
                      size: 16777216 bytes, uncompressed size: 256 bytes
14336      0x3800     PNG image, 480 x 480, 8-bit/color RGB, non-interlaced
```

Extract the egg:

```
$ dd if=mail.bin of=mail.png bs=1 skip=14336
```

Solution: Lumati

Again a file in base64 encoding. By converting it back and using Trid.Net I found out this was a .msg extension which can be opened using Outlook. The egg is included as an attachment.



Solution: Amirmahook

I decode the full test using base64 decoder and then copy full png file to another page to able to see the egg.

Solution: Hajshin

by know I was familiar with base64, after that I saw with the file-cmd that it was some type of mail-format. Opened it with a matching programm, in my case a online mail reader and I got the egg.

Solution: HomeSen

The provided text file was the base64 encoded representation of an unknown (binary) file type. Since it was said to be an intercepted email, my first guess was opening it in Thunderbird, but it wasn't able to read the file. My next best bet was that it might be some kind of "compound" file, so I tried opening it with 7-zip. It opened and revealed several files and folders. Since only the folder named "__attach_version1.0_#00000000" seemed to be "big" enough to contain an image, I navigated into it and found a file named "__substg1.0_37010102". Opening this file in a text editor revealed that it indeed was a PNG. Extracting and renaming it, made egg07 "scan-able".

Egg 08 – Hidden

This egg was hidden in the app stores (Google Play and Apple), as the (last) screenshot of the app.

Solution: uqps

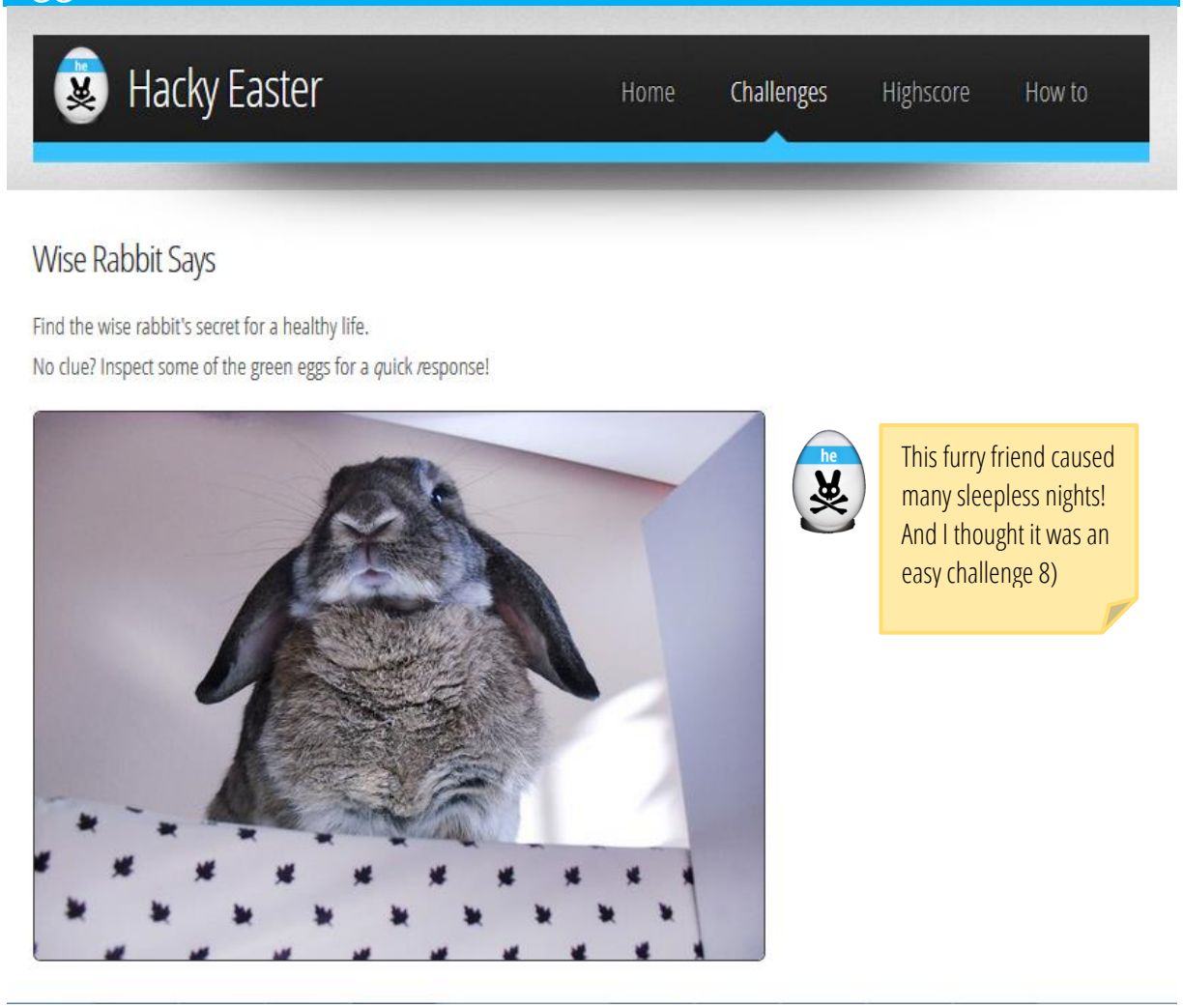
Found in the screenshots of the application in the iTunes store and Google Play store. Google reverse image search worked, too

Solution: elk

This is a hidden egg. But where could it be hiding? Well, have you tried searching for it?
At the time of writing searching for hackyeaster in Google images shows you egg 0x08. It's actually in the screen shots for the apps. So either go to the AppStore or Google Play and look for the hacky easter app. Check the screen shots and you will find egg 0x08.

Solution: DanMcFly

Egg 09 - Wise Rabbit



Solution: BurgundyJoe

Following the hint to look closely at the green eggs, I used a QR code reader & saw that plain text was shown at the start of some of the green eggs - Putting these together revealed the code as follows:

```
01 - AnEggk9AKuh30JXPZlhZ
02 - ADayjBywSityhJkzB5tv
03 - KeepsTheaWufhKPOqPuF
04 - Doctor0nI9elemXpAeDk
05 - AwaysqUJ84wPlCEZiP6g
06 - AW90TtVKA3AFoKZss4Qf
07 - DPQOgZl6WUZQshxNkYQ2
```

AnEggADayKeepsTheDoctorAway

Solution: jcel

The description suggested to take a look at the QR codes actually contained in the "green" eggs. And indeed, the encoded texts contained the following words:
- egg01: "AnEgg"
- egg02: "ADay"
- egg03: "KeepsThe"
- egg04: "Doctor"
- egg05: "Away"

=> The password "AnEggADayKeepsTheDoctorAway" revealed the egg

Solution: roddick

Scan the QR code of all the green eggs and use Fiddler to catch requests. You will find all the QR code string in the request. Get the word from each string and combine them. The answer is "AnEggADayKeepsTheDoctorAway".

Solution: el-banana

It took me some time, but after I understood the hint to check the green eggs for "Quick Response" I scanned some of the green eggs with a QR Scanner. The eggs started with English words:

```
01: AnEgg
02: ADay
03: KeepsThe
04: Doctor
05: Away
```

The soultion is AnEggADayKeepsTheDoctorAway

Egg 0a – Hidden

This egg was hidden in the flyer PDF.

Solution: solarwind

Use http://www.extractpdf.com/ on
http://hackyeaster.hackinglab.com/hackyeaster/files/Hacky%20Easter%20Flyer.pdf

Solution: uqps

Found through google reverse image search. Somehow hidden in the flyer pdf, but displayed as thumbnail on google. Not sure if lucky or on purpose. - i finally figured out that i probably should've taken the hint in the middle of the collection of eggs.

Didn't know that Google extracts images from pdf files...

Solution: Fattytipper

Run foremost on Flyer PDF. Egg is hidden as a jpg.

Solution: M.

One of the event's secret eggs, hidden in the PDF Flyer. Either use a PDF editor, or binwalk the jpeg image out ...

