Hacky Easter 2016 – Write-up

by Eash# - eduardo.sh@gmail.com – May 2016

Challenge 01

Easy One

As always, the first challenge is very easy. Even babies can solve this one!
Find the code and enter it in the Egg-O-Matic™ below! One word, all lowercase.

xt hex yhi dde nyy str
in gyy isy ymo lly cod
dl exy sox xsi mpl ey

Answer:

Putting all the letter in the same line and splitting "x", "xy", and "yy" showed the quote:
"the hidden string is mollycoddle so simple"

Password is: mollycoddle

Challenge 02

Just Cruisin'

Need a holiday? Book a cruise on our new flag ship!
Seek out the promotion code below (lowercase only, no spaces) and get a free welcome package!

Answer:

Using the International naval flag code was simple reach the code:

```
en?jo
y?afr
?eshs
eabrm
eez1e
```

Password: enjoyafreshseabreeze

FLAG and NAME	Spoken	Written	FLAG and NAME	Spoken	Written	FLAG and NAME	Spoken	Written
A	ALFA	A	M	MIKE	M	Y	YANKEE	Y
B	BRAVO	B	N	NOVEMBER	N	Z	ZULU	Z
C	CHARLIE	C	O	OSCAR	O	1	ONE	1
D	DELTA	D	P	PAPA	P	2	TWO	2
E	ECHO	E	Q	QUEBEC	Q	3	THREE	3
F	FOXTROT	F	R	ROMEO	R	4	FOUR	4
G	GOLF	G	S	SIERRA	S	5	FIVE	5
H	HOTEL	H	T	TANGO	T	6	SIX	6
I	INDIA	I	U	UNIFORM	U	7	SEVEN	7
J	JULIETT	J	V	VICTOR	V	8	EIGHT	8
K	KILO	K	W	WHISKEY	W	9	NINE	9
L	LIMA	L	X	XRAY	X	0	ZERO	0

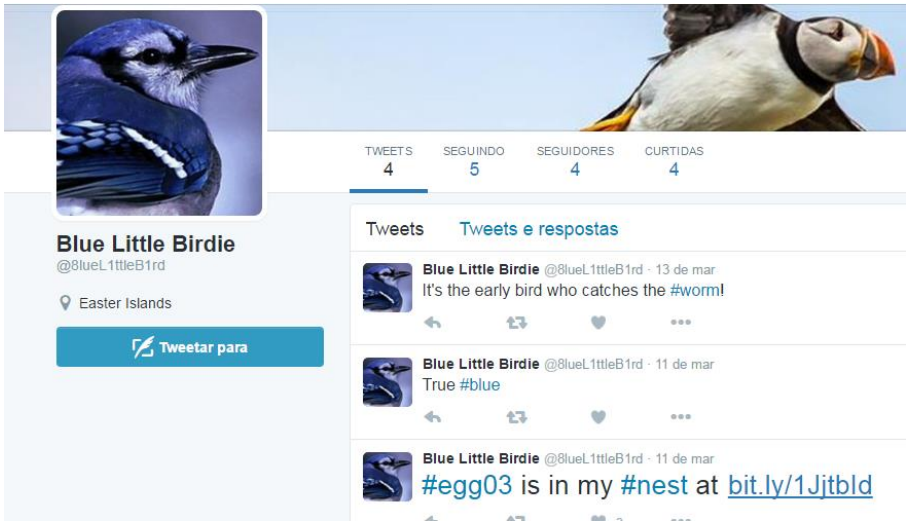
Challenge 03

Bird's Nest

The little bird has hidden an egg in its nest. Can you find it?

