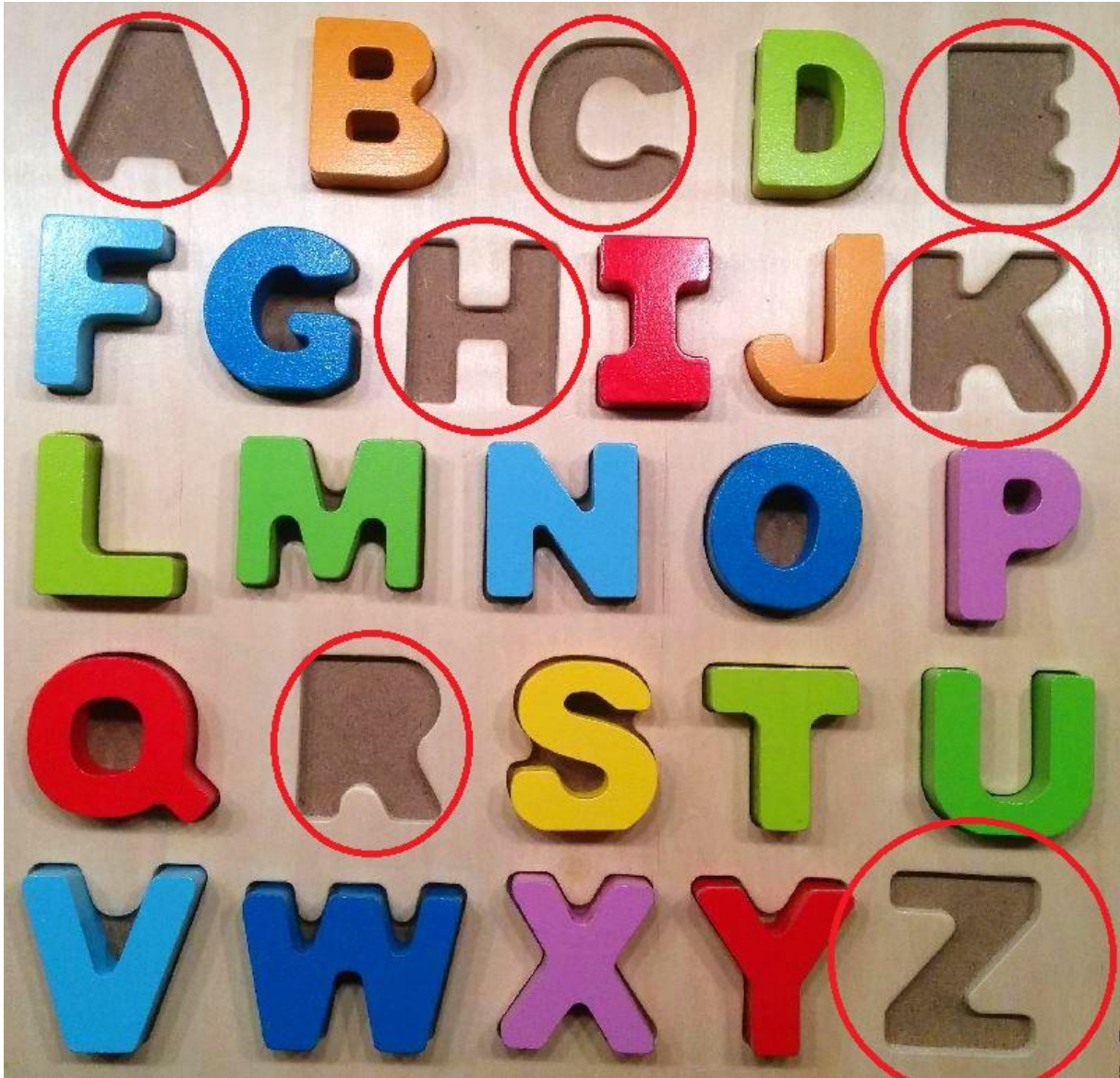


Challenge 1 - Puzzleword

In you look closer to the picture, you notice the missing letters "ACEHKRZ". This is an anagram of "HACKERZ". Solution Number 1 ☺



Challenge 2 - It's in the Media

The QR Code is covered with the word "NO" and the hint is "It's in the Media". Look in to the source code and search "media". You will find the following code:

Line 19:

```
<style>
    .page { background-color: white !important;} .h {
display:none;} .i3 { height: 4%; width: 4%; background: #fff;}
    .o2 { height: 4%; width: 4%; background: #000;}.l1 { height:
4%; width: 4%; background: #000;}
    .x5 { height: 4%; width: 4%; background: #fff;}@media print
{body {-webkit-print-color-adjust: exact;}
    .h { display:block;}.l1 { height: 4%; width: 4%; background:
#000;}.x5 { height: 4%; width: 4%; background: #000;}}
```

</style>

If you erase the CSS Element “@media”-element, the “NO”-Label will disappear and you got the solution:

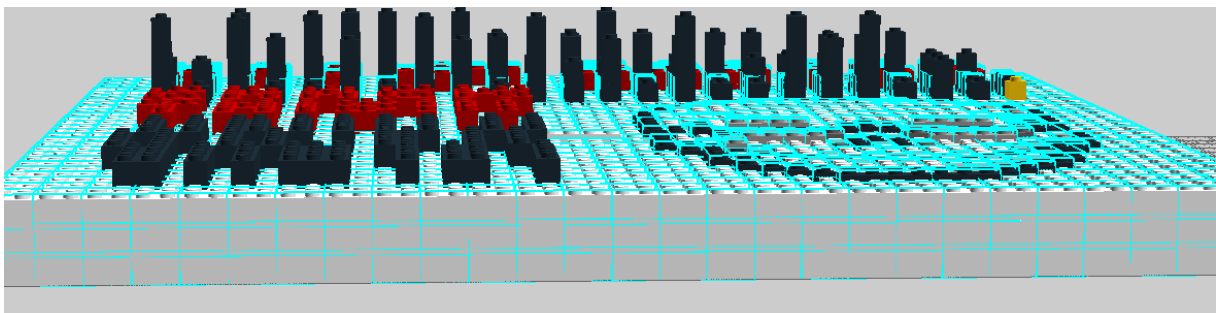


Challenge 3 - Lego Stego

The LXF-File is a normal ZIP File. If you unpack the zip file you will get a PNG File and a LXFML-File. The last one is a normal XML file. On line 4 you will see, which program used this format:

```
<Application name="LEGO Digital Designer" versionMajor="4" versionMinor="2"/>
```

Go and download the LEGO Digital Designer and load the LXF file in it. If you look from the site you will notice, that there are 3 layers.



Erase the third layer and all the stones above and you will see the QR-Code



Challenge 4 - Twisted Num63rs

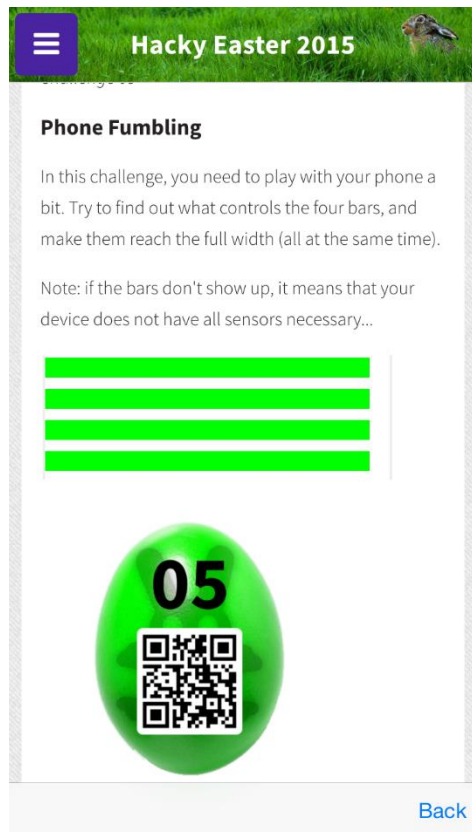
In this Challenge you should sort the values in ascending order. Here are the converted values, in ascending order:

Original	Converted
$\text{sqrt}(1296)$	36
$\text{Pi} \wedge \text{Pi}$	36,46215
ZmlmdHk=	50
Middle C [Hz]	261.6
10101111000	1.400
303240 (oktal)	100.000
$2 \wedge 20$	1.048.576
13 MiB [bytes]	13.631.488
Speed of Light [m/s]	299.792.458
127.0.0.1 as Integer	2.130.706.433
java.lang.Integer.MAX_VALUE	2.147.483.647
8 YiB[bytes]	9.671.406.556.917.033.397.649.408

Challenge 5 - Phone Fumbling

After a while it was easy to realizing that the bars are influenced by the following things:

- Bar1: time (only one minute or so)
- Bar2: gyroscope (play with your smartphone, that you get the right direction)
- Bar3: gyroscope (play with your smartphone, that you get the right direction)
- Bar4: battery / charging value



The screenshot shows a mobile application interface for 'Hacky Easter 2015'. At the top, there is a purple header with a white hamburger menu icon on the left and the text 'Hacky Easter 2015' in white. Below the header is a green banner image of a rabbit in a field. The main content area has a white background with the title 'Phone Fumbling' in bold black text. Below the title, there is a paragraph of text: 'In this challenge, you need to play with your phone a bit. Try to find out what controls the four bars, and make them reach the full width (all at the same time).' followed by a note: 'Note: if the bars don't show up, it means that your device does not have all sensors necessary...'. Below the text are four horizontal green bars of varying lengths, representing progress indicators. At the bottom of the challenge area is a green Easter egg with the number '05' in black and a QR code. At the very bottom of the screen, there is a light gray bar with the word 'Back' in blue text.