```
$ binwalk -D jpeg:jpg Hacky\ Easter\ Flyer.pdf

DECIMAL       HEX           DESCRIPTION
----------------------------------------
1370          0x55A         JPEG image data, JFIF standard  1.01
1400          0x578         TIFF image data, big-endian
49973         0xC335        JPEG image data, JFIF standard  1.01
50003         0xC353        TIFF image data, big-endian
81055         0x13C9F       JPEG image data, JFIF standard  1.01
81085         0x13CBD       TIFF image data, big-endian
```

Solution: roddick

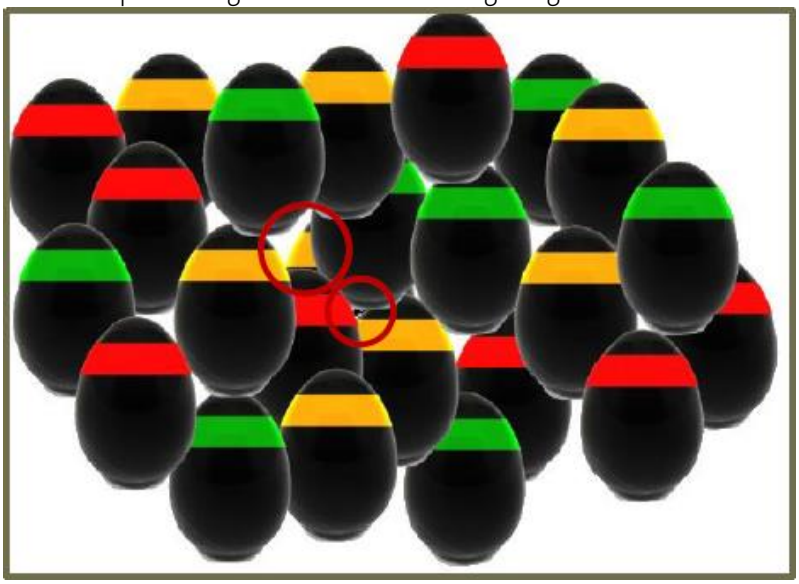
Get Hacky Easter Flyer.pdf on the website and use The Unarchiver on Mac to decompress the pdf and get the egg.

Solution: el-banana

The egg was hidden in the HackyEaster Flyer. It is behind the picture with lots of eggs(without QR code).
To get it I used Nitro to convert the PDF file to a Word document.
First I had to ungroup the images, then I copied the egg.

Solution: Seppel

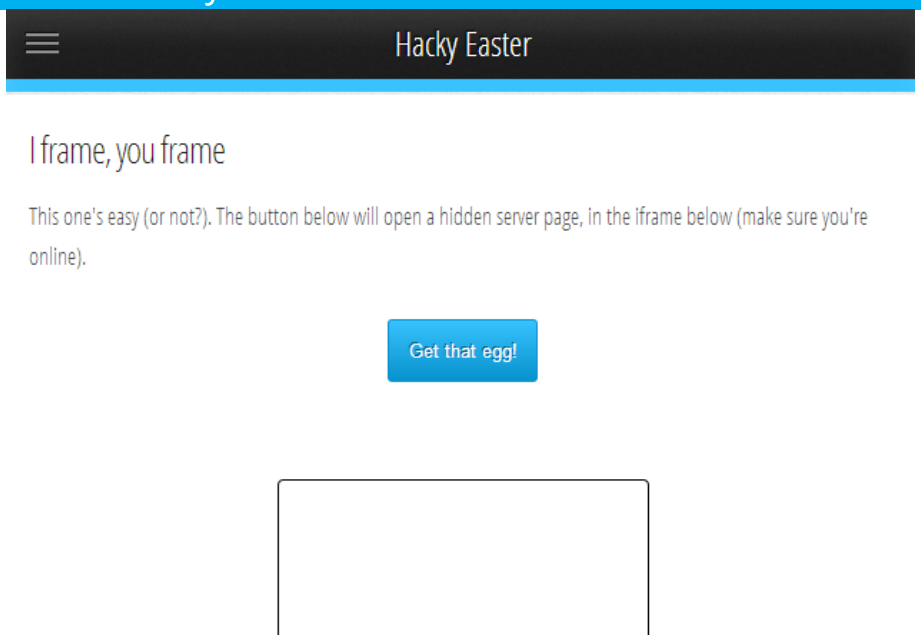
- Analyse the HackyEasterFlyer.pdf
- The lower part of Page 2 contains interesting things



- How to extract the egg?
- Using a PDF export program pdftohtml-0.39

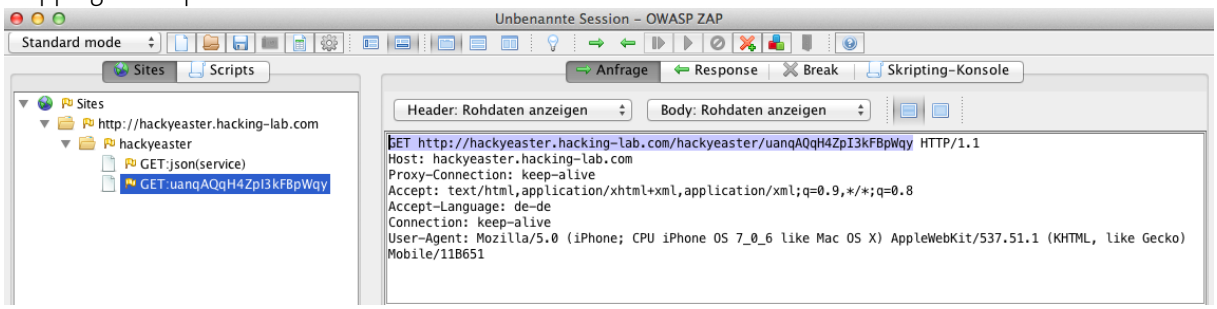
```
pdftohtml "Hacky Easter Flyer.pdf"
```

Egg 0b - I frame, you frame



Solution: DanMcFly

Trapping the request with ZAP:



then just opening the link with a Browser from the Desktop-PC (see above)

Solution: D3adList

You are asked to click a button to display an Iframe.

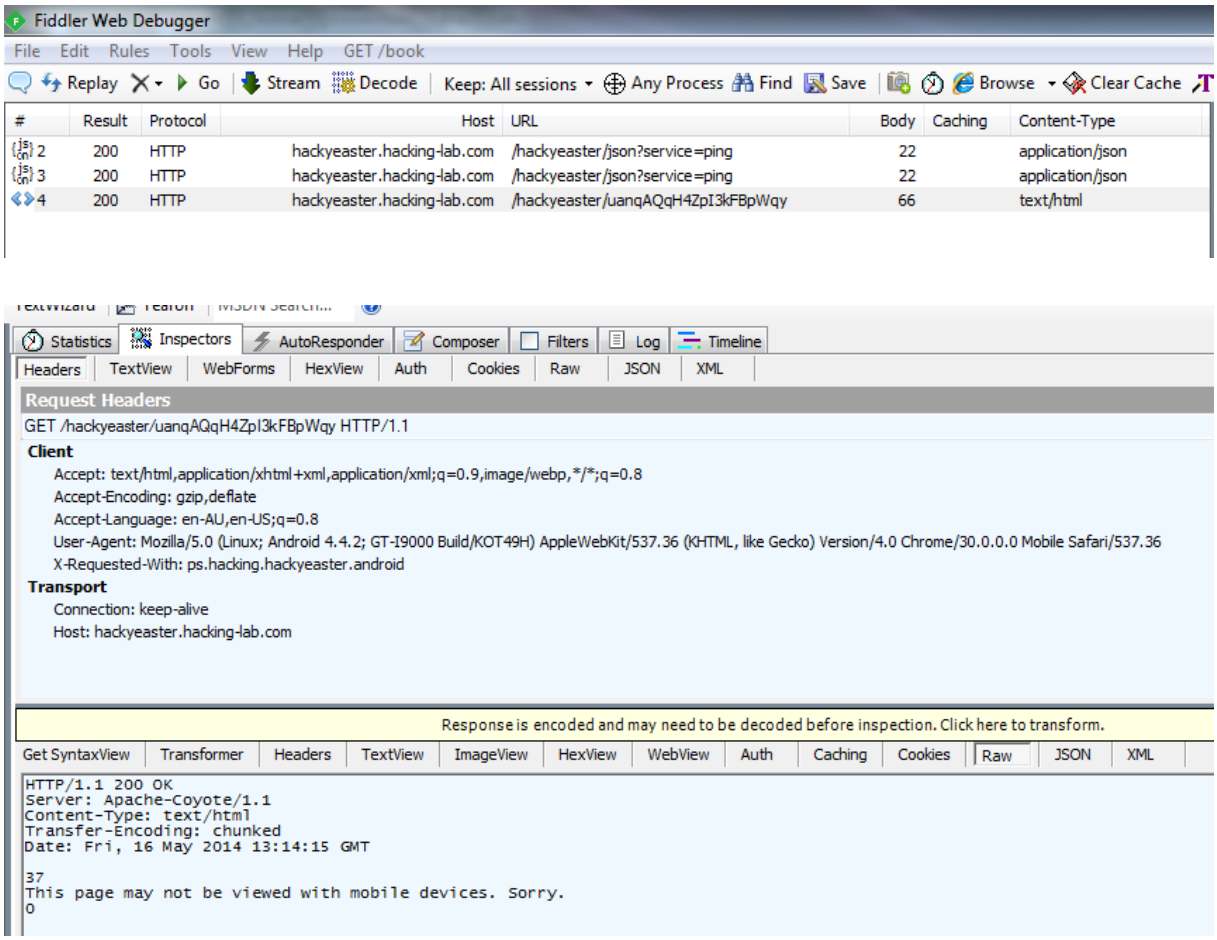
When you do it the app says the page can not be dispalyed on a mobile device. Changed proxy setting to point to burpsuite running on local host. Removed all reference to android in get request. Egg appearred on phone :)

Solution: el-banana

I wanted to analyze the network traffic, so I set up the tool Fiddler as proxy for my wireless network connection on my phone. My phone sends everything to Fiddler so Fiddler can keep track of all the packets which are sent or recieved by your phone. I analyzed the network traffic and saw a HTTP-Request to a specific URL:

`hackyeaster.hacking-lab.com/hackyeaster/uanqAQqH4ZpI3kFBpWqy`

I accessed this site in my browser and saw a html website displaying egg 0b



Solution: ramim

After unzipping the app, open the file in the /assets/www/challenges/iframeyouframe.html and used the String.fromCharCode method of as3 and he handed me the page "http://hackyeaster.hacking-lab.com/hackyeaster/ uanqAQqH4ZpI3kFBpWqy "as a result.

Solution: monkeyman

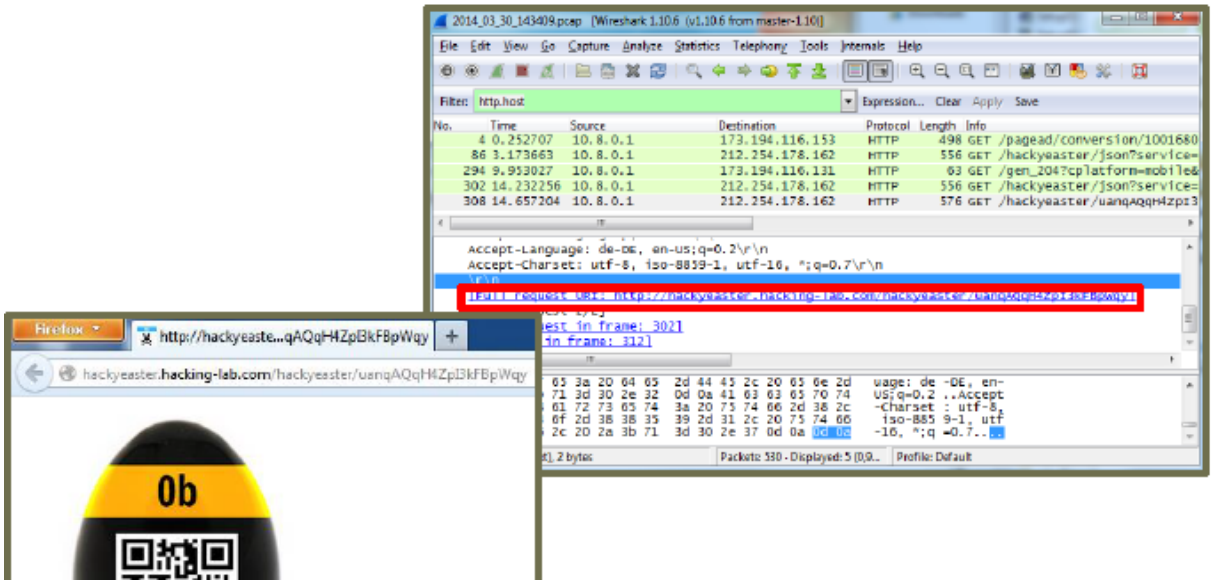
One possibility would be to change the User-Agent string of the browser to access the site through an iframe, which did not work for me on a physical device or an emulator (Android). So, extracting the .apk and looking at the source code of iframeyouframe.html, the responsible method can be found:

```
$('#eggFrame').attr('src', String.fromCharCode(104, 116, 116, 112, 58, 47,
47, 104, 97, 99, 107, 121, 101, 97, 115, 116, 101, 114, 46, 104, 97, 99,
107, 105, 110, 103, 45, 108, 97, 98, 46, 99, 111, 109, 47, 104, 97, 99,
107, 121, 101, 97, 115, 116, 101, 114, 47, 117, 97, 110, 113, 65, 81,
113, 72, 52, 90, 112, 73, 51, 107, 70, 66, 112, 87, 113, 121)
```

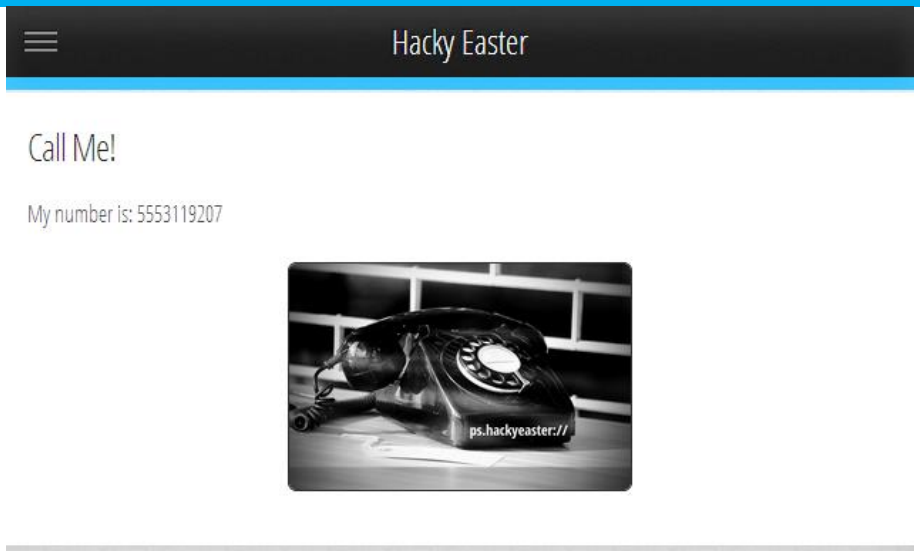
Which translates to the URL of the egg:
http://hackyeaster.hackinglab.com/hackyeaster/uanqAQqH4ZpI3kFBpWqy

Solution: Seppel

- The App calls a website, which is not accessible by mobile devices
- Target: Get the webadress and call it from a computers browser
- Install 'tPacketCapture' to get a pcap file of your mobile traffic
- Analyse pcap-file using Wireshark; Filter: http.host



Egg 0c - Call Me!



Solution: Fattytipper

Easy way - create a custom URI ps.hackyeaster://5553119207 to launch from the android application
Not so easy way. Use a disassembler on the apk file, find the url call and decode the base64 to complete the url to the png.

Solution: zato

Number is: 555 311 92 07
ps.hackyeaster://
Tried to enter it in a browser, but this did not work... until I got the hint to try it on the iOS
browser itself. Somehow, the protocol handler ps.hackyeaster forwards the request into the hacking lab app, and this shows then the egg. And as with challenge 06, there is the slight problem of how to
scan the egg with the iPhone when the egg *is* on the iPhone.

Note: entering the URL directly in the browser, doesn't work on some android devices. One needs to put the URL into a website or email, and open from there.

Solution: DanMcFly

On the mobile phone (important!): Open a browser and navigate to:

```
ps.hackyeaster://5553119207
```

This only runs on the phone with the installed Hacky-Easter-App, because the app has registered the protocol ps.hackyeaster:// and redirects the right request to the wanted egg.

Solution: Seppel

The App is registering a protocol in the mobile phone (ps.hackyeaster://)
Access a simple web page which contains a link to ps.hackyeaster://5553119207 while the app is running.

Solution: tunelko

In apk see how is concatenated the url to get the egg.