Answer:

The # is a clear Twitter reference. Searching on google by “twitter nest egg03” points to https://twitter.com/8lueL1ttleB1rd



Blue Little Birdie ?@8lueL1ttleB1rd 11 de mar

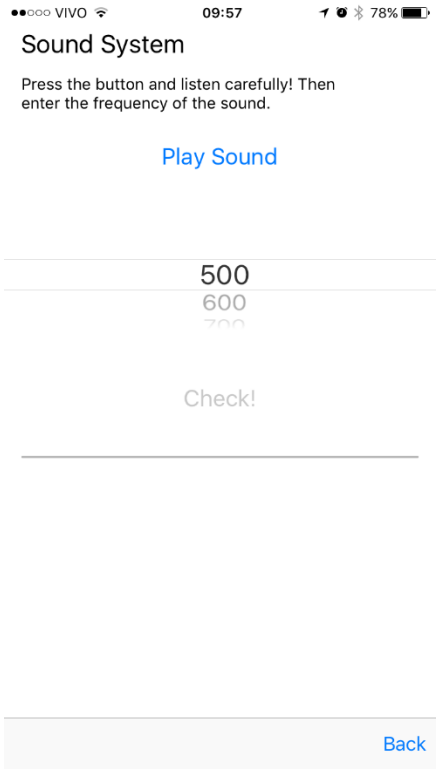
#egg03 is in my #nest at http://bit.ly/1JjtbId

Following the http://bit.ly/1JjtbId to http://hackyeaster.hacking-lab.com/hackyeaster/images/egg03_jPr5bJuNexwI4NsRkEcs.png revel the egg.

Challenge 04

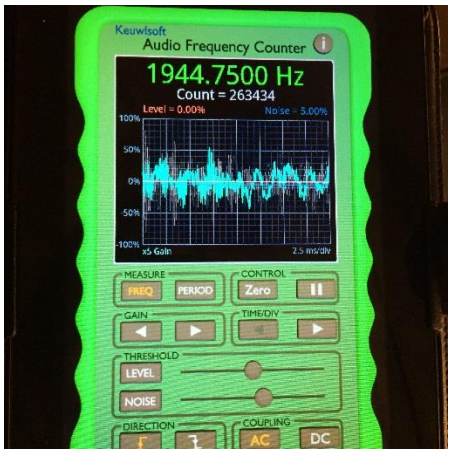
Sound Check

The new sound system needs a check. Sharpen your ears and listen carefully!



Answer:

I used an App on my mobile to decode the frequencies and grab the egg.



Challenge 05

Play it again, Paul

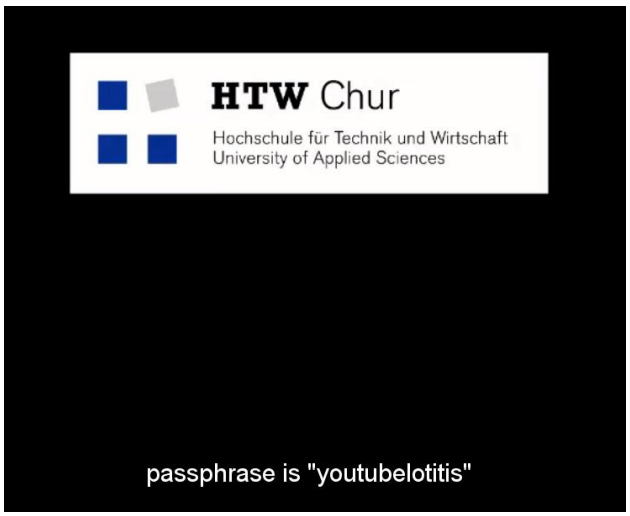
Do you know Paul? If not, it's about time to get to know him! Check out his video below!
Ĉu vi scias Paŭlo? Se ne, ĝi estas pri tempo ekkoni lin! Kontroli lian video sube!

Video at: http://media.hacking-lab.com/hackyeaster/he2016/video/video.mp4

Answer:

On VLC Player change the legends to esperanto following the hint "Cu vi parolas Esperanto?" that means “Do you speak Esperanto?”