Challenge 6 - Hack to the Future

The following message is written in Morse-code

"dah-dah-dit dit dah-dah-dah di-dah-dit dah-dah-dit dit dah-dah dah-di-dah-dit di-di-dah-dit di-dah-di-dit dah-di-dah-dah"

Decoded it is the solution code: "georgemcfly"



Wait we are too early. Set the local time 3 months later and we get the solution.



Challenge 7 - Vista de la Calle

On the iPhone the Challenge was broken and with the fix you are already on the right map. Look in the sky and you will see the QR-Code. The only thing you need is to make a border with the same color. Something like this:



Challenge 8

To get to the spreadsheet you need a link from an other spreadsheet, so that you can replace the ID:

The general link looks like this: [https://docs.google.com/spreadsheets/d/\[ID\]/edit?usp=sharing](https://docs.google.com/spreadsheets/d/[ID]/edit?usp=sharing)

So the original Link is the following:

https://docs.google.com/spreadsheets/d/1QPkfrnSVRAhQKL7AZx_HVXWrRXDvwCnVX2ih0jYp1CA/edit?usp=sharing

Open the spreadsheet in excel and sort the top values in line 1 in ascending order:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
2	3																										
3	19																										
4	10																										
5	1																										
6	21																										
7	22																										
8	8																										
9	16																										
10	4																										
11	2																										
12	12																										
13	11																										
14	25																										
15	24																										
16	7																										
17	20																										
18	5																										
19	18																										
20	14																										
21	13																										
22	17																										
23	23																										
24	15																										
25	6																										
26	9																										
27																											

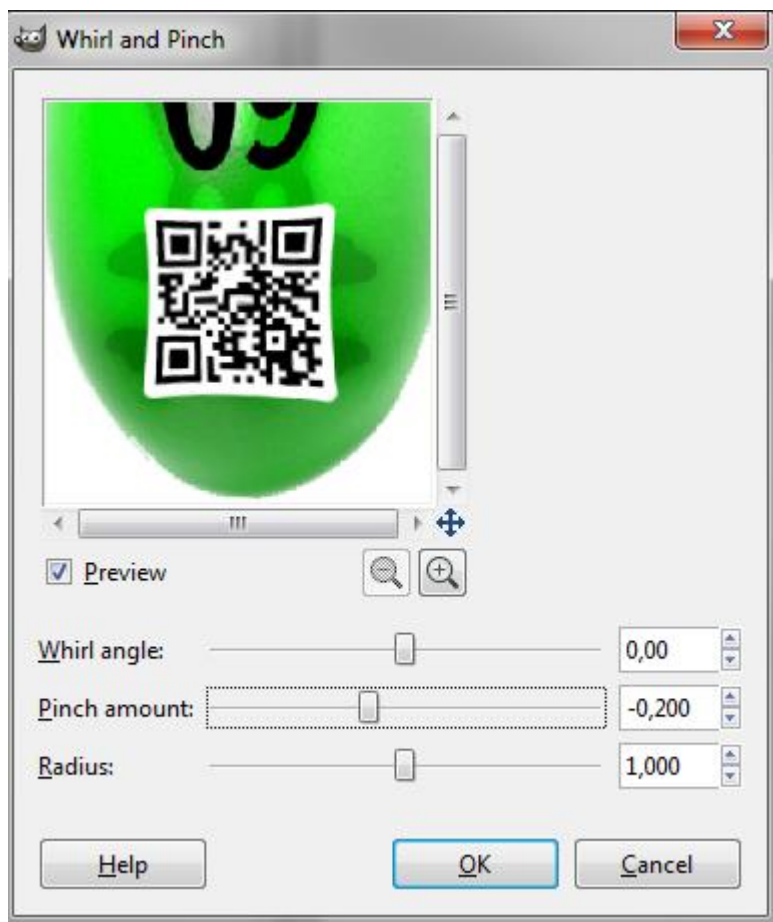
Now make a filter over A-Z and sort the column in ascending order. And now the QR-Code appear:

Challenge 9 – Fisheye

If you start the hackyeaster app, right at the beginning you will see the egg number 9. Make a screenshot or something like this. You can also unpack the apk to get the image.



I used in GIMP the Whirl and Pinch Filter to get some results:



Challenge 10 - Thumper's Den

After I realized, that I am able to look at the baskets of our players, I came up with the idea to look into Thumpers basket to:

<http://hackyeaster.hacking-lab.com/hackyeaster/eggs.html?name=Thumper>

Egg Basket of Hacker Thumper



Challenge 11 - You've got Mail

Extract the zip file and open the inbox file with a text-editor

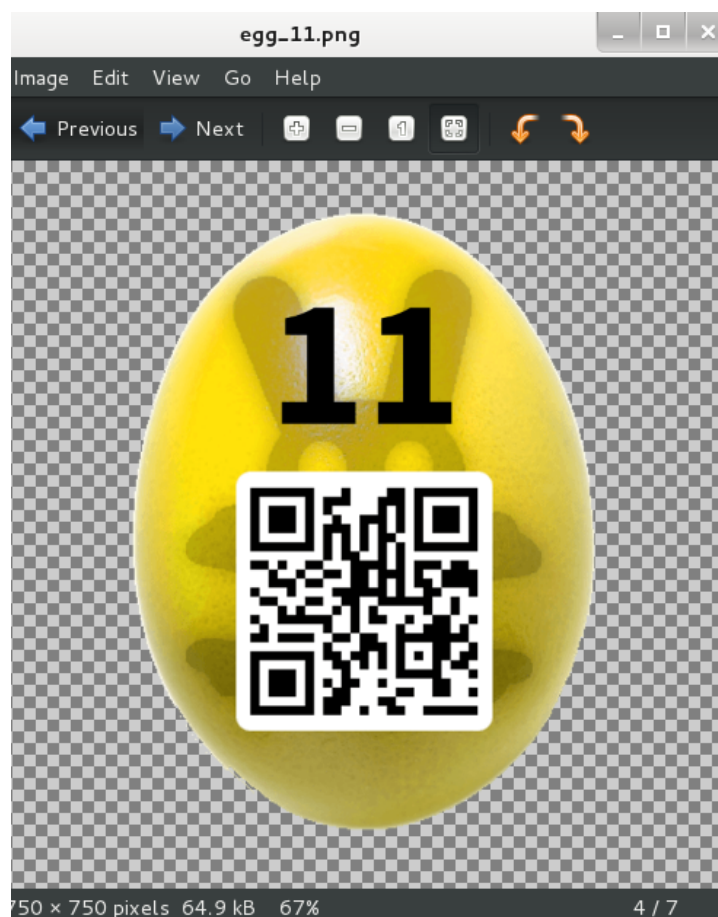
On line 311 you will see the beginning of the attachment:

```
--047d7b4501642dc6f905043957bd--  
--047d7b4501642dc6fe05043957bf  
Content-Type: application/zip; name="signature.zip"  
Content-Disposition: attachment; filename="signature.zip"  
Content-Transfer-Encoding: base64  
X-Attachment-Id: f_i0o7q80j0  
  
UESDBBQAAAAIAJ2iMUVXUT5FQfwAAJP9AAAKAAAAZWdnXzExLnBuZ3xYdzQbXhhuKmoU  
1apNW7PU  
VlSM+qm9WrR+paFq7y2IUbX3rhqhpShae0fsvfdKImLWSGImCPH1+/8733vO/ePec+57  
znvPc5/n  
ed/YVwaad2jYaW7cuHFHW+u10Y0bN7H/XbfJ/57091if3bhBfuOVnon63+35ERq92rOx  
PoBYqD46  
gG9sDJ6erB4dIE+O0Qe45Y6OBpzJGv50vb8/k3C61t+X3tOdjEI0w5eqr6+v/16/XqPp  
+UXVU0zX  
[...]
```

Save the base64-block in a separate file. The following command in Unix will help you, to decode the base64 block in the zip file:

```
"base64 -i -d base64encoded > signature.zip"
```

After extracting the zip you will get the image:



Challenge 12 - This is just a Test

The test form is really bad designed. The answers of the test are the following:

Question 1

What is the name of the popular port scanner, implemented by Fyodor?

Answer: nmap

Question 2

In the context of PKI systems, the shorthand "CRL" stands for "certificate _____ list".

Answer: revocation

Question 3

A group of 100 people plans to use symmetric encryption for secure communication. How many keys are needed to let everybody communicate with each other?

Answer: 4950

Question 4

Which hash sizes are supported by the SHA2 family? Choose two!

384 bit and 512 bit

Question 5

Which port number is used by Kerberos?