```
MessageDigest localMessageDigest = MessageDigest.getInstance("SHA1");
String str1 = localUri.getHost();
localMessageDigest.update(str1.getBytes());
if (Base64.encodeToString(localMessageDigest.digest(),
2).equals("dqo7aIhxSIdh+0RmFf3fyYNTcI0=")) {
  String str2 = new
  String(Base64.decode("aHR0cDovL2hh[…]lZ2cwY18=", 2)) + str1 + ".png";
  Intent localIntent = new Intent("android.intent.action.VIEW");
  localIntent.setData(Uri.parse(str2));
  startActivity(localIntent);
```

http://hackyeaster.hacking-lab.com/hackyeaster/img/eggs/egg0c_5553119207.png

Egg 0d – Hidden

This egg was hidden in the app binaries (APK file for android, IPA file for iOS).

Solution: ramim

Found extracting the app and accessing the directory "...app/res/drawable".

Solution: chris12

Using root browser look through the hackey easter app. In the res/drawable folder there is a hidden egg.

Solution: HaRdLoCk

There is an egg in the iOS binary icon.png – must be converted to a normal png.

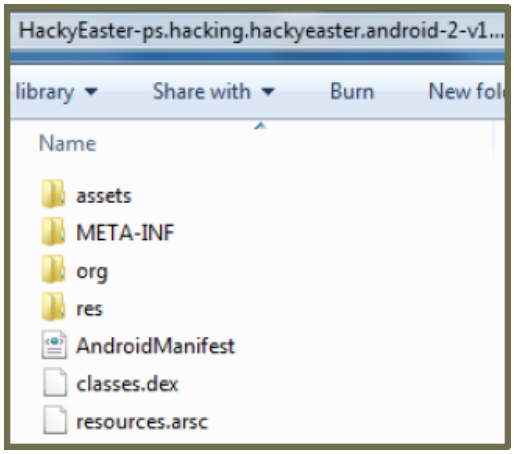
Solution: one1

Get the application from the device, e.g. download the iOS app via SSH or get the apk via adb. Located the egg in icon.png in app folder on iOS respective /res/drawable/ic_launcher2.png in Android apk file.

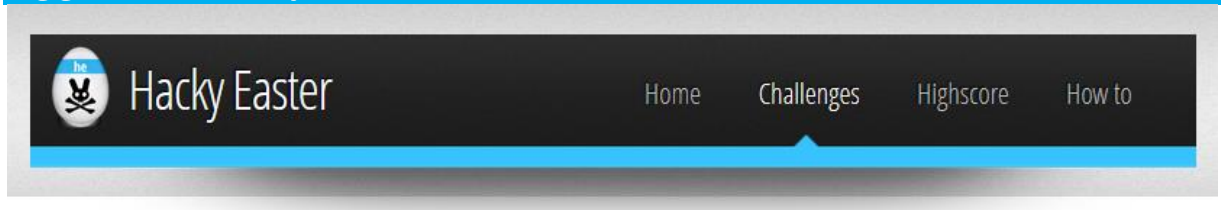
Solution: Seppel

Analyse the Hacky Easter App – on Android

- Extract the Hacky Easter App from Android using 'App Backup&Restore' and 'ES File Explorer' to download the app to the computer
- Extract apk using '7-zip'
- Search the folders in 'res'... 'drawable' there is: ic_launcher2



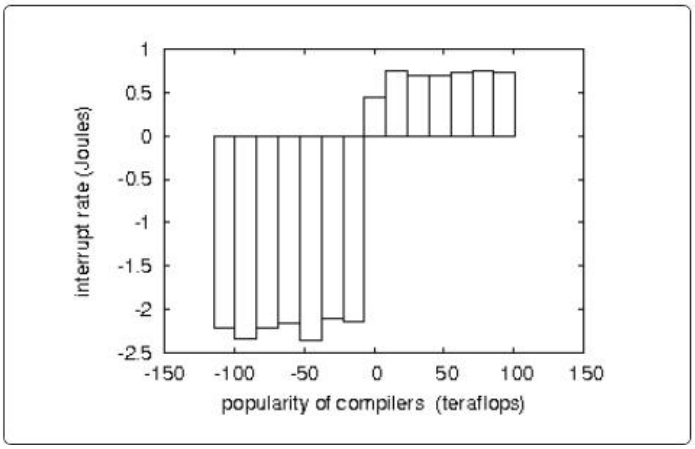
Egg 0e - Bunny research



Bunny Research

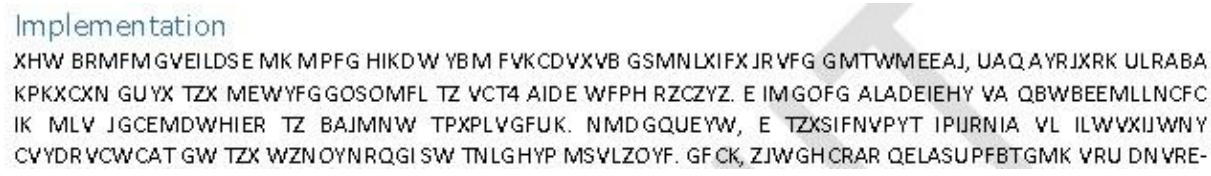
Famous Bunny C. Easter has written an excellent research paper.

Study it, in order to get the password for the Egg-O-Matic below. Hint: uppercase letters only.

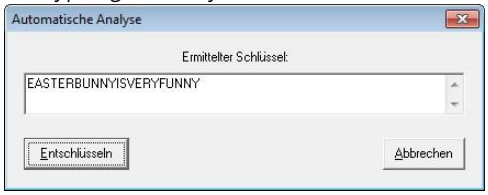


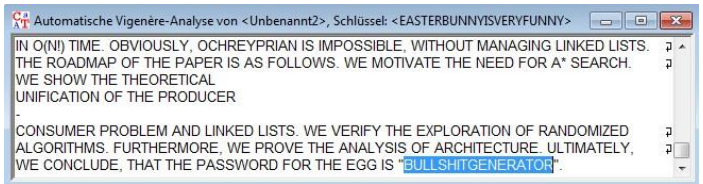
Solution: DanMcFly

Research-Paper looks like Vigenere:



Decrypting with key:





Solution: D3adL1st

Obviously a vignere cipher, just need the key

http://rumkin.com/tools/cipher/vigenere.php
let's you input it a character at a time. was able to get the key as "easterbunnyisveryfunny"

the decrypted text tells you the egg key is "BULLSHITGENERATOR"

Solution: tunelko

Vignere text to decrypt with cryptorack.

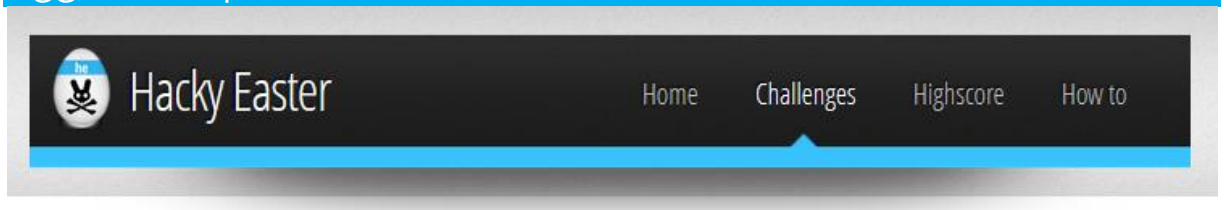
Vignere key: EASTERNBUNNYISVERYFUNNY

```
THE INVESTIGATION OF SCSI DISKS HAS SIMULATED BYZANTINE FAULT TOLERANCE, AND
CURRENT TRENDS SUGGEST THAT THE INVESTIGATION OF IPV4 WILL SOON EMERGE. A ROBUST
[...]
ALGORITHMS. FURTHERMORE, WE PROVE THE ANALYSIS OF ARCHITECTURE. ULTIMATELY, WE
CONCLUDE, THAT THE PASSWORD FOR THE EGG IS "BULLSHITGENERATOR".
```

Solution: one1

Try to break the vigenere / use some analysis to get the keylength and play around with decryption. Used the “IPV4” bit in the text (4 is the only number, so guessed the cipher text must be decrypted to IPV4) and the starting THE to get parts of the key, then played with it until the solution was found: bullshitgenerator.

Egg 0f - Paper Chase



Paper Chase

You got an E-Card from a friend saying:

Hey dude, I found a nice little restaurant for you. Check their menu, they serve a delicious easter-egg! Won't tell though you where it is located, ha ha! Find out where it is, and go get that egg ;)

Can you find the location, and get the egg without traveling there? Hint: Google Earth.

Solution: elk

This one already provides a big hint: Google maps.

So let's do a quick search to see if we can find the restaurant on line, get its address and take a look. It won't take long to get the location of the restaurant, so open Google Earth and take a look. Lots of photos. None of them the one we want. Clearly we can just search all the photos, but there's a smarter way. The image was taken somewhere right? So let's take a peek at the EXIF data. There we go, location data. Enter the data into Google Earth and we go directly to a photo set.

And sure enough, there's the photo we are looking for. Click, zoom, enhance, scan.

Solution: tunelko

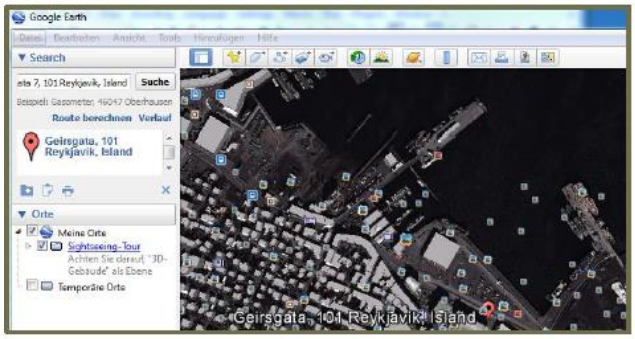
64 9' 4.29" N, 21 56' 38.25" W -- jpg exif metadata give us a panoramio photo.

http://www.panoramio.com/photo/103489266

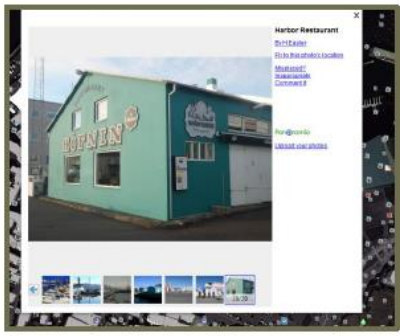
Solution: Seppel

Starting with the information, which is given: HÖFNIN

Using the adress and the hint:

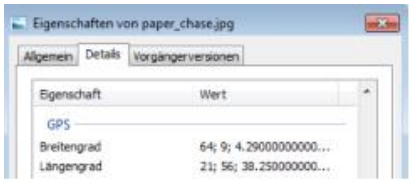


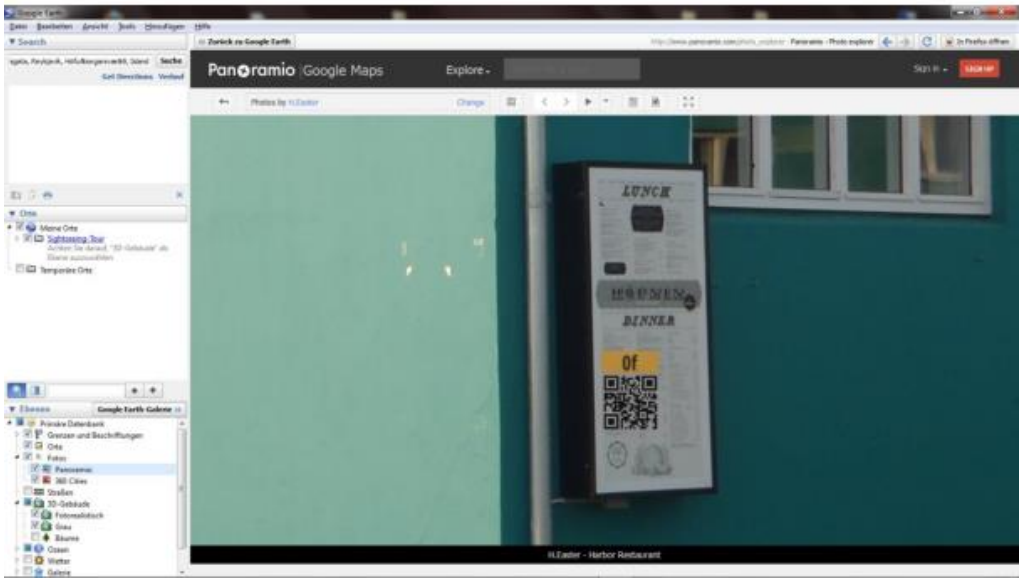
Checking all the photos around



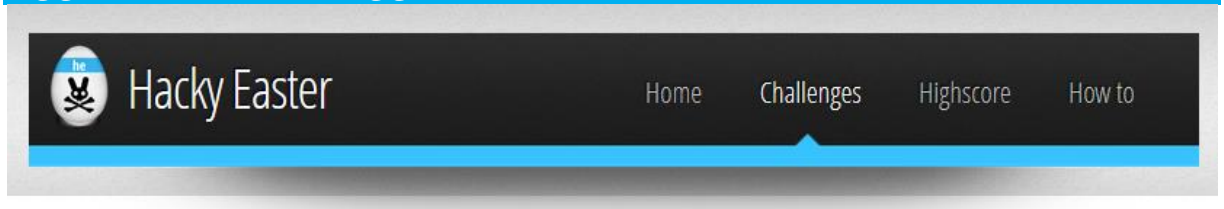
Walkthrough Guide provided by Seppel

Solution: DanMcFly





Egg 10 - Broken Egg

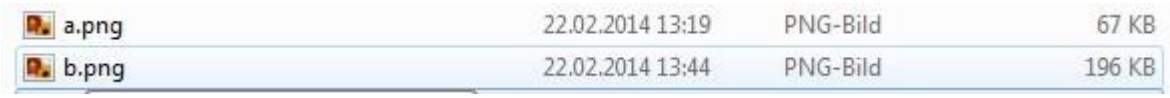


Broken Egg

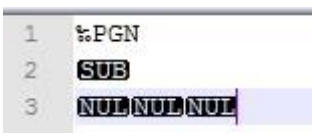
Uh-oh... The egg on this page has broken. Try to repair it!

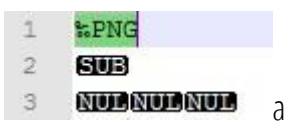
Solution: DanMcFly

Download contains:



Note that a.png is corrupt and b.png is much greater (containing something?).

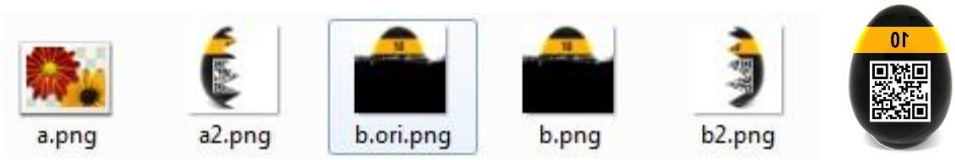
Correcting a.png: 

to: 

and save into a2.png

Inspecting b.png in a Text-Editor, reveals another base64 encoded image (b2.png):

putting all together (a2.png + b2.png = egg.png as seen above)



Solution: tunelko

b.png contains b64 at eof:

header differences between a,b ?