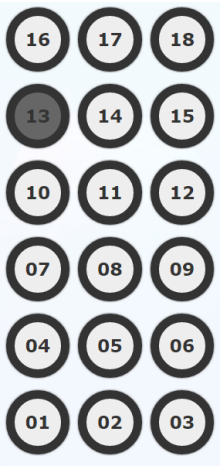
The passphrase showed on min 1:04 "youtubelotitis"



Challenge 06

Going Up

Time for an elevator ride. Guess the right floor, and find the hidden easter egg!



Answer:

Checking the link for each floor showed me that the floor 13 was ciphered in anyway.
http://hackyeaster.hacking-lab.com/hackyeaster/challenge06.html?sybbe=punatrzr

Piece of cake is Rot13: sybbe=punatrzr= floor=changeme

Following the tip to cipher the floor "thirteen" in rot13 and access the link.
http://hackyeaster.hacking-lab.com/hackyeaster/challenge06.html?sybbe=guvegrra

http://hackyeaster.hacking-lab.com/hackyeaster/challenge06_8ce833745ef167679aa44d6d6c098ab6abf8ec80.html

Challenge 07

Wise Rabbit Once More

Wise Rabbit says:

The solution is in the solutions!

Go back and scroll to 123!

Answer:

Very funny challenge. “The solution is in the solutions!” means the previous year solution write-up on http://media.hacking-lab.com/hackyeaster/HackyEaster2015_Solutions_high.pdf

“Go back and scroll to 123!” means the page 123. :)

Password is goldfish

```
        fi
done <text_neu2.txt
```

password: goldfish

Challenge 08

Just Drive

Answer:

Another very creative challenge. The trick here is “drive” using your mobile up to grab the egg.

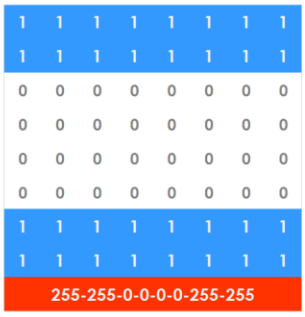
The solution is: Moving your mobile on the following way Left-Right-Right-Left-Right-Letf-Letf

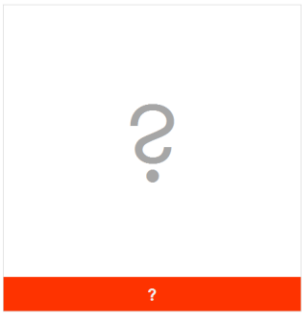
Challenge 9

Fish eye

Brain Game

What about a little brain game?



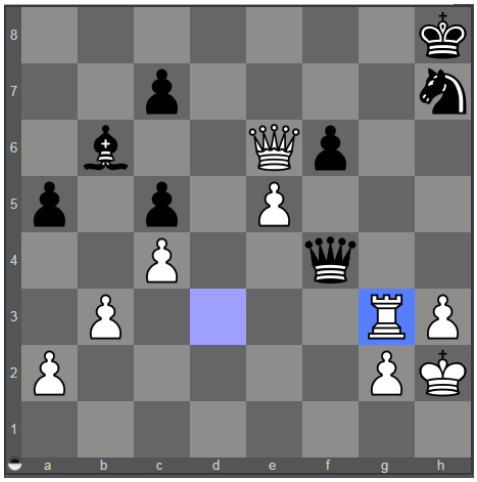


1. e4 e5 2. Nf3 Nc6 3. Bb5 Nf6 4. d3 Bc5 5. O-O d6 6. Nbd2 O-O 7. Bxc6 bxc6 8. h3 h6 9. Re1 Re8 10. Nf1 a5 11. Ng3 Rb8 12. b3 Bb4 13. Bd2 Ra8 14. c3 Bc5 15. d4 Bb6 16. dxe5 dxe5 17. c4 Nh7 18. Qe2 Nf8 19. Be3 c5 20. Rad1 Qf6 21. Nh5 Qe7 22. Nh2 Kh7 23. Qf3 f6 24. Ng4 Bxg4 25. Qxg4 Red8 26. Qf5+ Kh8 27. f4 Rxd1 28. Rxd1 exf4 29. Bxf4 Qe6 30. Rd3 Re8 31. Nxg7 Kxg7 32. Qh5 Nh7 33. Bxh6+ Kh8 34. Qg6 Qg8 35. Bg7+ Qxg7 36. Qxe8+ Qf8 37. Qe6 Qh6 38. e5 Qc1+ 39. Kh2 Qf4+ 40. Rg3 1-0

Answer stage 1:

Analyzing the image and the text was easy understand that the text are Chess match movements.