Answer: 88

Also you have the hidden field success, which is always false. Manipulate the request and send it with HttpRequister or something like this to the ceh site:

The screenshot shows the HttpRequister tool interface. The REQUEST tab is active, displaying a POST request to `http://hackyeaster.hacking-lab.com/hackyeaster/ceh?q1=nmap&q2=rev...`. The Parameters section contains the following data:

Name	Value
q1	nmap
q2	revocation
q3	4950
q4	512
q5	384
q6	88
success	true

The RESPONSE tab shows a 200 OK status with the following HTML content:

```
<article class= "box post" >
<header id="challenge-header"></header>
<script>addChallengeHeader(</script>
<p>
Congratulations, you passed!!!
</p>
<p>
<div class="eggImage" style="background-image: url('/images
/egg_12_an8snbui1FeLO3Ugw89N.png')"></div>
</p>
</article>
</div>
</div>
</div>
```

The HEADERS section shows the following details:

Header	Value
Date	Sat, 02 May 2015 15:01:15 GMT
Server	Apache/2.4.6 (CentOS) PHP/5.4.16
Access-Control-Allow-Origin	*
Access-Control-Allow-Methods	POST, GET, OPTIONS
Access-Control-Allow-Credentials	true
Access-Control-Allow-Headers	ACCEPT, ORIGIN, X-REQUESTED-WITH, CONTENT-TYPE, AUTHORIZATION
Accept-Ranges	bytes

The HISTORY tab shows the request details:

Request	Response	Date	Size	Time
POST http://hackyeaster.hacking-lab.com/hackyeaster/ceh?q1=nmap&q2=revocation&q3=4...	200 OK	Aug 2 2015 - 5:55:53 PM	1487 B	218 ms

In the image above you see the image URL of the solution egg:

http://hackyeaster.hacking-lab.com/hackyeaster/images/egg_12_an8snbui1FeLO3Ugw89N.png



Challenge 13

First I converted the MP4 into images.

```
ffmpeg -i leety.mp4 -r 1.0 leet%4d.jpg
```

After I got all the images (901 images) I decided to scan some Images from 450 (middle point of 900) up to 550 manually. I think image 525 was the right one and I got some points 😊



Challenge 14 - Wise Rabbit's Return

The read the barcode with the following Website: <http://www.onlinebarcodereader.com/de.html>



The decoded barcode was "yckgKB2iV1rvNEfCoNiR".

This decoded content must be now converted in a QR-Code. I do this with the following site:

<http://goqr.me/de/>

If you scan the created QR-Code you got the points.



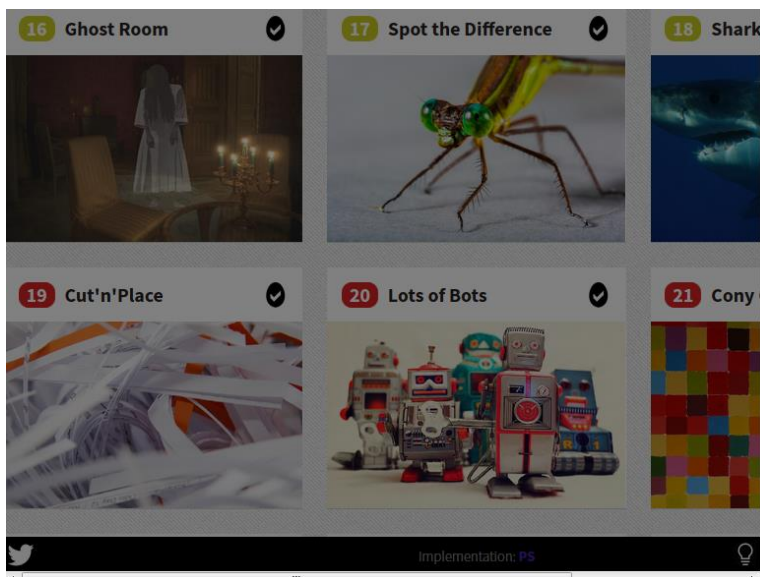
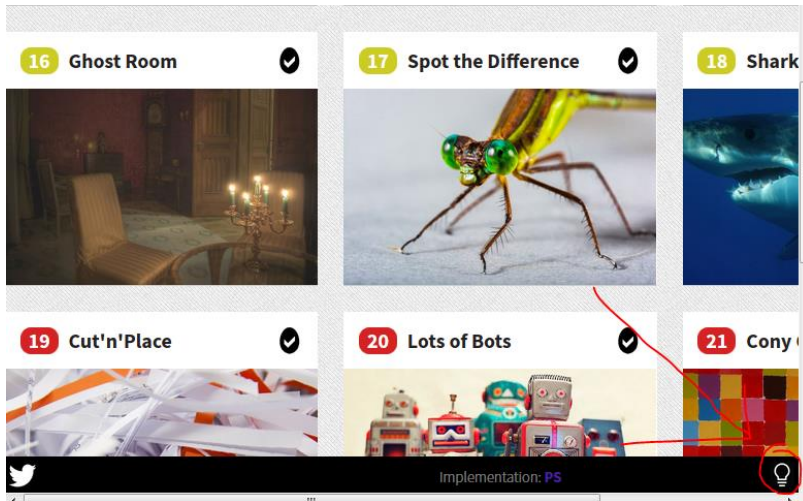
Funny fact, I solved the challenge the second way and didn't noticed it. After a while I flipped out, because the QR-Code of the Images doesn't worked. I looked into my basket and saw that the challenge was solved some hours ago :D



Decoded Image of the double-encoded base64 string

Challenge 16 - Ghost Room

In the Challenge Overview there is in the right bottom corner a light bulb which turns the light of:



When we turned off the light, we can go to the main part of the challenge:

Ghost Room

Dark is beautiful. A GOST with chaining appears and has a message for you:

```
d5++xytj6RiGwmqEecm63Kow7RZGAAHh  
VFskshFuj/Anap7pWHDZ1XQw8DAApUEN  
R5ExOGUKTzG0tvSA1CHkHq6NneL6ZUTX  
ej8Taxz+kHK9w9U8dxTOSksZ4HKS2YYD
```



spooky

There is a good hint. It is the GOST-Encryption. The password is "spooky". I found an online decoder here: http://www.tools4noobs.com/online_tools/decrypt/

Encrypted Text:

```
d5++xytj6RiGwmqEecm63Kow7RZGAAHh  
VFskshFuj/Anap7pWHDZ1XQw8DAApUEN  
R5ExOGUKTzG0tvSA1CHkHq6NneL6ZUTX  
ej8Taxz+kHK9w9U8dxTOSksZ4HKS2YYD
```

Key: spooky

Decrypted Text: http://hackyeaster.hacking-lab.com/hackyeaster/images/egg_16_a3ellACKSy02sJ6LxXeh.png

And now we have our next solution 😊

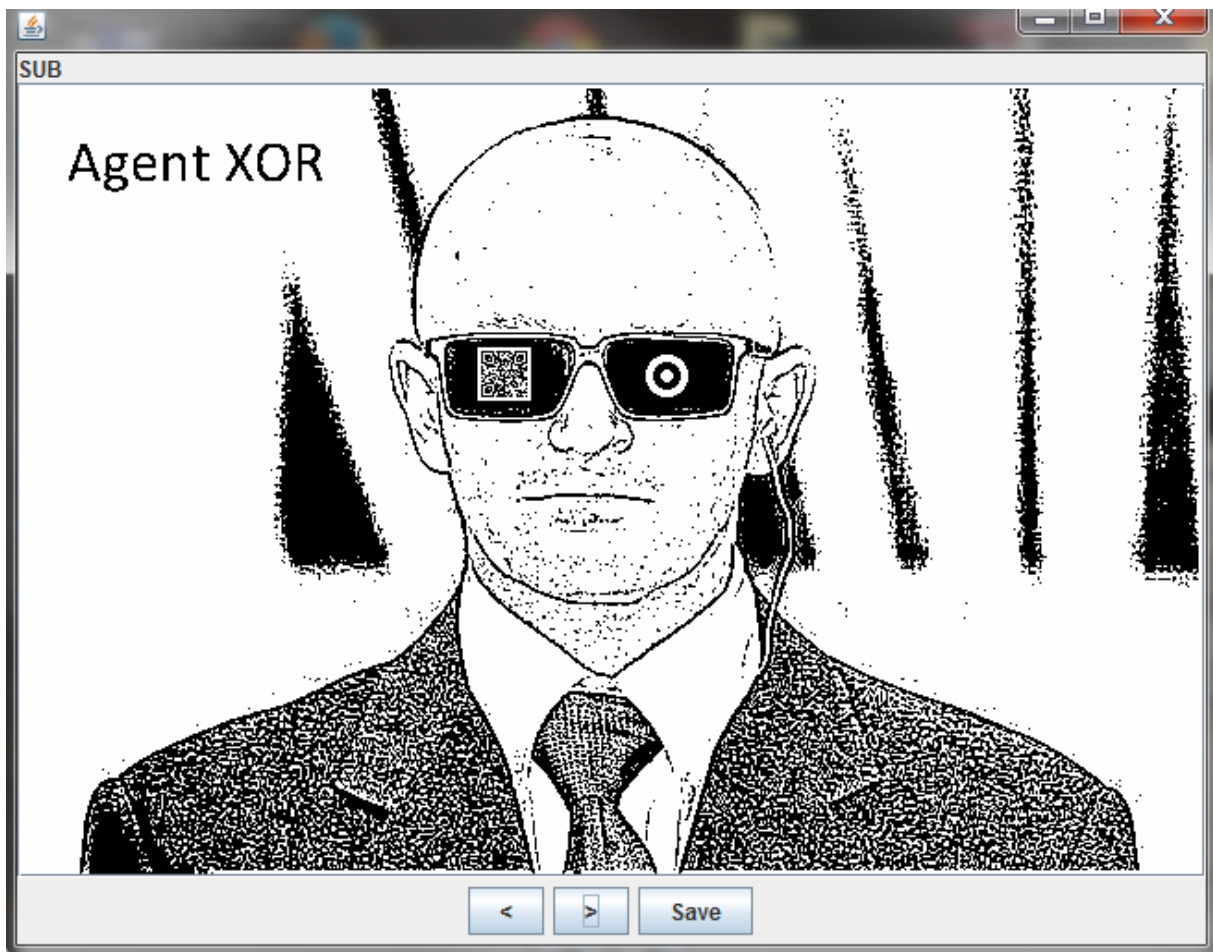


Challenge 17 - Spot the Difference

We got two images:



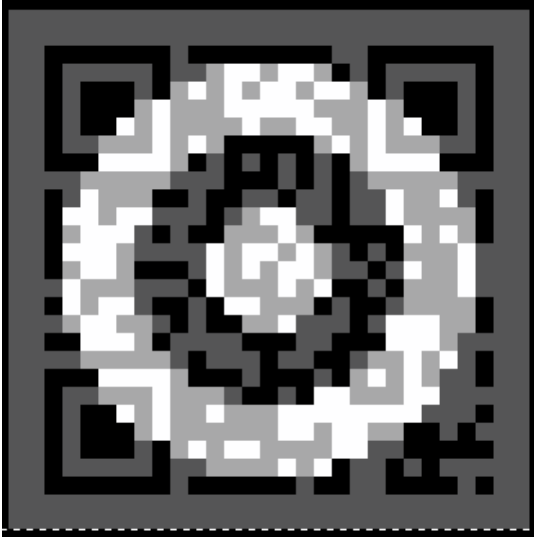
To get the difference, we play a little bit with the awesome stegano tool “Stegsolve” from Caesum. In this Java-Tool you find the Image-Combiner. First load the difference1.bmp in to the tool and chose the image-combiner with the difference2.bmp:



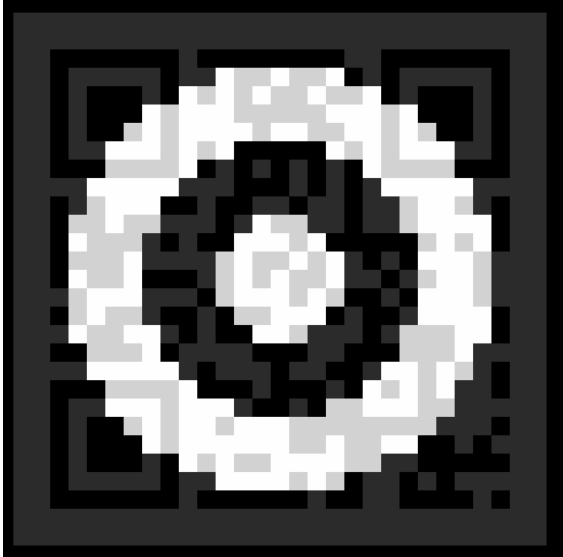
Save the image and look Mr. “Agent XOR” in the eyes:



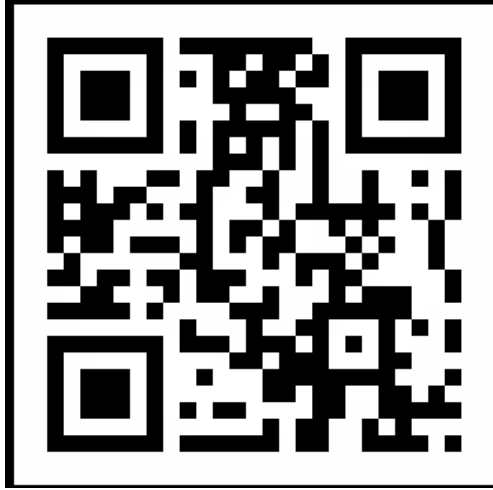
That is not a real QR-Code, but when you compare the right eye, with the left you will notice that some Pixels in the QR-Code are inverted like the right eye. Invert all Pixels in the white area and you get the right QR-Code. See the pictures to understand, I don't know how to describe it in better English 😊



Broken QR-Code Left eye + overlay of right eye



Invert all pixels in white area



Solution 😊

Challenge 18 - Sharks on the Wire

Download the pcap file and follow the last TCP-Stream (Wireshark Filter: "tcp.stream eq 3"). You will find really fast the Header with the Basic-Authorization:

```
GET /hackyeaster/sharks/sharks.css HTTP/1.1
Host: 10.11.0.48:8080
Connection: keep-alive
Authorization: Basic c2hhcmttYW46c2hhcmtzX2hhdmVfajR3cw==
Accept: text/css,*/*;q=0.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36
Referer: http://10.11.0.48:8080/hackyeaster/sharks/sharks.html
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,de;q=0.6,it;q=0.4
```

Decode the Authorization Part with base64 and you get the username:password
"sharkman:sharks_have_j4ws"

Now we can Login in and look at the website with another login-panel. The last header in the TCP Stream looks like this:

```
POST /hackyeaster/sharks/auth HTTP/1.1
Host: 10.11.0.48:8080
Connection: keep-alive
Content-Length: 82
Cache-Control: max-age=0
Authorization: Basic c2hhcmttYW46c2hhcmtzX2hhdmVfajR3cw==
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
;q=0.8
Origin: http://10.11.0.48:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: http://10.11.0.48:8080/hackyeaster/sharks/sharks.html
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8,de;q=0.6,it;q=0.4
```

```
user=supershark&pass=hashed%21%21%21&hash=b3f3ca462d3fa58b74d6982af14d8841b074994a
```

We see user supershark logging in with the hash b3f3ca462d3fa58b74d6982af14d8841b074994a. You don't to crack this, because you need only the hash (Pass the Hash-Attack). I manipulate the POST-Request with Tamper-Data:

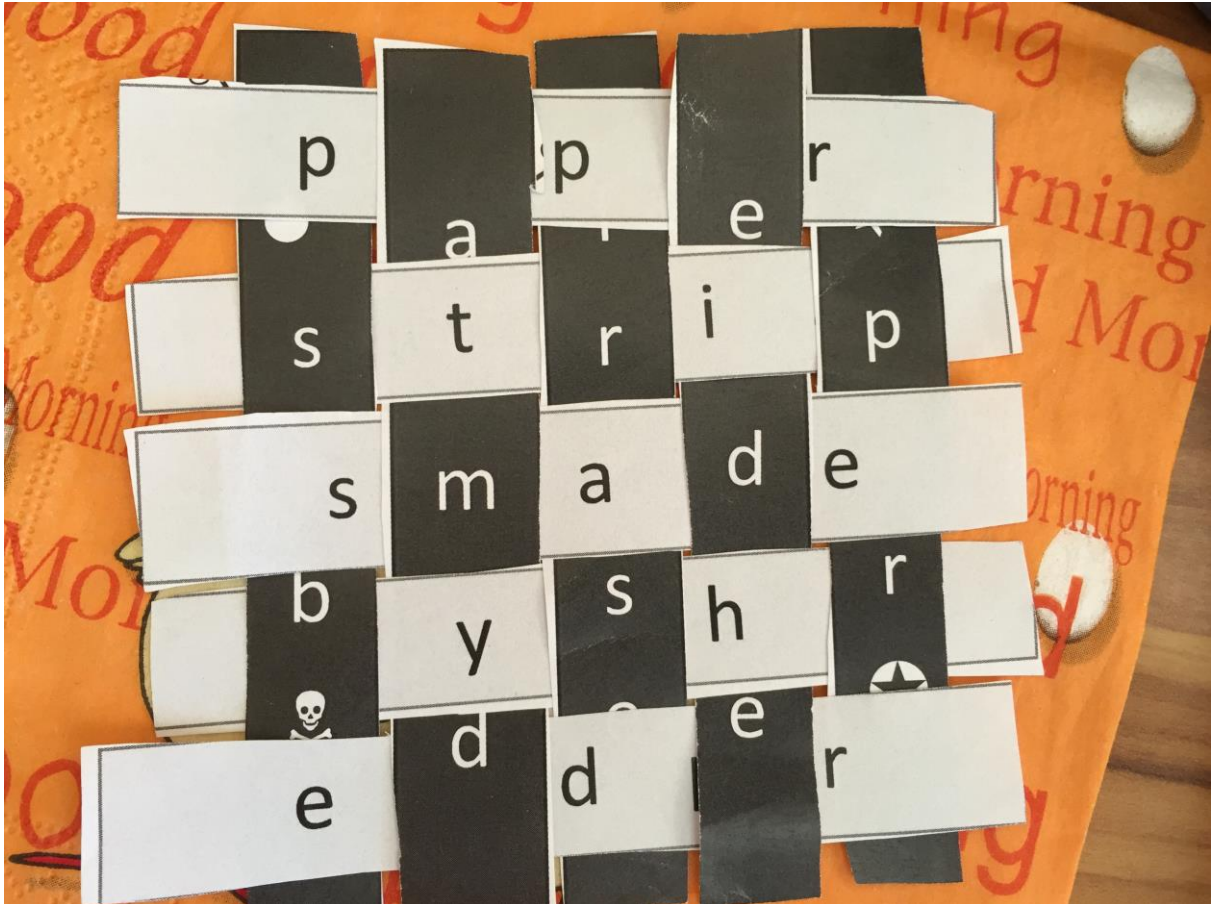
POST-Parameter Name	POST-Parameter Wert
user	supershark
pass	hashed%21%21%21
hash	62d3fa58b74d6982af14d8841b074994a

After sending the POST-Request, we got the solution:



Challenge 19 - Cut'n'Place

On this challenge, the biggest hint was that it is a passphrase and not a password. A passphrase is something like 20-40 Characters long. And now it is time for paper and scissors. Print the pdf and play a little bit with the stripes. After a while I come up with the idea, that I weave the paper:



The Solution was: paperstripsmadebyshredder

Awesome Challenge and the last one I finished 😊

Challenge 20 - Lots of Bots

"Robots have placed an egg on this web server. If you wanna find it, you need to think and act like a bot."