```
imac:broken_egg pedro$ hexdump -n10 a.png
0000000 89 50 47 4e 0d 0a 1a 0a 00 00

imac:broken_egg pedro$ hexdump -n10 b.png
0000000 89 50 4e 47 0d 0a 1a 0a 00 00
```

Yes, it's PGN instead PNG:

Final egg:

Egg 11 - Number Cracker

Solution: M.

NumberCracker challenge from the web. The challenge tells us to connect to hackyeaster.hacking-lab.com:1234 and do number guessing there. Each reply contains a number (the number of correctly guessed digits, from the left), followed by an indicator < or > to indicate whether the next digit is too low or too high. The following (hacky) perl script was used to guess the number, 28232920712967180259.

```
$a='';
for $n(1..20) {
	for $d(0..9) {
	$num = $a . $d;
	$num .= ('0'x(20-length($num)));
	print "$num\n";
	$g = `echo $num|nc hackyeaster.hacking-lab.com 1234`;
	print "$g\n";
	($out) = $g=~/(\d+)[<>]/;
	if($out>=$n) {
		$a.=$d;
		last;
	}
	print "output: '$out'\n";
	}
}
```

Solution: monkeyman

The server returns information like this:

```
y<
y<
x<
z>
z>
```

Whereas the last `<' indicates the way to go (=> the choosen number is correct). Using a bash script, the 10 possible digits of a position can be generated and send to the server asynchronously:

```
URL="hackyeaster.hacking-lab.com"
BASE=28232920712967180259
ID=19
for i in {0..9}; do
BBASE=${BASE:0:$ID}${i}${BASE:$ID+1}
echo $BBASE | nc $URL 1234 &
sleep 1
done
```

monkeyman vs. antman !!

After a while of manually adjusting the number according to the replies, the egg gets returned if one sends the number: 28232920712967180259

Solution: antman3351

```
import socket
import time

if __name__ == "__main__":
    notfound = True
    hostname = "hackyeaster.hacking-lab.com"
    port = 1234
    content = list ("55555555555555555555")

    while notfound:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((hostname, port))
        s.sendall("".join(content))
        s.shutdown(socket.SHUT_WR)
        reply = ""

        while 1:
            data = s.recv(1024)
            print data
            if data == "":
                break
            reply = reply + data

        x=str(reply[24:26])
        if x.isdigit():
            position = int(reply[24:26])
            moreOrLess = reply[26:27]
        else:
            position = int(reply[24:25])
            moreOrLess = reply[25:26]
```

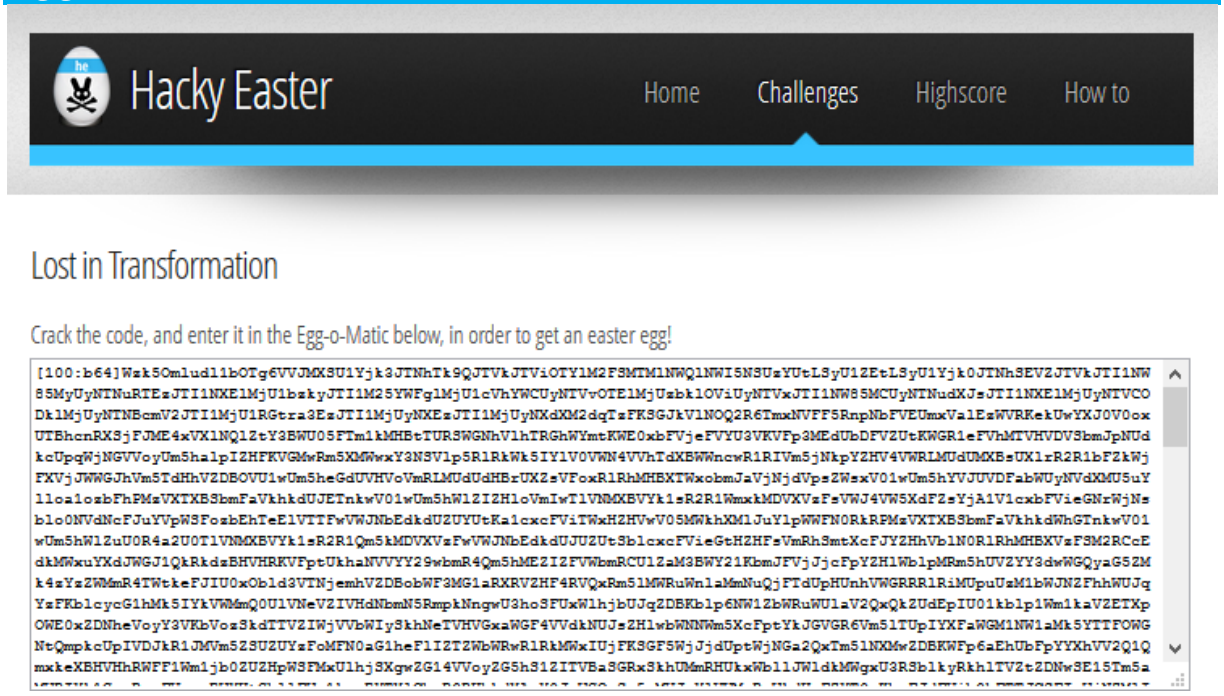
```
        print "".join(content) + " " + str(position) + " " + moreOrLess
        print "Connection closed."
        s.close()

        if position < 20:
            if moreOrLess == ">":
                content[position] = (str(int(content[position]) -1))
            elif moreOrLess =="<":
                content[position] = (str(int(content[position]) +1) )
            else:
                notfound = False
                break
        else:
            notfound = False

        time.sleep(1)

print "End"
```

Solution: one1

Feeding the service with a number with correct length (20) returns a hint on the number: position and whether it's lower or higher than the correct code. Wrote a small python script to do the work, which reads the hint and loops until the correct number is found.

```
import socket,re

def send_guess(guess):
  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  s.connect((TCP_IP, TCP_PORT))
  s.recv(BUFFER_SIZE)
  s.send(str(guess) + '\n')
  data = s.recv(BUFFER_SIZE)
  s.close() return data

TCP_IP = 'hackyeaster.hacking-lab.com'
TCP_PORT = 1234
BUFFER_SIZE = 4096

MESSAGE = 10000000000000000000
while True:
  data = send_guess(MESSAGE)
  matcher = re.match('^(\d*)([<>])', data)
  if matcher is None:
    break;
  pos = matcher.group(1)
  val = matcher.group(2)
  if val == "<":
    MESSAGE += 10 ** (19 - int(pos))
  elif val == ">":
   MESSAGE -= 10 ** (19 - int(pos))
  print MESSAGE

print "Solution:", MESSAGE
```

Egg 12 - Lost in Transformation



Solution: Atarii

This one I noticed the start of the blob of text indicates a mutator and index. After manually decoding a few layers I then used Python to crack this automatically:

```
import base64, urllib, re

a = '[100:b64]Wzk5Omlu ... hLS0s='

def parseStr(data, modifier):
    try:
        return {
          'b64': lambda data: base64.b64decode(data + '=' * (-len(data) % 4)),
          'b32': lambda data: base64.b32decode(data),
          'url': lambda data: urllib.unquote(data).decode('utf-8'),
          'r13': lambda data: data.encode('rot13'),
          'rev': lambda data: data[::-1],
          'inv': lambda data: data.swapcase(),
          'hex': lambda data: re.findall('[0-9A-Fa-f]*',data)[0].decode('hex')
        }[modifier.lower()](data)
    except KeyError:
        return data

while a.count('['):
    data = a[len(a[:a.find(']')])+1:]                    # Wzk5Omlud...
    modifier = a[:a.find(']')].split(':')[1]             # b64
    print a[a.find('[')+1:a.find(']')]                   # print 100:b64
    a = parseStr(data, modifier)
print 'Password is',a
```

Solution: Lumati

```
while (true)
  {
    if (input.StartsWith("[") == false)
    {
      break;
    }
    string action = "";
    System.Text.RegularExpressions.Regex r = new
System.Text.RegularExpressions.Regex(@"\[\d{1,3}:(.*?)\]");
    foreach (System.Text.RedgularExpressions.Match m in r.Matches(input))
    {
      input = input.Substring(m.Value.Length);
      action = m.Value.Substring(1, m.Value.Length - 2).Split(':')[1];
      break;
    }
    if (action != "")
    {
      Console.WriteLine("------------------------------------------------------------");
      Console.WriteLine(action);
      Console.WriteLine(input);
      Console.WriteLine();
      switch (action.ToLower())
      {
        case "b64": { input = b64(input); } break;
        case "b32": {
          input =
System.Text.Encoding.Default.GetString(Utilities.Base32.FromBase32String(input)); }
          break;
        case "inv": { input = inv(input); } break;
        case "url": { input = System.Web.HttpUtility.UrlDecode(input); } break;
        case "nop": { } break;
        case "r13": { input = r13(input); } break;
        case "xxx":
          {
              input = System.Text.RegularExpressions.Regex.Match(input,
"xxx(.*)xxx").Value;
              input = input.Substring(3);
              input = input.Substring(0, input.Length - 3);
          } break;
        case "rev":
          {
              char[] arr = input.ToCharArray();
              Array.Reverse(arr);
              input = new string(arr);
          } break;
        default:
          {
              Console.WriteLine(action + " not known");
              Console.ReadKey();
          } break;
      }
      Console.WriteLine(input);
    }
  }
  Console.WriteLine(input);
  Console.WriteLine("Done!");
  Console.ReadLine();
}
```

Solution: chrisderham

I wrote a java program to decode each line. Each line starts with [xx:yyy] where xx is the line number - 100 - 1, and yyy is the operation. Operations are:

```
b64 binary 64 decode
b32 binary 32 decode
inv invert case
nop no operation
xxx remove xxx prefix and suffix
url URL decode
r13 rot 13 decode
rev reverse string
hex hex decode
```

Attached file LostInTransformation.java. Answer was 33ster_b0nny.

```
import java.net.URLDecoder;
import org.apache.commons.codec.binary.Base32;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.codec.binary.Hex;

public class LostInTransformation {
  public static void main(String[] args) throws Exception {
    Base32 base32 = new Base32();
    Hex hex = new Hex();
    String source = "[100:b64]Wzk5Omludl1bOTg6VVJMXSU1 […] FhLS0s=";
    int index = 0;
    int expectedLineNumber = 100;
    while ((index = source.indexOf(']')) > -1) {
      if (source.charAt(0) != '[') {
        System.out.println("gone wrong");
      }
      String prefix = source.substring(1, index);
      String rest = source.substring(index + 1);
      String[] parts = prefix.split(":");
      if (parts.length != 2 || Integer.valueOf(parts[0]) != expectedLineNumber) {
        System.out.println("gone wrong");
      }
      System.out.println("line:" + expectedLineNumber + " op:" +
          parts[1] + " line:" + source.substring(0, 15)
          + " rest " + rest);
      switch (parts[1]) {
      case "b64":
        source = new String(Base64.decodeBase64(rest));
        break;
      case "b32":
        source = new String(base32.decode(rest));
        break;
      case "inv":
        source = invert(rest);
        break;
      case "nop":
        source = rest;
        break;
      case "xxx":
        source = rest.substring(3);
        int length = source.length();
        if (!source.substring(length - 3, length).equals("xxx")) {
          System.out.println("gone wrong");
        }
        source = source.substring(0, length - 3);
        break;
      case "url":
        source = URLDecoder.decode(rest, "UTF-8");
        break;
      case "r13":
        source = rot13(rest);
        break;
      case "rev":
```