I used the site http://chesstempo.com/pgn-viewer.html to load the movements and check the final chessboard. See below.



Answer stage 2:

Translate the chessboard to binary, where empty spaces are “0” and not empty are “1”.

```
00000001
00100001
01001100
10101000
00100100
01000011
10000011
00000000
```

Answer Stage 2 part2:

Convert from binary to decimal using http://www.asciitohex.com/

Password is: 1-33-76-168-36-67-131-0

Challenge 10

Blueprint

Time for some math! Find the number which produces the plot on the bottom!

Try these two samples: sample 1, sample 2.

Answer:

Sample 1 =

2228207

Sample 2 =

607016736537520995594815505941151025266085572371665528200038731931478673846727319684910259261305939019800769557502357959237333126657517770919783070679372916294368356299408372540498306857438421105135681283249346869449792967320249238095120706670142921097242873599571883821374208364354014530087192947246803225868116625497030891935667180689024971315282228337059319954976085961871005910163343474177499795224476259823484556802260391642987108868423291931973521858141803706752591127589236027411683906358835596869641180286211179357749072326208778420088

Answer:

Buddy, I have to congratulate who ones create this challenge. I spent hours searching up to reach the final solution.

Searching on Google, I reach the following sites:

http://www.petervis.com/mathematics/tuppers_self-referential_formula/Plot_Tuppers_Self-Referential_Formula.html

http://www.petervis.com/mathematics/tuppers_self-referential_formula/how_to_graph_tuppers_self_referential_formula.html

Finally the site that help me to plot the image and get the code. I have plotted the image by hand until get the right number.

http://tuppers-formula.tk/

Password is:

176579492015814901528872625529774615508215478614638628392406643239116428074897541681641793325671248874580950499668382723958388333354648 5

```
32262931698930639856835422348683939828636055448533804591049653503261
37397441646486218169598347856207906783361422905911386919743769975974
23736740030288615354760270915522436168657354576976561054444295062385
84383051262100293283222118456901855469818763894181110080508013645884
49772605640341039292322155464883254208546726202316901388360683605114
28818496286445011029605684824940457834254542384972955648O
```

http://hackyeaster.hacking-lab.com/hackyeaster/challenge10_7fe2ea3751694cb6653d6bd3e7f39d089da29dc7.html

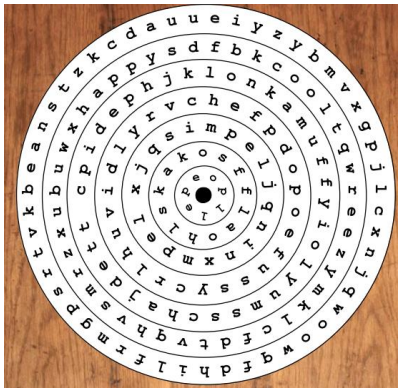
Challenge 11

Twisted Disc

You found a secret disc which conceals a secret password. Can you crack it?

Hint: Each ring of the disc holds one letter. The first letter sits on the outermost ring.!

File at: http://hackyeaster.hacking-lab.com/hackyeaster/files/disc.pdf:



Answer:

First I saved the PDF to TXT to get all discs characters.

After I wrote a small script in python to order the letter for each disc. The script is on Appendix section.