OK, let's have a look at robots.txt:

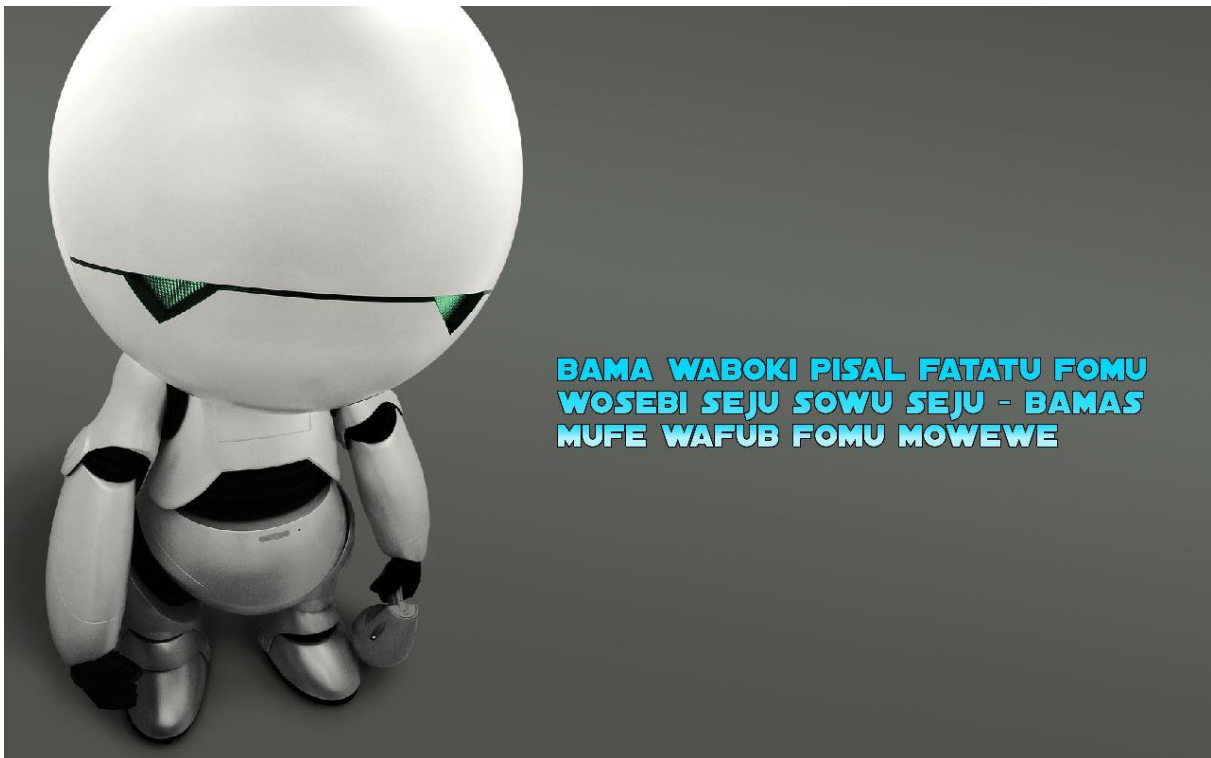
```
User-agent: EasterBot
Disallow: /
Allow: /hackyeaster/bots/bots.
```

```
User-agent: *
Disallow: /
```

The only thing you must add is "html". So we got the following site:

<http://hackyeaster.hacking-lab.com/hackyeaster/bots/bots.html>

Oh I forgot to tell you, that you must deactivate JavaScript. Otherwise you will land on a Wikipedia article of some famous bots ☺

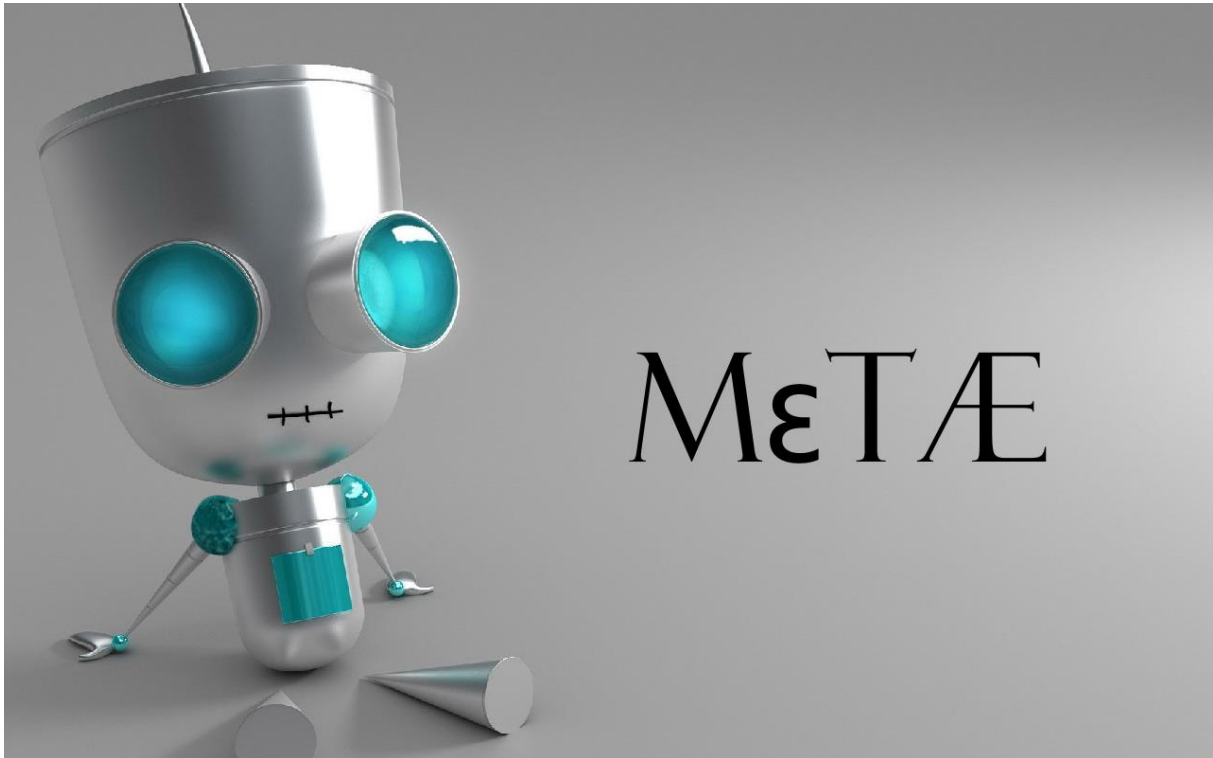


"BAMA WABOKI PISAL FATATU FOMU WOSEBI SEJU SOWU SEJU - BAMAS MUFE WAFUB FOMU MOWEWE"

After some research in google I found out, that the language is ROILA, the Robot Interaction Language. The translated text is the following:

"you must make word of addition two and two - this be name of page"

Two plus two is four. So maybe we should try <http://hackyeaster.hacking-lab.com/hackyeaster/bots/four.html>



OK let's have a look at the META-DATA.

```
<meta name="description" content="Robots talk in ROILA language:  
eman egap eht esrever tsum">  
<meta name="keywords" content="secret, page, robots, fun, hacky  
easter, blrt, five, beep">
```

“eman egap eht esrever tsum” that is not ROILA. If you try to read backwards you get “must reverse the page name”.

Reversing four is ruof and this is the site: <http://hackyeaster.hacking-lab.com/hackyeaster/bots/ruof.html>



Challenge 21 - Cony Code

The hint tells us everything:

“Hint: 110 is blue, the rest's up to you...”

First at all I counted all colors:

1. White
2. Red
3. Pink
4. Black
5. Yellow
6. Blue
7. Light blue
8. Light green

Eight different colors! Fits perfectly in 3 bits. After this I came up with the idea of RGB, because all colors in the cony-code are special colors in RGB.

RGB

110 - BLUE

You see, 1 means off and 0 means on. So my theory was red must be 011 and green must be 101. If you going on, you get the following binary table:

RGB

110 - Blue

011 - Red

101 - Green

001 - Yellow

010 - Pink (Light Red)

000 - White

111 - Black

100 - Light Blue

Now you have the cony-code image. Replace the colors with the binary value. Read it line by line from left to right.