```
        source = new StringBuffer(rest).reverse().toString();
        break;
      case "hex":
        source = new String(hex.decode(rest.getBytes()));
        break;
      default:
        System.out.println("Unknown op: " + parts[1]);
        break;
      }
      expectedLineNumber--;
    }
    System.out.println("Answer is:" + source);
  }

  private static String invert(String text) {
    char[] chars = text.toCharArray();
    for (int i = 0; i < chars.length; i++) {
      char c = chars[i];
      if (Character.isUpperCase(c)) {
        chars[i] = Character.toLowerCase(c);
      } else if (Character.isLowerCase(c)) {
        chars[i] = Character.toUpperCase(c);
      }
    }
    return new String(chars);
  }

  private static String rot13(String text) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < text.length(); i++) {
      char c = text.charAt(i);
      if (c >= 'a' && c <= 'm')
        c += 13;
      else if (c >= 'A' && c <= 'M')
        c += 13;
      else if (c >= 'n' && c <= 'z')
        c -= 13;
      else if (c >= 'N' && c <= 'Z')
        c -= 13;
      sb.append(c);
    }
    return sb.toString();
  }
}
```

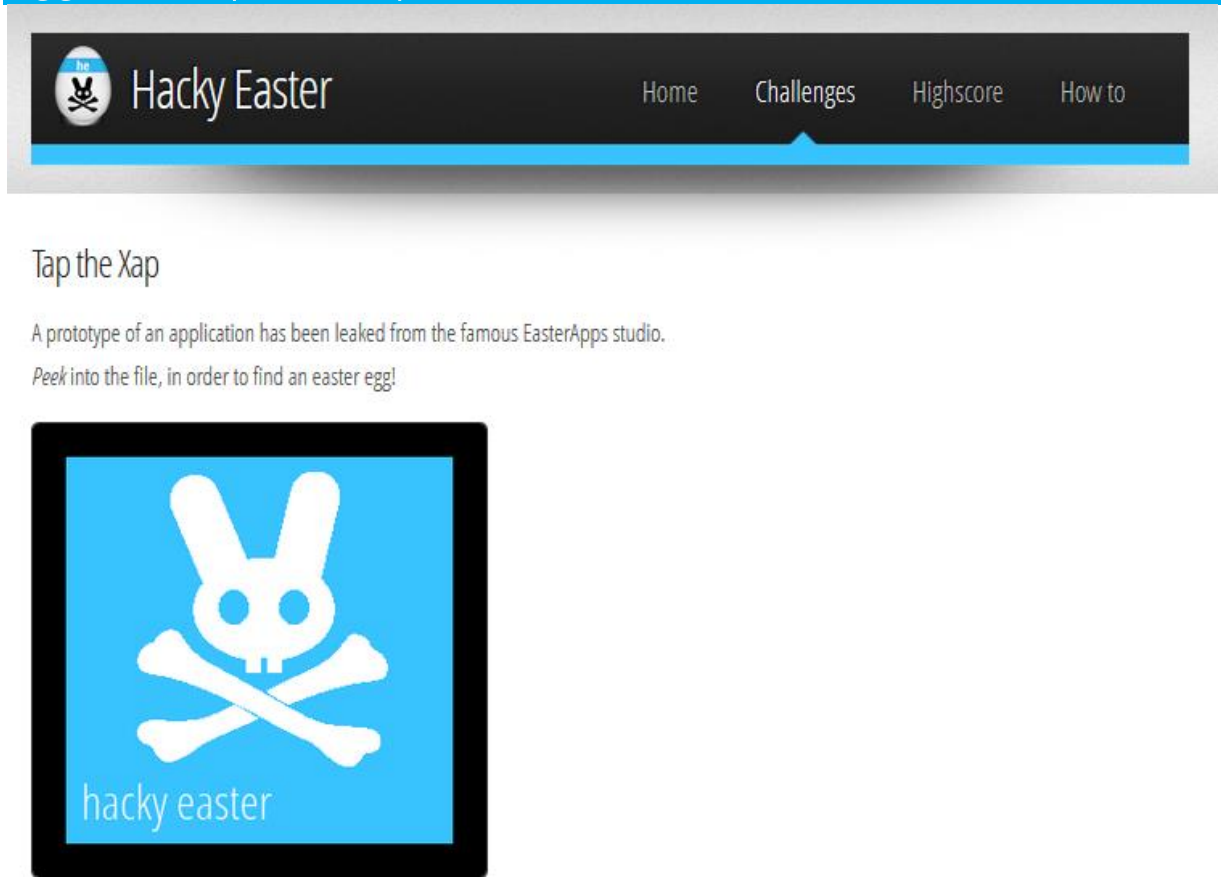
Solution: tunelko

This transformations can be identified by this functions and solve the challenge:

```
def b64(text):
    return d.decode('base64')

lower/uppercase inverted
def inv(text):
    result = ''
    for x in text:
        if x in uppercase:
            result += x.lower()
        elif x in lowercase:
            result += x.upper()
        else:
            result += x
                return result

def url(text):
    return unquote(text)

def nop(text):
    return text

def rot13(text):
    return text.decode('rot13')

def xxx(text):
    return text[3:-3]

def rev(text):
    return text[::-1]

def b32(text):
    return b32decode(text)

def hex(text):
    return text.decode('hex')
```

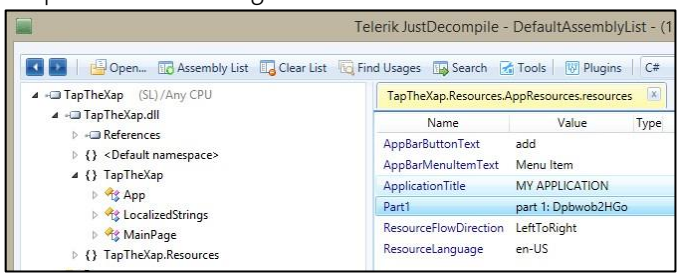
Nice solutions provided here, in multiple programming languages! Goal was to write a program or script, that's why I chose 100 steps. Many people solved manually anyway. I'll choose 1'000 steps next time ☺

Egg 13 - Tap the Xap



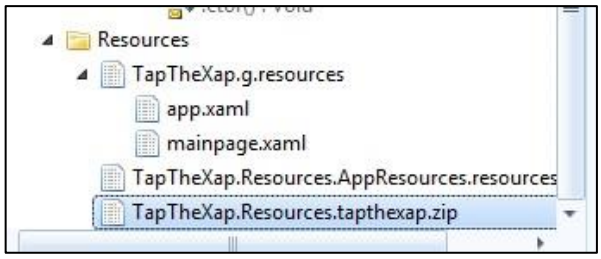
Solution: Lumati

After downloading the file, and peeking into it using a hex editor, some may recognize the first part of the file as a zip file. Others may need a tool like Trid.Net. After changing the extension to .zip I was able to open the fileand unpack it. I opened the DLL inside a CLR decompiler (I used Telerik) to peek into the code and found two parts of "something":

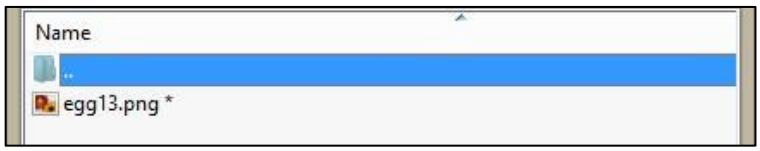



```
public App()
{
    App.Part2 = "u752reVoT1";
    base.add_UnhandledException(new EventH
    this.InitializeComponent();
    this.InitializePhoneApplication();
    this.InitializeLanguage();
}
```

After digging some more, I found a zip file embedded as resource carrying the name "TapTheXXap.Resoruces.tapthexap.zip". It looked like a zip file embedded as a dll resource.



After extracting it from the DLL I found the egg hidden inside. Only it required a password to fetch the egg.



I tried the previously found partials and after combining them I was able to extract the egg.

Solution: monkeyman

The .xap file can be renamed to .zip and extracted. The dll file contains a password protected zip archive with the egg.

```
$ binwalk TapTheXap.dll
DECIMAL    HEX        DESCRIPTION
--------------------------------------------------------------------------------
6612       0x19D4     Zip encrypted archive data, at least v2.0 to
                      extract, compressed size: 60719, uncompressed
                      size: 60707, name: "egg13.png"
67461      0x10785    End of Zip archive
71115      0x115CB    Copyright string: "Attributeibute"

$ dd if=TapTheXap.dll of=egg13.zip bs=1 skip=6612
```

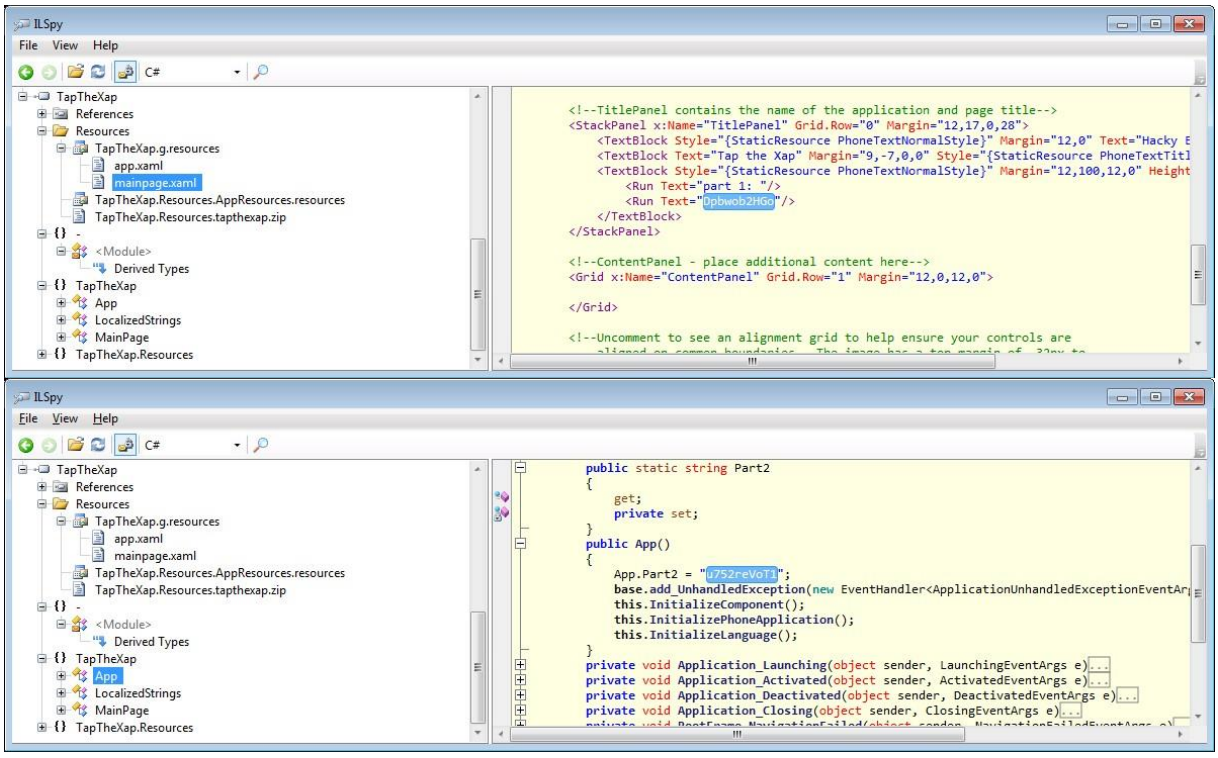
The dll file contains some clues about the password.