The script output is:

Whell 1 : aabbccddeeffgghiijjkkllmmnnooppqqrrssttuuvvwwxxyyzz
Whell 2 : abbccddeeffhhkkllmmooppqqrrssttuuvvwwxxyyzz
Whell 3 : aaccddeeffhhiijjkkllmmnooppssttuuyy
Whell 4 : ccddeeffhhilloopprrssuuvvyy
Whell 5 : eeiijjllmmnnppqqsxx
Whell 6 : aaffhkkllooss
Whell 7 : eellopp

There is one letter that repeat once in each disc:

h
a
n
i
s
h
o

The password is: hanisho

Challenge 12

Version Out Of Control

Version control is a powerful tool. Thinking she was oh so smart, Fluffy used it to hide an easter egg. Can you pull out the egg from her file?

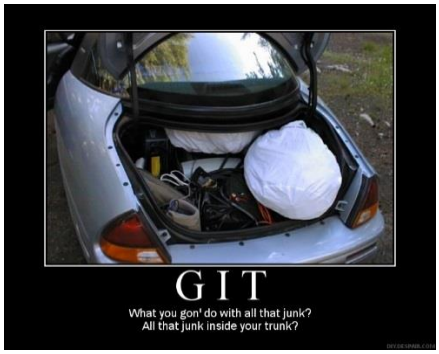
Fileat: http://media.hacking-lab.com/hackyeaster/he2016/files/1000.zip:

Hint: If you get stuck, go one step back.

Answer stage1:

I started unzipping the file 1000.zip and realized that the challenge is based on git repository. After I create a script to clone the repository in a recursive way. The script is on Appendix.

When I reached the file 0722.zip the script stopped and I got the file trunk.jpg



To move ahead O follow the Hint “If you get stuck, go one step back.”

I have checked the repository 0724 with git command an realized that have three repositories, and I need to move to the old one.

#git log –oneline
dcf4797 Change committed
44dc751 Change committed
93d6302 Commit committed

#git checkout 44dc751

Gave me 722.zip with follow content.

#git checkout 93d6302
Previous HEAD position was 44dc751... Change committed
HEAD is now at 93d6302... Commit committed

Now I got the right 722.zip and back to my script to unzip recursively.

Answer stage2:

Running git.py starting with 722.zip file. My script stopped on 396.zip file. On the zip file the content was the picture below.

Checking the repository.
#git log –oneline
a0c4344 Commit committed
27dbde6 Commit committed

#git checkout 27dbde6

Now I got the correct 396.zip and back to my script to unzip recursively.

Answer stage3:

Running git.py starting with 396.zip file. My script stopped on 0045.zip file. The 0045.zip file is password protected. Checking with git command, I have figured out the password “fluffy99”.

#git log –oneline
37f69df Commit committed. Pass is fluffy99

#unzip 0045.zip

Answer stage4:

Running git.py starting with new 0045.zip file extracted all file up to 1.zip with egg12.png.

Challenge 13

Fractal Fumbling

Do you need a new wallpaper? What about a fancy fractal?

Find the password hidden in the wallpaper image, and enter it in the Egg-O-Matic below.

File at: http://media.hacking-lab.com/hackyeaster/he2016/files/wallpaper.jpg:

Answer:

First I sliced the wallpaper.jpg in 441 new images. After I removed the blank ones resting 230 chunks.

Step 2: Slicing the 230 chunks again using my slicer.py script. After I removed the blank ones resting ~91k QRCodes to read.

After I have sliced was impossible to read the QRCodes due the size. My solution was written a script to resize the 91k QRCides. The script is on Appendix.

Final step I wrote a small python script to read the QRCodes and save in a list.

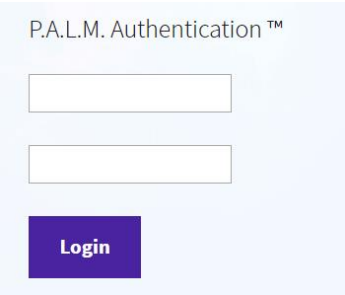
On slice_07_19-dir-slice_07_19.png is the password: fractalsaresokewl

Challenge 14

P.A.L.M.