Now you will get the following binary code:

```
011010000111010001110100011100000011101000101111001011110110100001100001011000110
110101101111001011001010110000101110011011101000110010101110010001011100110100001
100001011000110110101101101001011011100110011100101101011011000110000101100010001
011100110001101101111011011010010111101101000011000010110001101101011011110010110
010101100001011100110111010001100101011100100010111101101001011011010110000101100
111011001010111001100101111011001010110011101100111010111110011001000110001010111
110110101000110111011001110011011000110111010110100010111001110000011011100110011
100100000111111111111
```

If you converted this binary code to text you get the following output:

http://hackyeaster.hacking-lab.com/hackyeaster/images/egg_21_i7g67Z.png 

And there is our solution.



Challenge 22 - Hashes to Ashes

If you are familiar with hashcat, the task should be not a problem for you. I used cuda-hashcat on windows for the following commands:

Hash: ADA2EEEEBE7809857A57F6FEE4B2FFAEE24EAE7B1 (SHA1)

Command: cudaHashcat64 -m 100 -a 3 -o cracked.txt hashes.txt -1 1234567890
?1?1?1?1?1?1?1?1?1?1

Solution: 1199019170177790

Hash:

6BBF7528D9DD2959A7AFB37898425F67555F67F677987CAE7E86210A2C8A0DBDFC248EC2D7B2401
0F440BADC2223B4B5 (SHA512)

Command: cudaHashcat64 -m 1700 -a 3 -o cracked.txt hashes.txt -1 abcdefghijklmnopqrstuvwxyz
?1?1?1?1?1?1?1?1?1?1?1?1

Solution: hopelessly

Hash: B80814C5E0F386B0637163FD8AFE9A929 (MD5)

Command: cudaHashcat64 -m 0 -o cracked.txt -r hackyeaster_upper.rule -r hackyeaster_subst.rule
hashes.txt wordlist.txt ?s?d

hackyeaster_subst.rule:

sa@,sb8,sc(,sd6,se3,sf#,sg9,sh#,si!,si1,sk<,sli,sl1,so0,sq9,ss5,ss&,st+,sv>,sv<,sx%,
(Attention, replace the comma with a line break)

hackyeaster_upper.rule : T0,T1,T2,T3,T4,T5,T6,T7,T8,T9,TA,TB,TC,TD,TE,TF

(Attention, replace the comma with a line break)

Solution: Disc0very.5

Before I got to the last one, I created some custom dictionaries. The first one was that all words are lowercase (wordlist_lower.txt). After this I used the “prince processor” from hashcat, that I can merge the dictionaries:

```
pp64 -o wordlist_lower_doubleword.txt --elem-cnt-max=2 <  
wordlist_lower.txt
```

This dictionary contains all words in lower case and also a mix, that are all words are combined with each other. With this wordlist I am able to combine all passwords up to 4 times 😊

Hash: 9791CBE0AE919A0330994A2D6BA26B8F0C3A1DA15C73BCE5FCA39495881A6C90 (SHA256)

Command: cudaHashcat64 -m 1400 -a 1 -o cracked-txt hashes.txt wordlist_lower_doubleword.txt
wordlist_lower_doubleword.txt

Solution: enginebulbgoatimportant

Sadly most of the passwords appears on google after some time. So to solve most of them you only need google, but this way is the better way to learn how hashcat works.

Challenge 23 - Beat the Nerd Master

This was my favorite challenge. Old school Monkey Islands fights, with nerd questions and answers. I write a little python script to beat the nerd master:

```
#!/usr/bin/env python

import socket

questions=[]
questions.append("Go 127.0.0.1 to your mummy.")
questions.append("Go play with your toys, yellow-belly.")
questions.append("I have more friends than you.")
questions.append("This fight is like a hash function - it works in one
direction only.")
questions.append("You should leave your cave and socialize a bit.")
questions.append("You'll be 0xdeadbeef soon.")
questions.append("Af7ter thls flgh7, I wlll pwn ur b0x3n.")
questions.append("I bet you don't even understand binary.")
questions.append("You must be jealous when seeing my phone's display.")
questions.append("format C:")

answers=[]
answers.append("Won't work. I only support IPv6.")
answers.append("That's not even close to be nerdy!")
answers.append("Yeah, but only until you update your Facebook profile with
a real picture of you!")
answers.append("Too bad you picked LM hashing.")
answers.append("I'm not anti-social. I'm just not user friendly.")
answers.append("Not as long as I have my 0xcafebabe.")
answers.append("Check your settings - you seem to have chosen the Klingon
keyboard layout.")
answers.append("Sure I do. Me and you, we are 10 different kind of
persons.")
answers.append("Not really - Your pixels are so big, some of them have
their own region code!")
answers.append("Specified drive does not exist.")

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('hackyeaster.hacking-lab.com', 1400))
data=""
while data == "":
    data = s.recv(1024)
print data
send = "y\r\n"
s.send(send)

while data == "":
    data = s.recv(4096)
print data
data=""
while data == "":
    data = s.recv(4096)
print data

send = questions[9].join("\r\n")
print send
s.send(send)
```

```
data=""
while data == "":
    data = s.recv(4096)
print data

data=""
while data == "":
    data = s.recv(4096)
print data

s.close()
```


Challenge 24 - SHAM Hash

Everything is described in the PDF File. I write a little bruteforce programm in C#. I used also a C# implementation of HashLib. The Main Idea is, that this encryption only have a protection of 6x6 character password. Cracking 6 different 6 character long passwords is not so complicated:

```
void attack(int beginsix, int steps)
{
    string brutestring = "abcdefghijklmnopqrstuvwxyz1234567890!?" ;
    string test = "hackyeaster2015isforeveryone!!";
    string brutinhash = "757c479895d6845b2b0530cd9a2b11";
    steps = (steps == -1)?brutestring.Length-beginsix:steps;
    string[] splitter = new string[5];
    string[] hashsplitter = new string[5];
    for (int i = 0; i < splitter.Length; i++) {
        splitter[i] = test.Substring(i*6,6);
        hashsplitter[i] = brutinhash.ToUpper().Substring(i*6,6);
    }

    IHash MD2 = HashFactory.Crypto.CreateMD2();
    IHash MD5 = HashFactory.Crypto.CreateMD5();
    IHash SHA1 = HashFactory.Crypto.CreateSHA1();
    IHash SHA256 = HashFactory.Crypto.CreateSHA256();
    IHash SHA512 = HashFactory.Crypto.CreateSHA512();
    string lolo="";
    HashResult r = MD2.ComputeString(splitter[0], Encoding.ASCII);
    lolo += BitConverter.ToString(r.GetBytes()).Replace("-", "").Substring(0,6);
    r = MD5.ComputeString(splitter[1], Encoding.ASCII);
    lolo += BitConverter.ToString(r.GetBytes()).Replace("-", "").Substring(6,6);
    r = SHA1.ComputeString(splitter[2], Encoding.ASCII);
    lolo += BitConverter.ToString(r.GetBytes()).Replace("-", "").Substring(12,6);
    r = SHA256.ComputeString(splitter[3], Encoding.ASCII);
    lolo += BitConverter.ToString(r.GetBytes()).Replace("-", "").Substring(18,6);
    r = SHA512.ComputeString(splitter[4], Encoding.ASCII);
    lolo += BitConverter.ToString(r.GetBytes()).Replace("-", "").Substring(24,6);

    int[] char_counter_arr = {beginsix,0,0,0,0,0};
    bool[] notcracked = {true,true,true,true,true};
    string temp = "";
    string answer = "";
    while (char_counter_arr[0] < beginsix+steps) {
        string word = "" +
            brutestring[char_counter_arr[0]] +
            brutestring[char_counter_arr[1]] +
            brutestring[char_counter_arr[2]] +
            brutestring[char_counter_arr[3]] +
            brutestring[char_counter_arr[4]] +
            brutestring[char_counter_arr[5]];

        if (notcracked[0]) {
```

```
r = MD2.ComputeString(word, Encoding.ASCII);
temp = BitConverter.ToString(r.GetBytes()).Replace("-", "");
if (temp.Substring(0,6) == hashsplitter[0]) {
    answer += "MD2: " + word + "\r\n";
    notcracked[0] = false;
}
}
```

```
if (notcracked[1]) {
    r = MD5.ComputeString(word, Encoding.ASCII);
temp = BitConverter.ToString(r.GetBytes()).Replace("-", "");
if (temp.Substring(6,6) == hashsplitter[1]) {
    answer += "MD5: " + word + "\r\n";
    notcracked[1] = false;
}
}
```

```
if (notcracked[2]) {
    r = SHA1.ComputeString(word, Encoding.ASCII);
temp = BitConverter.ToString(r.GetBytes()).Replace("-", "");
if (temp.Substring(12,6) == hashsplitter[2]) {
    answer += "SHA1: " + word + "\r\n";
    notcracked[2] = false;
}
}
```

```
if (notcracked[3]) {
    r = SHA256.ComputeString(word, Encoding.ASCII);
temp = BitConverter.ToString(r.GetBytes()).Replace("-", "");
if (temp.Substring(18,6) == hashsplitter[3]) {
    answer += "SHA256: " + word + "\r\n";
    notcracked[3] = false;
}
}
```

```
if (notcracked[4]) {
    r = SHA512.ComputeString(word, Encoding.ASCII);
temp = BitConverter.ToString(r.GetBytes()).Replace("-", "");
if (temp.Substring(24,6) == hashsplitter[4]) {
    answer += "SHA512: " + word + "\r\n";
    notcracked[4] = false;
}
}
```

```
char_counter_arr[5]++;
if (char_counter_arr[5] == brutestring.Length) {
    char_counter_arr[5]=0;
    char_counter_arr[4]++;
}
```

```

}
if (char_counter_arr[4] == brutestring.Length) {
    char_counter_arr[4]=0;
    char_counter_arr[3]++;
}
if (char_counter_arr[3] == brutestring.Length) {
    char_counter_arr[3]=0;
    char_counter_arr[2]++;
}
if (char_counter_arr[2] == brutestring.Length) {
    char_counter_arr[2]=0;
    char_counter_arr[1]++;
}
if (char_counter_arr[1] == brutestring.Length) {
    char_counter_arr[1]=0;
    char_counter_arr[0]++;
}
}
}

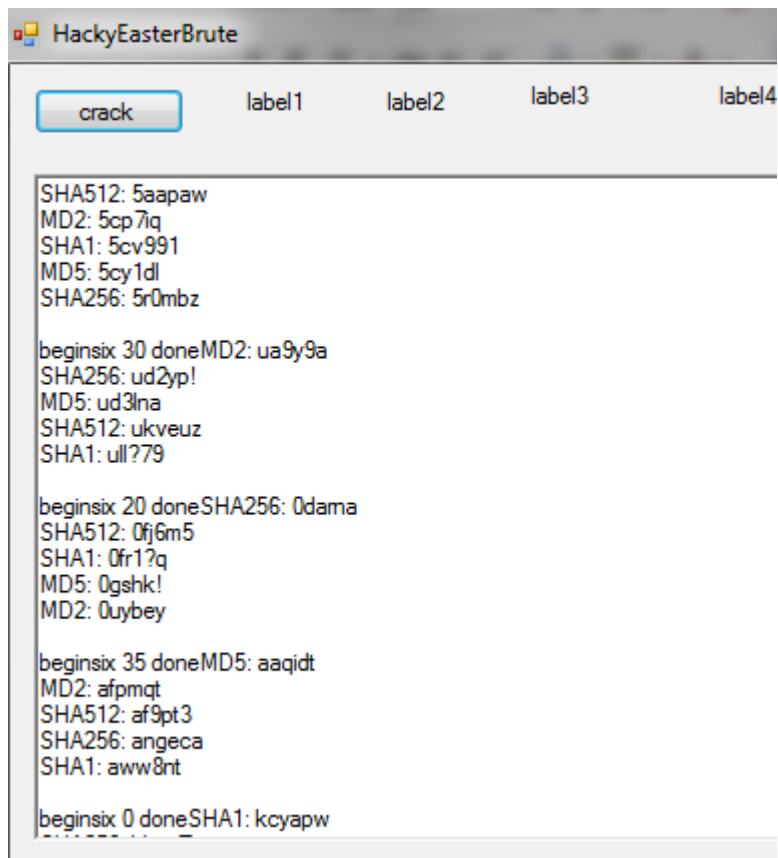
```

```

MethodInvoker inv = delegate{txt_debug.Text += answer + "\r\nbeginsix " + beginsix + "
done"};
    this.Invoke(inv);
}