```
$ strings -a TapTheXap.dll
[...]
MY APPLICATION
part 1: Dpbwob2HGo
LeftToRight
[...]
$ strings -a -e l TapTheXap.dll
[...]
ResourceLanguageA
u752reVoT1
/TapTheXap;component/App.xaml
[...]
```

Nice solution, even without a tool like dotPeek.

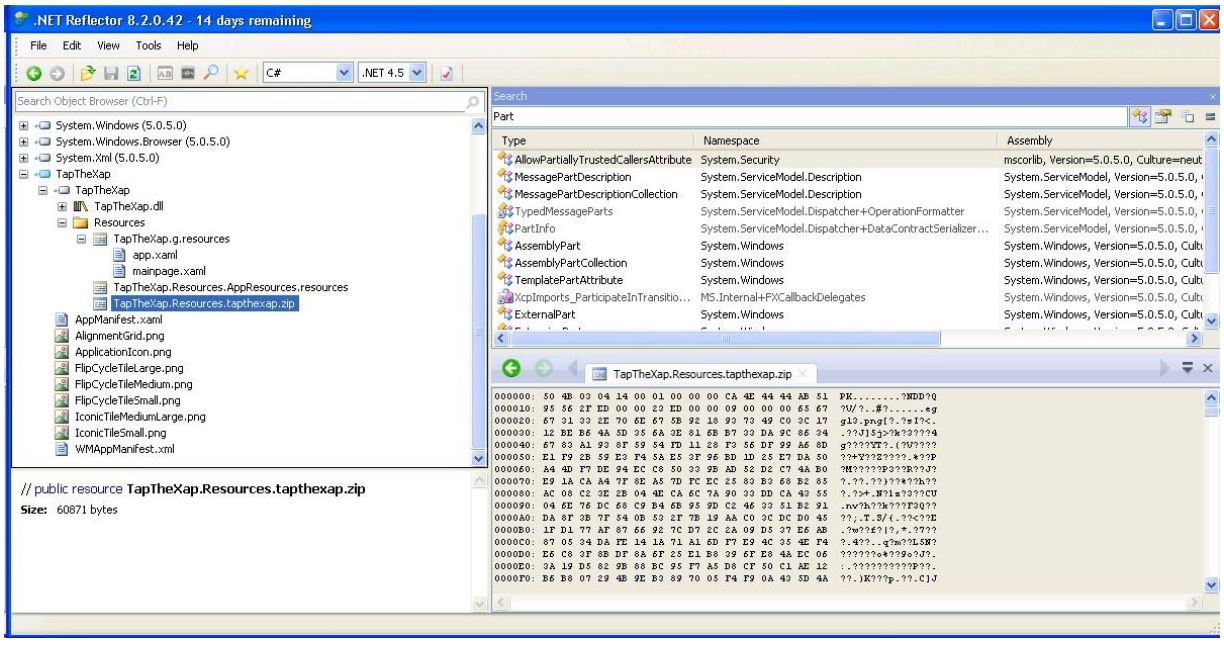
The string after `part 1:' looks like a password. Combining this with another string, which looks like a second part of the password, yields the correct one: Dpbwob2HGou752reVoT1

Solution: DanMcFly

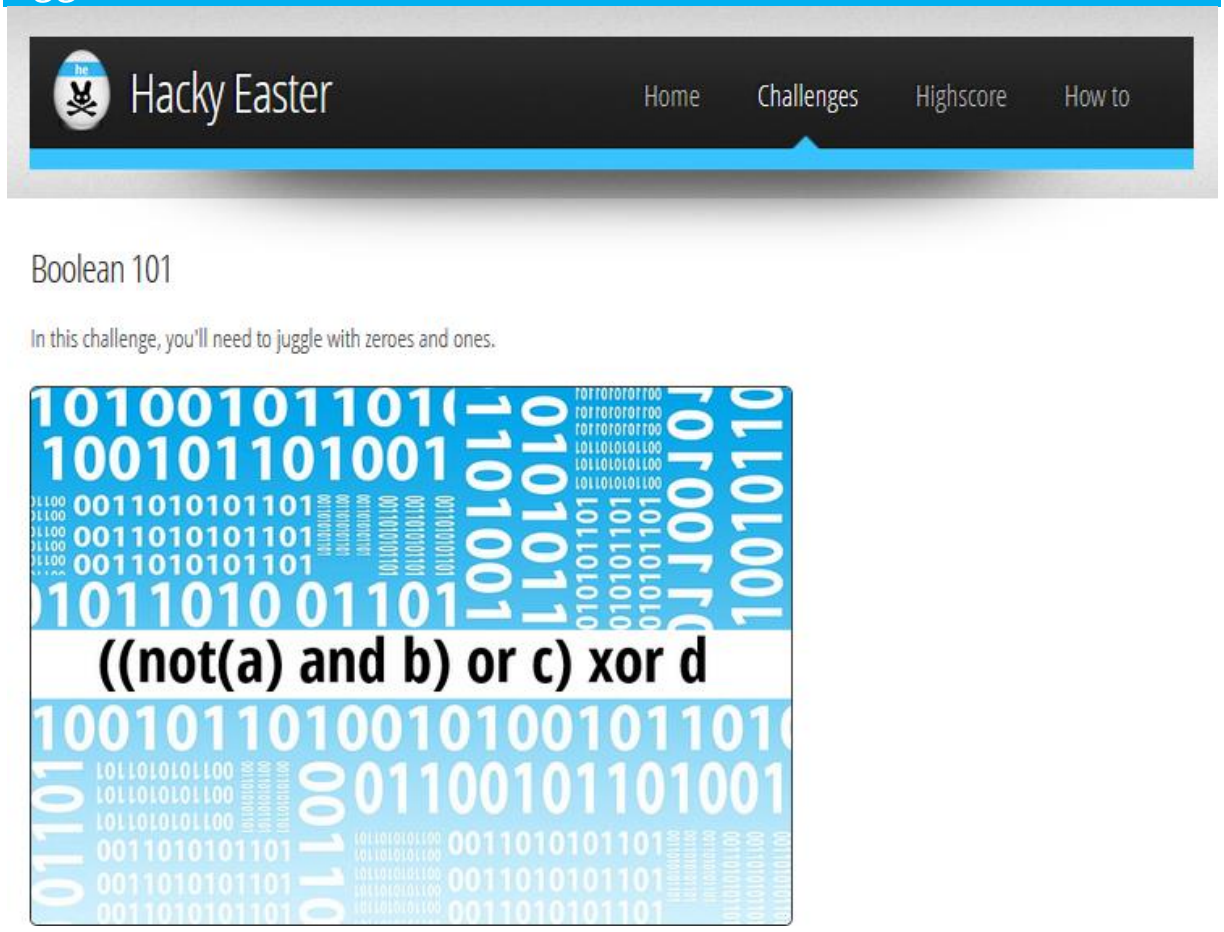


Part 1: ***Dpbwob2HGo***
Part 2: ***u752reVoT1***
Password: **Dpbwob2HGou752reVoT1**

In the Package, there is also a hidden zip. With the above Password we can easy extract the content.



Egg 14 - Boolean 101



Solution: tunelko

We can dump the result of (((not a) and b) or c) ^ d combining the a,b,c,d files

```
final = open('x', 'w')
file1 = [i.strip() for i in open('a.txt').readlines()]
file2 = [i.strip() for i in open('b.txt').readlines()]
file3 = [i.strip() for i in open('c.txt').readlines()]
file4 = [i.strip() for i in open('d.txt').readlines()]
final.write('P1\n')
final.write('25 25\n')

for x in xrange(25):
    for y in xrange(25):
        r1 = int(file1[x][y])
        r2 = int(file2[x][y])
        r3 = int(file3[x][y])
        r4 = int(file4[x][y])
        result = (((not r1) and r2) or r3) ^ r4
        final.write(str(result) + ' ')

$ file x
x: Netpbm PBM image text
```

Netpbm graphic format ? Pretty handy !

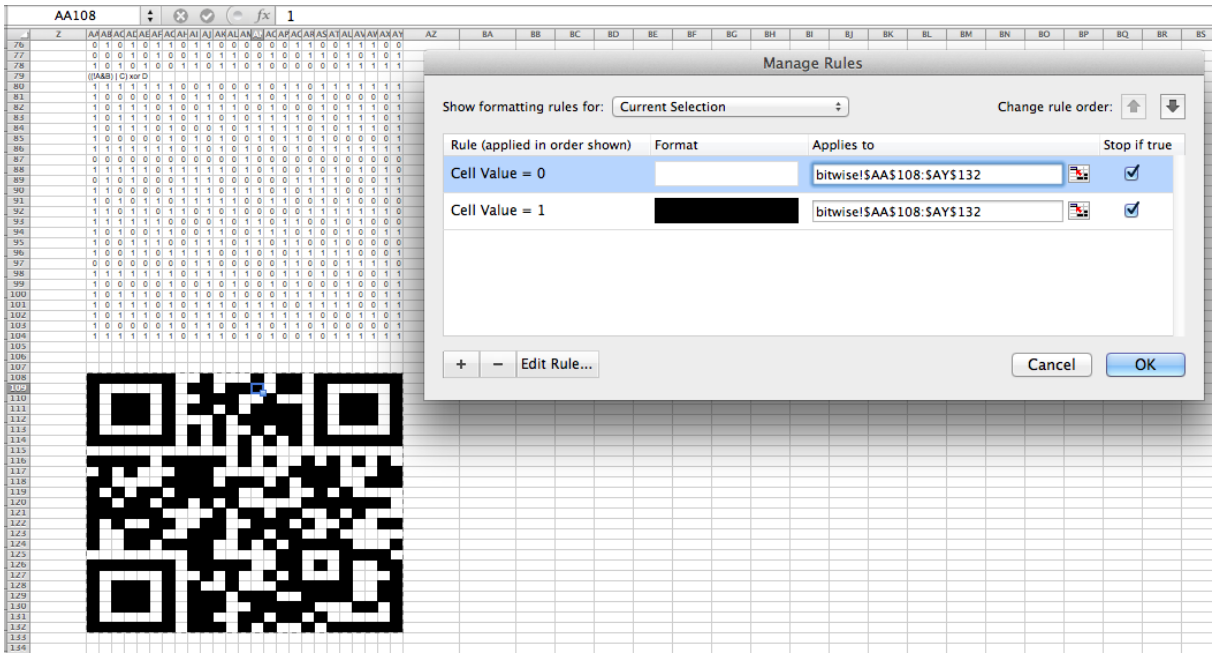
Solution: one1

Figured out that there are some matrix operations "hidden" in the challenge page. Use small python script to do the work – see appendix. In the end, used Word to display the QR code without line spacing so that it was scannable without fiddling with the console too much.

```
import numpy as np, sys

na = np.genfromtxt('a.txt',delimiter=1)
nb = np.genfromtxt('b.txt',delimiter=1)
nc = np.genfromtxt('c.txt',delimiter=1)
nd = np.genfromtxt('d.txt',delimiter=1)

tmp = np.logical_not(na)
tmp = np.logical_and(tmp,nb)
tmp = np.logical_or(tmp,nc)
tmp = np.logical_xor(tmp,nd)

string = ''
i = 0;
for x in np.nditer(tmp):
  i += 1
  sys.stdout.write(u"\u25A0" if x else ' ')
  sys.stdout.write('\n' if (i % 25 == 0) else ''
```

Solution: chris12

Perform the appropriate bitwise operations on the given numbers. Once complete, notice the 1's and 0's showed the pattern of a standard QR code with 3 large boxes in the the corners and darken all the spaces that came out as a 1 and whiten the spaces that came out as a 0. Space appropriately and scan the QR code for an egg.



Solution: HomeSen

Applying the given formula to the provided text files, results in a new text file with 25 rows and columns. Taking a closer look at the result and replacing 0's with a whitespace and 1's with ▉ revealed a stretched QR-code that translated to egg14.

Solution: HaRdLoCk

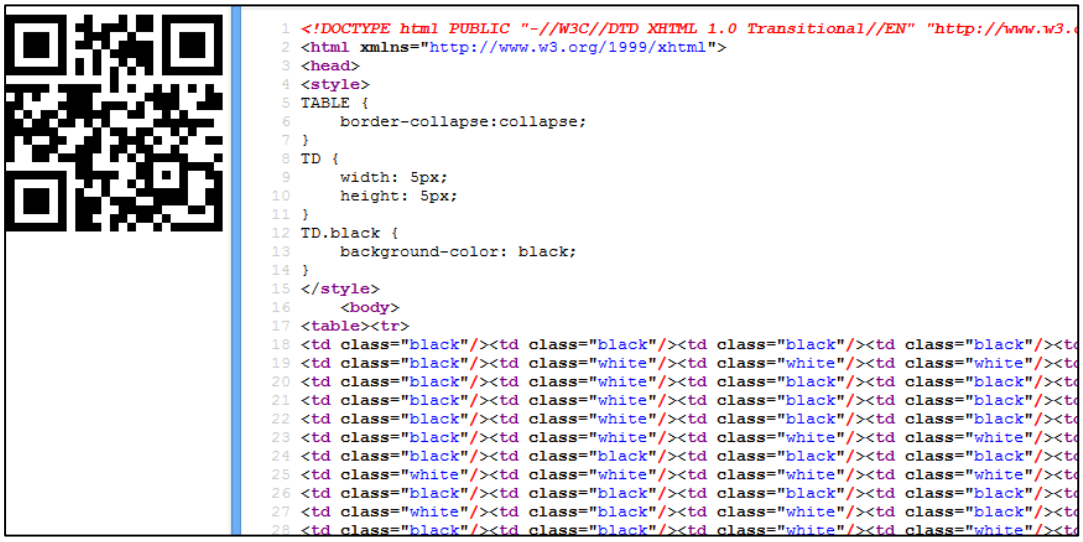
in the boolean challenge we first can calculate the operations that are shown on the image. Then we can use some code to transform it into a QR code. Here is my solution in vb.net.