Folks at HOBO Authentication Systems implemented a new authentication system named P.A.L.M. ™

Prove that you can break it and find a pair of username and passcode to log on.



Answer stage1:

I check the source code of page to understand what the “Authentication” works.
After a preliminary analysis, I understood that the core was ciphered in base64.

```
<script>addFooter();
eval(atob("ZXZhbChmdW5jdGlvbihwLGEsYyxrLGUsZCl7ZT1mdW5jdGlvbihjKXtyZ
XR1cm4gYy50b1N0cmluZygzNil9O2lmKCEnJy5yZXBsYWNlKC9eLyxTdHJpbmcpKXt3a
GlsZShjLS0pe2RbYy50b1N0cmluZyhhKV09a1tjXXx8Yy50b1N0cmluZyhhKX1rPVtmd
W5jdGlvbihlKXtyZXR1cm4gZFtlXX1dO2U9ZnVuY3Rpb24oKXtyZXR1cm4nXFx3Kyd9O
2M9MX07d2hpbGUoYy0tKXtpZihrW2NdKXtwPXAucmVwbGFjZShuZXcgUmVnRXhwKCdcX
GInK2UoYykrJ1xcYicsJ2cnKSxrW2NdKX19cmV0dXJuIHB9KCdyIHEoKXsyIHU9Ny44K
Fwnd1wnKS5iOzIgcD03LjgoXCd2XCcpLmI7MiA0PVswLDAsMCwwLDAsMCwwLDAsMCwwX
TsyIDU9YzszKHU9PT1cJ2dcJyl7MyhwPjAmJnAuaD09OSl7NT1qO2YoaT0xO2k8PTk7a
SsrKXsyIDY9cC5rKGktMSk7MiBhPXAuZSgwLGkpOzMoNFs2XSE9MHx8YSVpIT0wKXs1P
WN9Myg0WzZdPT0wKXs0WzZdPTF9fX19Myg1KXs3Lm0ubj1cJ29cJyt1K1wneFwnK3ArX
Ccuc1wnfXR7ZChcJ2xcJyl9fScsMzQsMzQsJ3x8dmFyfGlmfHVzZWR8b2t8ZGlnaXR8Z
G9jdW1lbnR8Z2V0RWxlbWVudEJ5SWR8MTB8cGFydHx2YWx1ZXxmYWxzZXxhbGVydHxzd
WJzdHJpbmd8Zm9yfGVsc2F8bGVuZ3RofHx0cnVlfGNoYXJBdHxub3BlMXxsb2NhdGlvb
nxocmVmfGNoYWxsZW5nZTE0X3x8Y2hlY2tFbnRyaWVzfGZ1bmN0aW9ufGh0bWx8ZWxzZ
Xx8cHBhc3N8cHVzZXJ8Xycuc3BsaXQoJ3wnKSwwLHt9KSk="));</script>
```

```
Decoding base64.
eval(function(p,a,c,k,e,d){e=function(c){return
c.toString(36)};if(!''.replace(/^/,String)){while(c--
){d[c.toString(a)]=k[c]||c.toString(a)}k=[function(e){return
d[e]}];e=function(){return'\\w+'};c=1};while(c--
){if(k[c]){p=p.replace(new
RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('r q(){2
u=7.8(\'w\').b;2 p=7.8(\'v\').b;2 4=[0,0,0,0,0,0,0,0,0,0];2
5=c;3(u===\'g\'){3(p>0&&p.h==9){5=j;f(i=1;i<=9;i++){2 6=p.k(i-1);2
a=p.e(0,i);3(4[6]!=0||a%i!=0){5=c}3(4[6]==0){4[6]=1}}}}3(5){7.m.n=\'
o\'+u+\'x\'+p+\'.s\'}t{d(\'l\')}}',34,34,'||var|if|used|ok|digit|doc
ument|getElementById|10|part|value|false|alert|substring|for|elsa|le
ngth||true|charAt|nope1|location|href|challenge14_||checkEntries|fun
ction|html|else||ppass|puser|_'.split('|'),0,{}))
```