```

The Code is really dirty. But I got with it the following solution with it:



I took the first password: "5cp7iq5cy1dl5cv9915r0mbz5aapaw"

Congrats!

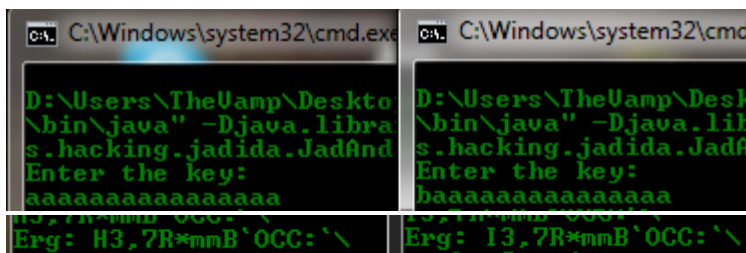


Challenge 25 - Jad & Ida

Unpack the ZIP file and analyze the JadAndIda.class file with JD-GUI. You will find on line 43 the following code:

```
42     BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
43     String k = in.readLine();
44     String h = k;
45     for (int z = 0; z < 10; z++) {
46         h = fizzle(rizzle(shizzle(bizzle(h))));
47     }
48     if ("v30] pmWm<Y(0=21".equals(h))
49     {
50         System.out.println("Congrats!");
51         byte[] plain = Files.readAllBytes(Paths.get("s3cr3t.bin", new String[0]));
52     }
```

So you see, the program does something with your input and if the result is "v30] pmWm<Y(0=21" you solved the challenge. Export the source code and try to compile it with your own code. I added the following line between line 47 & 48 "System.out.println(h) ; ", to see how the string is changed after the fizzle rizzle shizzle bizzle.



The „Erg“ stands for “Ergebnis”. That is the German word for solution.

As you see I only change the first letter, and only the first letter changed after the fizzle rizzle shizzle bizzle. The same thing happened with every other character. If you change the second character only the second character in the encryption will change. With this information, we know that we can easily bruteforce the password. I added the following code:

```
53     /* 53: 45 */         for (int z = 0; z < 10; z++) {
54     /* 54: 46 */             h = fizzle(rizzle(shizzle(bizzle(h))));
55     System.out.println(h);
56     /* 55: */         }
57     System.out.println("Erg: " + h);
58
59     String bruter = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890B?!";
60     String solver = "v30] pmWm<Y(0=21";
61     for(int bruterstring = 0; bruterstring < 16; bruterstring++){
62         for(int alda = 0; alda < 63; alda++){
63             String letters = "";
64             for(int neg = 0; neg < 16; neg++){letters += bruter.charAt(neg);}
65             String yoyo = letters;
66             for (int z = 0; z < 10; z++) {
67                 yoyo = fizzle(rizzle(shizzle(bizzle(yoyo))));
68             }
69             char sol1 = solver.charAt(bruterstring);
70             char sol2 = yoyo.charAt(bruterstring);
71             if(sol1 == sol2){
72                 System.out.println("numb: "+bruterstring+ " - "+bruter.charAt(alda));
73                 break;
74             }
75         }
76     }
77
78     /* 56: 48 */         if ("v30] pmWm<Y(0=21".equals(h))
79     /* 57: */         {
80     /* 58: 49 */             System.out.println("Congrats!");
81     /* 59: 50 */             byte[] plain = Files.readAllBytes(Paths.get("s3cr3t.bin", new String[0]));
```

The idea is, when I found the right character, the bruteforce can jump to the next letter ☺

After compiling this code, I got the following output:

```
D:\Users\TheVamp\Desktop\had
\bin\java" -Djava.library.pa
s.hacking.jadida.JadAndIda
Enter the key:
aaaaaaaaaaaaaaaaaaaa
.=Pe?:UeGHKPW`kx
=qHD%iP^\^Ri++/7C
FLXHDaemi=89ChA9
8AP-Hq`(>)hRNNUf%
&I`1giuh*Mz">6rt
3QXIdkXnUZY?77C<`
ad,o+q*ceBmELIH<
.:`N/$#rAof 11!
=y2RNK8^&7U.5ey=
H3.7R*mmB`OCC:`\
Erg- H3.7R*mmB`OCC:`\
numb: 0 - j
numb: 1 - a
numb: 2 - d
numb: 3 - n
numb: 4 - l
numb: 5 - d
numb: 6 - a
numb: 7 - l
numb: 8 - 0
numb: 9 - v
numb: 10 - e
numb: 11 - c
numb: 12 - o
numb: 13 - d
numb: 14 - 3
numb: 15 - n
nope!

D:\Users\TheVamp\Desktop\had
Drücken Sie eine beliebige T

D:\Users\TheVamp\Desktop\
\bin\java" -Djava.library
s.hacking.jadida.JadAndId
Enter the key:
jadnldal0vecod3n
7=S2.=U T9N^Ydzf
eqKQl1PgL+N-s2bP
21[pQdevvI;bB2TH
AAS:0S`bn ;qXY82
-[`Stlu??><.gYI*
!QlXsZn l/p(UAFgm
GJO<8S*QSNppiFze
\%` [vh#=KGp?fo=0
'yWz:z81uCYUto6G
v30l pmWm<Y<0=21
Erg- v30l pmWm<Y<0=21
numb: 0 - j
numb: 1 - a
numb: 2 - d
numb: 3 - n
numb: 4 - l
numb: 5 - d
numb: 6 - a
numb: 7 - l
numb: 8 - 0
numb: 9 - v
numb: 10 - e
numb: 11 - c
numb: 12 - o
numb: 13 - d
numb: 14 - 3
numb: 15 - n
Congrats!

D:\Users\TheVamp\Desktop\
Drücken Sie eine beliebig
```

Now we have the password: jadnldal0vecod3n and the s3cr3t.bin will be decrypted in an image file:



Challenge 26 - Clumsy Cloud

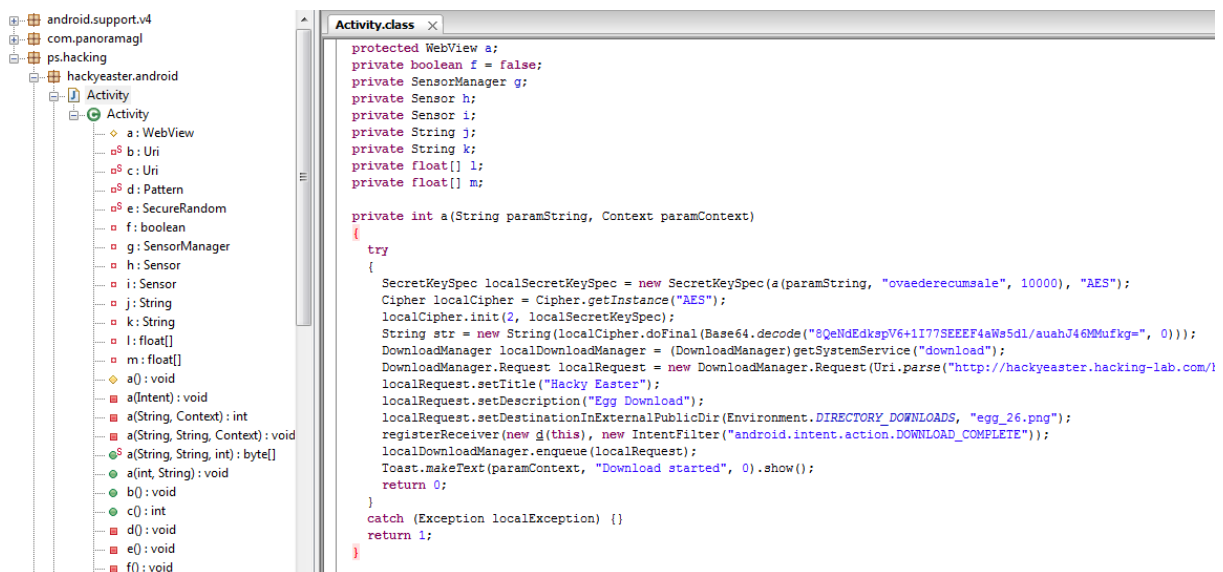
First let's have a look into this Backup file http://hackyeaster.hacking-lab.com/hackyeaster/files/passphrase_backup.txt :

```
{
  "name" : "Clumsy Cloud Backup",
  "comment" : "Backup of your passphrase, protected with your secret PIN.",
  "params" : {
    "s" : "ovaederecumsale",
    "h" : "1.3.14.3.2.26",
    "i" : 10000,
    "k" : 128,
    "e" : "2.16.840.1.101.3.4.1.1",
    "p" : "8QeNdEdkspV6+1I77SEEEF4aWs5dl/auahJ46MMufkg="
  }
}
```

After some research on google you will found out that "1.3.14.3.2.26" is the OID of SHA1 and "2.16.840.1.101.3.4.1.1" is the OID of AES128-ECB. So the Clumsy Cloud have something to do with SHA1 and AES128 in ECB mode.

In the second step we look into the APK File. Download the Hackyeaster APK file. You can download it with Evozi's downloader: <http://apps.evozi.com/apk-downloader/>

Decompile the apk with dex2jar and open the jar file with jd-gui. After some searching I found out, that the Activity class has our main functions for clumsy cloud:



```
Activity.class
protected WebView a;
private boolean f = false;
private SensorManager g;
private Sensor h;
private Sensor i;
private String j;
private String k;
private float[] l;
private float[] m;

private int a(String paramString, Context paramContext)
{
    try
    {
        SecretKeySpec localSecretKeySpec = new SecretKeySpec(a(paramString, "ovaederecumsale", 10000), "AES");
        Cipher localCipher = Cipher.getInstance("AES");
        localCipher.init(2, localSecretKeySpec);
        String str = new String(localCipher.doFinal(Base64.decode("8QeNdEdkspV6+1I77SEEEF4aWs5dl/auahJ46MMufkg=", 0)));
        DownloadManager localDownloadManager = (DownloadManager) getSystemService("download");
        DownloadManager.Request localRequest = new DownloadManager.Request(Uri.parse("http://hackyeaster.hacking-lab.com/?"));
        localRequest.setTitle("Hacky Easter");
        localRequest.setDescription("Egg Download");
        localRequest.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS, "egg_26.png");
        registerReceiver(new g(this), new IntentFilter("android.intent.action.DOWNLOAD_COMPLETE"));
        localDownloadManager.enqueue(localRequest);
        Toast.makeText(paramContext, "Download started", 0).show();
        return 0;
    }
    catch (Exception localException) {}
    return 1;
}
```

We also have a function with some SHA1 encoding:

```
public static byte[] a(String paramString1, String paramString2, int paramInt)
{
    MessageDigest localMessageDigest = MessageDigest.getInstance("SHA1");
    byte[] arrayOfByte1 = (paramString2 + paramString1).getBytes();
    for (int n = 0;; n++)
    {
        if (n >= paramInt)
        {
            byte[] arrayOfByte2 = new byte[16];
            System.arraycopy(arrayOfByte1, 0, arrayOfByte2, 0, 15);
            return arrayOfByte2;
        }
        arrayOfByte1 = localMessageDigest.digest(arrayOfByte1);
    }
}
```

This seems to be the main functions of clumsy cloud. We only need to implement this two functions and try to get the right pin. We have a pin with 4 digits. This mean, we are able to bruteforce the pin.

I exported the two functions and rewrite them, so they are able to run in Java. Additionally I needed the javax-crypto.jar. You can download it from here:

<http://www.java2s.com/Code/Jar/j/Downloadjavaxcryptojar.htm>

Here are my source code:

```
import java.io.Console;
import java.io.PrintStream;
import java.security.MessageDigest;
import java.security.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.xml.bind.DatatypeConverter;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.*;
import javax.crypto.*;

public class something
{
    public static void main(String[] paramArrayOfString)
    {
        for(int i=0; i<10000; i++){
            String paramString = String.format("%04d", i);
            try
            {
                String somebase64 =
"8QeNdEdkspV6+1I77SEEEF4aWs5dl/auahJ46MMufkg=";
                SecretKeySpec localSecretKeySpec = new
SecretKeySpec(a(paramString, "ovaederecumsale", 10000), "AES");
                Cipher localCipher = Cipher.getInstance("AES");
                localCipher.init(2, localSecretKeySpec);
```



```

        String str = new
String(localCipher.doFinal(DatatypeConverter.parseBase64Binary(someb
ase64)));
        System.out.println(paramString);
        System.out.println(str);
    }
    catch (NoSuchAlgorithmException localException) {}
    catch (NoSuchPaddingException nopad){ }
    catch (IllegalBlockSizeException blockshit) { }
    catch (BadPaddingException badpaddd) { }
    catch (InvalidKeyException invkey) { }
    }//END FOR
} //END FUNCTION

private static byte[] a(String paramString1, String paramString2,
int paramInt) throws NoSuchAlgorithmException
{
    try{
        MessageDigest localMessageDigest =
MessageDigest.getInstance("SHA1");
        byte[] arrayOfByte1 = (paramString2 + paramString1).getBytes();
        for (int n = 0;; n++)
        {
            if (n >= paramInt)
            {
                byte[] arrayOfByte2 = new byte[16];
                System.arraycopy(arrayOfByte1, 0, arrayOfByte2, 0, 15);
                return arrayOfByte2;
            }
            arrayOfByte1 = localMessageDigest.digest(arrayOfByte1);
        }
    }catch(NoSuchAlgorithmException someshit){
        System.out.println(someshit.toString());
        byte[] als = new byte[1];
        return als;
    }
}
}

```

Also quick and dirty. After starting the java application I got something like this:

```
0c0Rf\...
6390
6460
7113
wirestarter54321
7253
7256
7672
8255
8289
9314
9383
9852
9865
9880
D:\Program Files (x86)\Java\jdk1.7.0_
Drücken Sie eine beliebige Taste . .
```

The PIN 7113 seems to be the right PIN and we got the egg:



Challenge 27 - Too Many Time Pad

Using a One Time Pad with the same key is a bad thing. For solving this Challenge I used the following python scripts from SpiederLabs: <https://github.com/SpiederLabs/cribdrag>

The first step is, to XOR two intercepted Messages of your choice. For message 5 and 4 it is the following command:

```
python xorstrings.py 71c26929e96931698e2865d816d3624b687cd6  
6c8a7b6ce06a3161dd6a60d755d42d4d6d67
```

Now we get the XORed Hex-string 1d481245090300085342050f43074f06051b

```
python cribdrag.py 1d481245090300085342050f43074f06051b
```

We try to enter some random well known English words like "the" "at" or something like this:

Bit by bit I was able to decrypt the messages:

```
60c46964f83879618e2878de539f6f4a6271d716 //20chars  
    enemy hase the bonbon  
63c37a6ca177792092602cc553c9684b //16chars  
    five oh oh seven  
68d82c6bf4767f79dd617f9642d768057f63c1 //19chars  
    mr bunny is the spy  
6c8a7b6ce06a3161dd6a60d755d42d4d6d67 //18chars  
    i wear a black hat  
71c26929e96931698e2865d816d3624b687cd6 //19chars  
    the hq is in london  
6cda6d6df87764709c6c7bd357d361556d77 //18chars  
    ipadyoupadweallpad
```

And the last message is our solution:



ipadyoupadweallpad