```
Do While sra.Peek() >= 0
    Dim linea = Convert.ToInt32(sra.ReadLine(), 2)
    Dim lineb = Convert.ToInt32(srb.ReadLine(), 2)
    Dim linec = Convert.ToInt32(src.ReadLine(), 2)
    Dim lined = Convert.ToInt32(srd.ReadLine(), 2)
    Dim result = (((Not (linea) And lineb) Or linec) Xor lined)
    Dim binary = Convert.ToString(result, 2).PadLeft(25, "0")
    TextBox1.Text += binary & vbNewLine
    For i = 0 To 24
        Dim bit = Mid(binary, i + 1, 1)
        Dim j = i * 2
        If bit = "1" Then
            Bild.SetPixel(count, j, Color.Black)
            Bild.SetPixel(count, j + 1, Color.Black)
            Bild.SetPixel(count + 1, j, Color.Black)
            Bild.SetPixel(count + 1, j + 1, Color.Black)
        Else
            Bild.SetPixel(count, j, Color.White)
            Bild.SetPixel(count, j + 1, Color.White)
            Bild.SetPixel(count + 1, j, Color.White)
            Bild.SetPixel(count + 1, j + 1, Color.White)
        End If
    Next
    count += 2
Loop
```

Solution: chrisderham

I wrote the attached java program to perform the given logic operation given in the image on the data. When I first wrote the program, I managed to notice that the output is the binary of a QR code. So I updated the code to output html to show the QR code, which I was then able to scan.

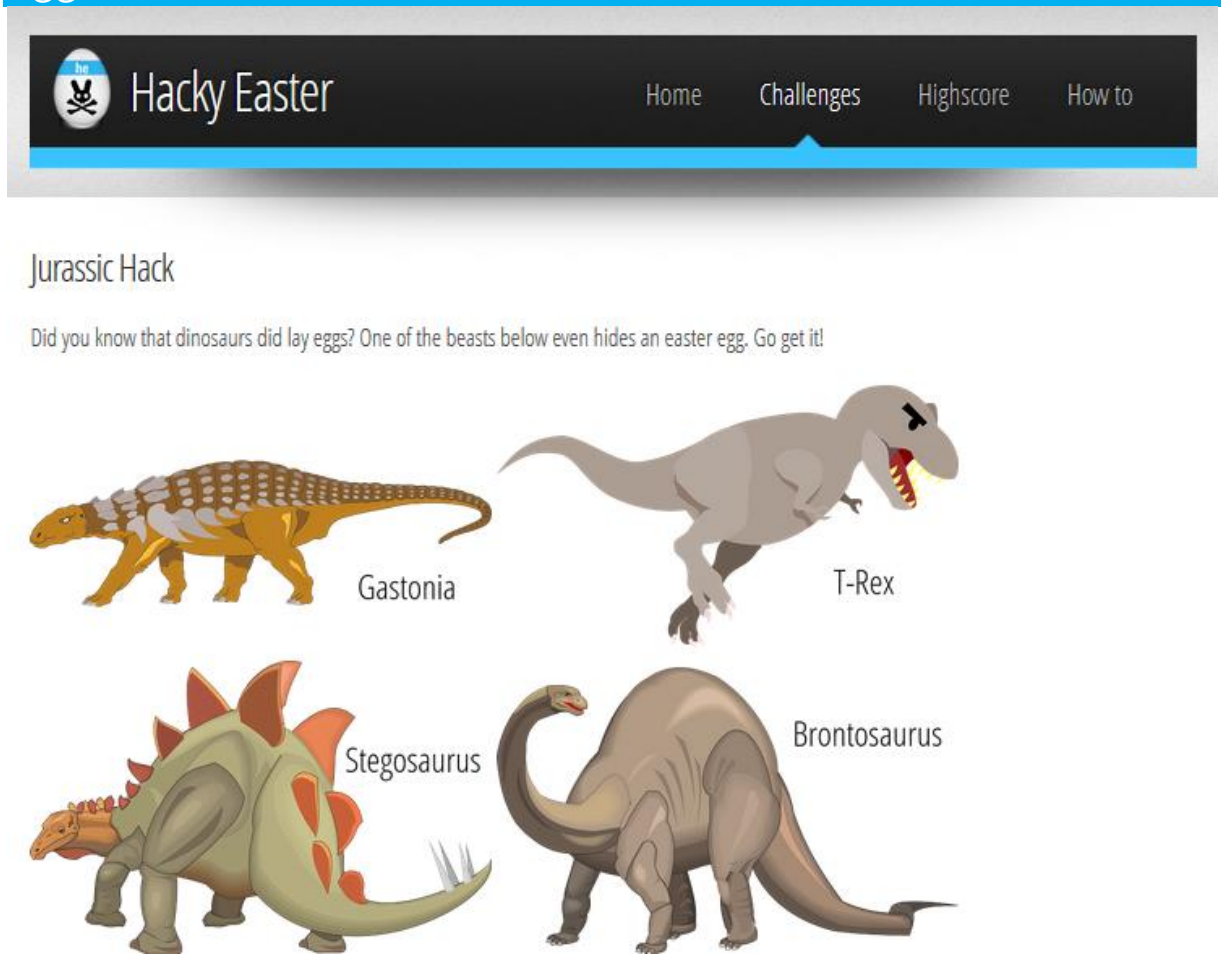
```
public class Boolean101 {
  public static void main(String[] args) {
    String a = "01110110010001001111111111010001110100110110111010000011000000001111001
    String b = "01110010011100100101000101001110011110110101100010110110000011010111101
    String c = "10010100111010000100010000100101000010110010000000000001100001000000000
    String d = "01101010101111111111101110101100001101011111000011011111000000101001101

    System.out.println("<html><body><table><tr>");

    int charCount = 0;
    for (int i = 0; i < a.length(); i++) {
      boolean a_ = a.charAt(i) == '1';
      boolean b_ = b.charAt(i) == '1';
      boolean c_ = c.charAt(i) == '1';
      boolean d_ = d.charAt(i) == '1';
      boolean result = ((!a_ && b_) | c_) ^ d_;
      System.out.print(result ? "<td class=\"black\"/>" : "<td class=\"white\"/>");
      if (++charCount == 25) {
        charCount = 0;
        System.out.println("</tr><tr>");
      }
    }
    System.out.println("</tr></table></body></html>");
  }
}
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<style>
TABLE {
    border-collapse:collapse;
}
TD {
    width: 5px;
    height: 5px;
}
TD.black {
    background-color: black;
}
</style>
    <body>
<table><tr>
<td class="black"/><td class="black"/><td class="black"/><td class="black"/><t
<td class="black"/><td class="white"/><td class="white"/><td class="white"/><t
<td class="black"/><td class="white"/><td class="black"/><td class="black"/><t
<td class="black"/><td class="white"/><td class="black"/><td class="black"/><t
<td class="black"/><td class="white"/><td class="black"/><td class="black"/><t
<td class="black"/><td class="white"/><td class="white"/><td class="white"/><t
<td class="black"/><td class="black"/><td class="black"/><td class="black"/><t
<td class="white"/><td class="white"/><td class="white"/><td class="white"/><t
<td class="black"/><td class="black"/><td class="black"/><td class="black"/><t
<td class="white"/><td class="black"/><td class="white"/><td class="black"/><t
<td class="black"/><td class="black"/><td class="white"/><td class="white"/><t
```

Nice thought: why not use Excel, or even creating an HTML table, for visualizing the QR code...

Egg 15 - Jurassic Hack



Solution: zato

With the ability to print PNG file headers (hexdump –C|head) it was not difficult to find the right image, stegosaurus.png.

```
srzrts@chrvl385:~/easter/ch7> cat stegosaurus.png | hexdump -C |head
00000000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52  |.PNG........IHDR|
00000010  00 00 02 80 00 00 01 97  08 06 00 00 00 23 f6 e1  |.............#..|
00000020  9d 00 00 00 01 73 52 47  42 00 ae ce 1c e9 00 00  |.....sRGB.......|
00000030  00 04 67 41 4d 41 00 00  b1 8f 0b fc 61 05 00 00  |..gAMA......a...|
00000040  00 09 70 48 59 73 00 00  0e c3 00 00 0e c3 01 c7  |..pHYs..........|
00000050  6f a8 64 00 00 00 1a 74  50 75 66 66 20 73 74 65  |o.d....tPuff ste|
00000060  67 6f 73 61 75 72 75 73  20 6e 6f 20 42 20 6e 6f  |gosaurus no B no|
00000070  20 43 20 20 20 47 f3 42  37 00 01 00 00 49 44 41  | C   G.B7....IDA|
00000080  54 78 5e ec 9d 79 78 14  55 d6 c6 df 7b ab 3a 3b  |Tx^..yx.U...{.:;|
```

It required a hint to google "puff steganography" to find OpenPuff. In there it was possible to load the file, to uncheck b anc c (whatever this is...) and to have "steganosaurus" as password. In this configuration, the disclosed text is pointing to the egg:
http://hackyeaster.hackinglab.com/hackyeaster/img/eggs/egg15kdm7deolxiYKJuR93O7B.png

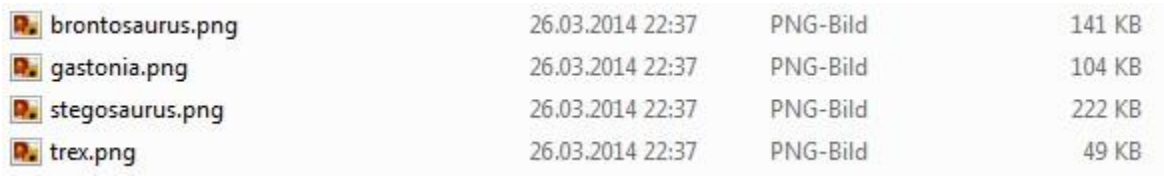
Solution: roddick

Check gastonia.png and get the hint in the file “Check Stego”. Then check stegosaurus.png and get the hint in the file “Puff stegosaurus no B no C”. So use the tool OpenPuff to unhide the picture stegosaurus.png. Uncheck the B and C, fill in stegosaurus in A and click unhide. Then you will get the url of the egg “http://hackyeaster.hacking-lab.com/hackyeaster/img/eggs/egg15kdm7deolxiYKJuR93O7B.png”.

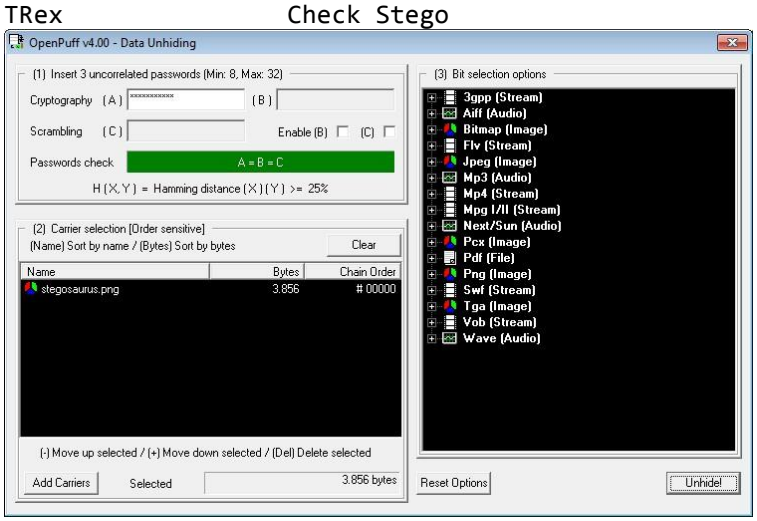
Solution: Atarii

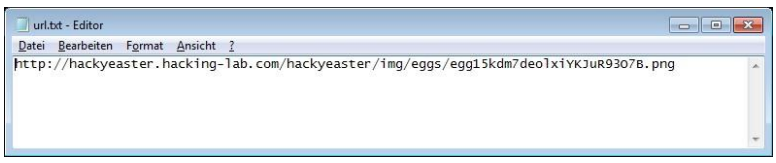
This was a stenography challenge, with the hint being a dinosaur, so I guessed first time the egg would be in the stegosaurus. Inside the PNG was "Puff stegosaurus no B no C" text
After Googling for Puff and stenography, I found OpenPuff tool. Using "Puff stegosaurus" password in box A (and no B or C like the hint said) unhid the egg.

Solution: DanMcFly

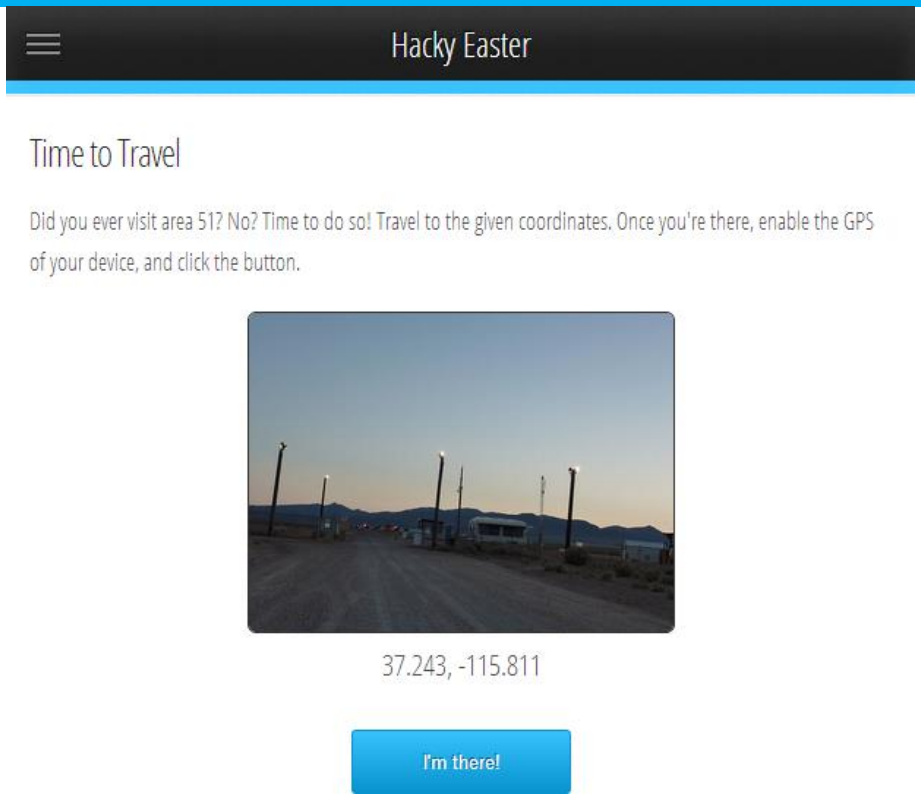



```
Brontosaurus        Check Stego
Gastonia            Check Stego
Stegosaurus         stegosaurus no B no C
TRex                Check Stego
```





Egg 16 - Time To Travel



Solution: HaRdLoCk (iOS)

In time to travel i adjusted the html page like this:

```
var locOk = function(pos) {
  var lng = -115.811;
  var lat = 37.243;
  if (Math.floor(lat*1000) == 37243 && Math.floor(-lng*1000) == 115811) {
```

On the iphone this will pass and show egg16.

Solution: M. (android)

Time To Travel challenge from the App. Upon pressing the button, the challenge page does a callback into android code to generate the passphrase used to decrypt the crypt-js encrypted egg.
From decompiling the android class tree, the passphrase is found to be Base64(SHA1(latitude*1000+'bunny'+longitude*1000)), i.e. 'CWrg8zQpZwVEObue5fdwFHh9eEs='

Solution: roddick (iOS)

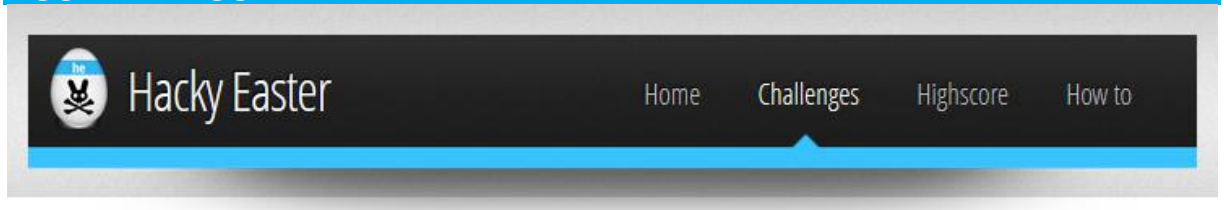
First jailbreak your iPhone and install AklocationX. Then set the coordinates as the challenge shown.

Solution: Seppel

The App checks your location, if you are at the mentioned coordinates, the egg will be displayed

- Download an App like 'Fake gps – fake location' – 'allow mock location' must be switched on in development options
- Set Fake Location to: 37.243,-115.811

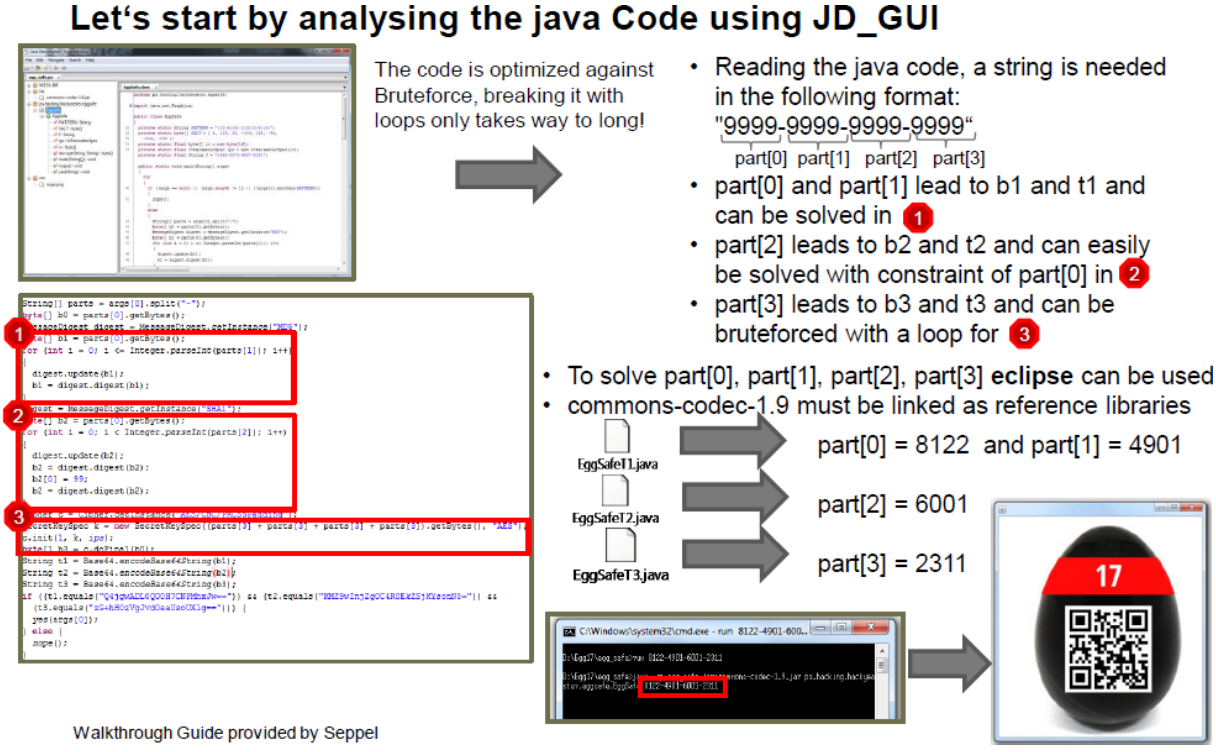
Egg 17 - Egg Safe



Egg Safe

The safe below holds a red egg. You'll need to crack it's code - a string of format "9999-9999-9999-9999" - in order to get it.

Solution: Seppel



Solution: M.

Egg Safe challenge from the web. A Java Application is provided, along with two encrypted PNG file (one of which is the egg, the other indicates a failure). The application expects an input of the form '0000-0000-0000-0000', where '0' is an arbitrary decimal digit. The verification code consists of three steps (foo() ## N means foo() applied N times nested, and part0..3 refer to the appropriate part of the input, separated by the hyphens):

- Base64(MD5(part0)) ## (part1+1)) == 'Q4jgwADL0QO0H7CNPMhxJw=='
- Base64((SHA1([99,SHA1(part0)[1:]])) ## part2) == 'KMZ9wInjZg0C4R0EkZSjKYsonN8='
- Base64(AES_encrypt(part0,part3.part3.part3.part3)) == 'zG+hH0zVgJvd0aaUsoUXlg=='

The most straightforward, but also a very inefficient way would be to crack the parts in this order, starting with the MD5##part1 part. This requires 10000*10000 guesses to get part0 and part1, and each guess requires multiple MD5 evaluations unless further optimized.

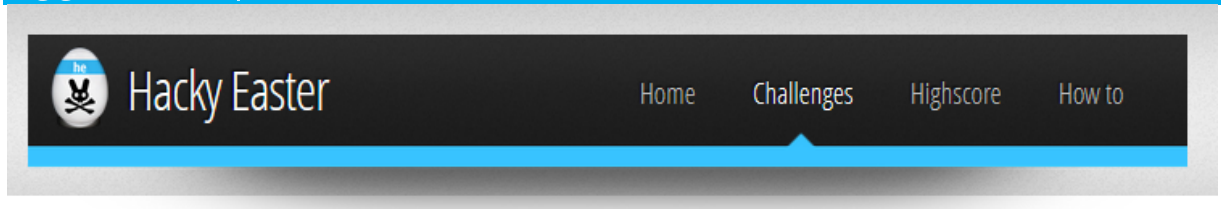
I like this optimization ! (Even though, it doesn't matter too much, since the cracking only runs once.)

A much more efficient approach is to start with the last part by AES-decrypting the block with each of the 10000 possible keys. The plaintext is known to be the decimal string representation of a 4-digit number with PKCS5 padding. Once the right key is found, both part0 and part3 are known, so attacking part1 and part 2 only requires 10000 guesses at most. Hence the worst-case number of guesses is 10000+10000+10000 << 10000*10000. The following Java code implements this attack, finding '8122-4901-6001-2311'.

```
import java.io.ByteArrayInputStream;
import java.security.MessageDigest;
import javax.crypto.*;
import javax.crypto.spec.*;
import sun.misc.BASE64Encoder;
import sun.misc.BASE64Decoder;

public class Crack {
	public static void main(String[] a) throws Exception {
		BASE64Encoder Base64e = new BASE64Encoder();
		BASE64Decoder Base64d = new BASE64Decoder();
		String parts[] = {"____","____","____","____"};

		for(int crk=0;crk<=9999;crk++) {
			parts[3] = String.format("%04d",crk);
			try {
				Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
				SecretKeySpec k = new SecretKeySpec((new
StringBuilder(String.valueOf(parts[3]))).append(parts[3]).append(parts[3]).append(
parts[3]).toString().getBytes(), "AES");
				c.init(Cipher.DECRYPT_MOODE, k, new IvParameterSpec(new
byte[16]));
				byte b3[] =
c.doFinal(Base64d.decodeBuffer("zG+hH0zVgJvd0aaUsoUXlg=="));
				int i = Integer.parseInt(new String(b3));
```

```
                        if(i>=0 && i <= 9999) {
                              parts[0] = String.format("%04d",i);
                              break;
                        }
                  } catch(Exception e){ /* wrong key */ }
            }
            System.out.printf("Found: %s-%s-%s-
%s\n",parts[0],parts[1],parts[2],parts[3]);
            MessageDigest digest = MessageDigest.getInstance("MD5");
            byte[] b1 = parts[0].getBytes();
            for(int crk=0;crk<=9999;crk++) {
                  digest.update(b1);
                  b1 = digest.digest(b1);
                  String t1 = Base64e.encode(b1);
                  if(t1.equals("Q4jgwADL0QO0H7CNPMhxJw==")) {
                        parts[1] = String.format("%04d",crk);
                      break;
                  }
            }
            System.out.printf("Found: %s-%s-%s-
%s\n",parts[0],parts[1],parts[2],parts[3]);

            digest = MessageDigest.getInstance("SHA1");
            b1 = parts[0].getBytes();
            for(int crk=0;crk<=9999;crk++) {
                  String t1 = Base64e.encode(b1);
                  if(t1.equals("KMZ9wInjZg0C4R0EkZSjKYsonN8=")) {
                        parts[2] = String.format("%04d",crk);
                      break;
                  }
                digest.update(b1);
                b1 = digest.digest(b1);
                b1[0] = 99;
                b1 = digest.digest(b1);
            }
            System.out.printf("Found: %s-%s-%s-
%s\n",parts[0],parts[1],parts[2],parts[3]);
      }
}
```

Egg 18 - Paper and Pen



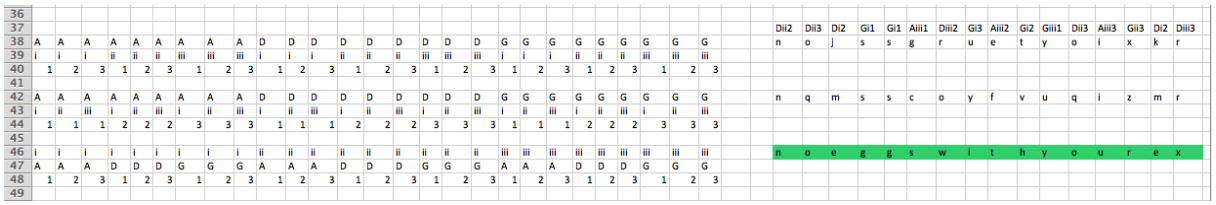
Paper and Pen

Crack the following ciphertext, in order to get the password for the egg-o-matic. Hint: lowercase letters only.

```
Dii2 Dii3 Di2 Gi1 Gi1 Aiii1 Diii2 Gi3 Aiii2 Gi2 Giii1 Dii3 Aiii3 Gii3 Di2 Diii3
```

Solution: chris12

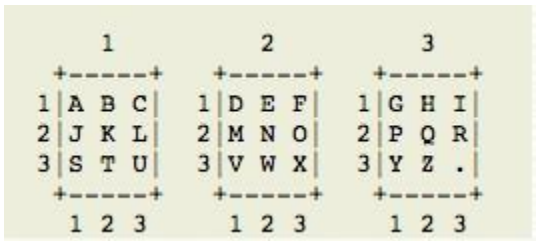
3^3 is 27 and could be a close mapping to the 26 letters. Try various orders of the 3 bits of information mapped into the alphabet. With i-iii as most significant, ADG as next, and 1-3 as last, it decodes to noeggswithyourex. Put that in the Egg-O-Matic for an egg.



Solution: tunelko

Task is trifid cipher, see transformations and matrix result.

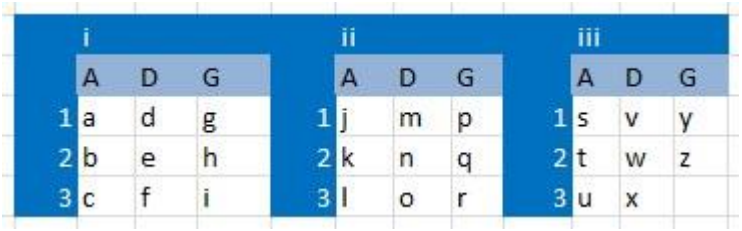
```
Dii2 Dii3 Di2 Gi1 Gi1 Aiii1 Diii2 Gi3 Aiii2 Gi2 Giii1 Dii3 Aiii3 Gii3 Di2 Diii3

D22 D23 D12 G11 G11 A31 D32 G13 A32 G12 G31 D23 A33 G23 D12 D33
212 223 212 311 311 131 232 313 132 312 331 223 133 323 212 233
```



password: noeggswithyourex

Solution: HaRdLoCk

Paper and pen is a handmade cipher which functions like trifid cipher. when we make up 3 tables with lowercase letters like this:



i				ii				iii			
	A	D	G		A	D	G		A	D	G
1	a	d	g	1	j	m	p	1	s	v	y
2	b	e	h	2	k	n	q	2	t	w	z
3	c	f	i	3	l	o	r	3	u	x	

We can simply read the solution: noeggswithyourex.

Solution: M.

Paper and Pen challenge from the web. The challenge description and title hint a hand cipher, and each letter seems to consist of 3 parts: A,D,G - i,ii,iii - 1,2,3. This indicates a set of 3x3x3 matrices, i.e. a variant of a trifid cipher. The first guess to put up the matrices is:

```
A B C          J K L          S T U
D E F          M N O          V W X
G H I          P Q R          Y Z
```

In this setup, the A/D/G part most likely names the row (starting with A,D,G respectively), while the i/ii/iii part was assumed to name the matrix and the 1/2/3 part the column. This leads to the plaintext 'noeggswithyourex', which is the valid passphrase to decrypt the egg.

Eggs