Answer stage2:

Checking the code was easy to understand that the JavaScript was packed.
I unpacked using the site http://matthewfl.com/unPacker.html

```
function checkEntries()
	{
	var u=document.getElementById('puser').value;
	var p=document.getElementById('ppass').value;
	var used=[0,0,0,0,0,0,0,0,0,0];
	var ok=false;
	if(u==='elsa')
		{
		if(p>0&&p.length==10)
			{
			ok=true;
			for(i=1;
			i<=10;
			i++)
				{
				var digit=p.charAt(i-1);
				var part=p.substring(0,i);
				if(used[digit]!=0||part%i!=0)
					{
					ok=false
				}
				if(used[digit]==0)
					{
					used[digit]=1
				}
			}
		}
	}
	if(ok)
		{
		document.location.href='challenge14_'+u+'_'+p+'.html'
	}
	else
		{
		alert('nope1')
	}
}
```

Answer stage3:

Ok, now I know that the user is “elsa” but was needed to crack the password. I wrote a script to the crack and retrieve the password. The script is on Appendix.

The script output is:
FOUND!!! 3816547290
Duration: 22:58:23.319806 (my poor computer spent long time working to grab the password ☹)

The password is: 3816547290

Challenge 15

Big Bad Wolf

Three little pigs have hidden in their house. You're the big, bad wolf, and your stomach is growling. Huff and puff and blow the pigs' house in! Get that juicy bacon!

Hints:
the pigs have hidden in three different media types (image, sound, text)
no password cracking is necessary
File at: http://media.hacking-lab.com/hackyeaster/he2016/files/disk.img

Answer :

Followint the hints...

First I have extracted the files in the disk image.
#mount -o loop disk.img /mnt/
#ls –l /mnt

```
drwx------ 2 root root   12288 Nov 29 01:51 lost+found
-rwxr-x--- 1 root root   93810 Nov 29 01:51 piglet.jpg
-rwxr-x--- 1 root root   95865 Nov 29 01:51 pigs.jpg
-rwxr-x--- 1 root root 2654809 Nov 29 01:51 song.mp3
-rwxr-x--- 1 root root   73322 Nov 29 01:51 story.pdf
-rwxr-x--- 1 root root    1620 Nov 29 01:51 story.txt
-rwxr-x--- 1 root root  190646 Nov 29 01:51 wolf.jpg
```

Pig one was hidden in story.txt file using Snow. http://www.darkside.com.au/snow/. Running snow command without password showed the pig 1 name. BTW, I spend long time to solve Pig one name ☺

c:\SNOW.EXE story.txt
pig 1: Filbert

Pig two was hidden in wolf.jpg file. That was the easy one.
#xxd wolf.jpg

0001070: 0000 0000 0000 0000 0000 0000 7000 6900 ....p.i.
0001080: 6700 2000 3200 3a00 2000 4300 6100 7300 g. .2.:. .C.a.s.
0001090: 7300 6100 6400 6500 6500 0000 ffe1 08dd s.a.d.e.e.....

Pig 2 name is: Cassadee

Pig three was hidden in song.mp3 file. Using Mp3stego program the pig 3 name was reveled.
http://www.petitcolas.net/steganography/mp3stego/

C:\mp3stego.exe song,mp3
pig 3: Wynchell

Challenge 16

Egg Coloring

Egg coloring is fun!

Can you get the yellow egg?

Answer:

Was clear that using the app was impossible to get the Yellow egg.
To solve this challenge I reversed the HE app and got the ColorActivity.class source code to analyze.

Chunks code from ColorActivity.class source code.

```
String s = (new StringBuilder()).append("http://hackyeaster.hacking-
lab.com/hackyeaster/egg?code=").append(a[i]).append("&key=").append(
"eggsited").append("&hmac=").append(b[i]).toString();


ArrayAdapter arrayadapter = new ArrayAdapter(this, 0x1090008, new
String[] {"Red", "Green", "Blue", "Cyan", "Magenta", "White",
"Black"});


private static String a[] = {"ff0000", "00ff00", "0000ff", "00ffff",
"ff00ff", "ffffff", "000000" };


private static String b[] = {
"f4e075524ba4470867e1891c1a8d1fc21df1f56a",
"b23f66454417de5be448da84a846989b42f304c8",
"f5ecdd0f12749fe75734b42bf29943d28acf4573",
"2cf2a7cd695a462adcbc324df9302003a99c688a",
"543e3853ac9318587c10c7645b6828e2a858ecf5",
"c46ffadf392698e28fdeb344239130e2ade2c809",
"7b06466eb80d88533a2d1c7b9de62d98c4e20d1d"
};
```

The solution is recreate the URL to grab the Yellow egg. To do that I need the color Yellow in hex the key and the correct Hmac. I generated the Hmac using following Hmac online generator:
http://www.freeformatter.com/hmac-generator.html

Color = ffff00
key = eggsite
hmac=1da02c68080863fa302c20c3312371f4e365a5f9

The final URL is:

http://hackyeaster.hacking-lab.com/hackyeaster/egg?code=ffff00&key=eggsited&hmac=1da02c68080863fa302c20c3312371f4e365a5f9

I got the egg in base64 format.

iVBORw0KGgoAAAANSUhEUgAAAeAAAAHgCAYAAAB91L6VAAAABGdBTUEAALGPC/xhBQAAACBjSFJ
…
… mdHdhcmUAcGFpbnQubmV0IDQuMC42/Ixj3wAAAABJRU5ErkJggg==

Decoding the base64.

#base64 –d egg.b64 > egg16.png

Challenge 17

Bunny Hop

Wannabe programming guru Hazel B. Easterwood created a new programming language called "Bunny Hop". You suspect Hazel to have cheated, because the language looks very familiar to you.

Download the following code and complete it! It will yield the QR code for egg 17.

File at: http://hackyeaster.hacking-lab.com/hackyeaster/files/egg17.bunny:

This challenges was provided by volunteer inik. Do you feel like providing a challenge, too? Let us know!

Answer:

After some search in Google, I have figured out that the language used was Logo. To solve the challenge I have replaced the Bunny Hop commands by Logo commands and add a function named “egg” to draw the QR Code squares. The fixed code is on Appendix section.

Egg Function:

```
to egg
   PD
   setpensize 9
   filled 'black [repeat 4[FD 1 RT 90] ]
   PU RT 45 FD 4 PD
   PU BK 4 PD
   LT 45
   PU
end
```

I used the online Logo plotter in http://www.calormen.com/jslogo/ to plot the QR Code.

Challenge 18

Bug Hunter

Lacking of time, you were not able not complete your DeggCryptor program. In an act of desperation, you instructed Sammy, the junior programmer, to implement the missing key generation function.

As always, Sammy miserably failed. Can you fix his code? It's the KeyGen class. Pay attention to the comments!

File at: http://hackyeaster.hacking-lab.com/hackyeaster/files/sourcecode.zip:

Answer:

After analyze the source code I realized that I needed to fix the DeggCryptor function. The "Sammy" comments help me a lot. Below is the function source code and in red the comments of what I have fixed on the source code.
Then I have recompiled the code and grab the egg.

```
//
//  KeyGen
//  Implemented by the one and only Sammynator!
//
namespace DeggCryptor
{
    class KeyGen
    {
        public static string generateKey()
        {
            // Init the four seed values. 1111 and multiples of it.
            int h1 = 1111; //Fixed values
            int h2 = 2222; //Fixed values
            int h3 = 3333; //Fixed values
            int h4 = 4444; //Fixed values

            // Init variables.
            int a = h1;
```

```
            int b = h2;
            int c = h3;
            int d = h4;
            int e = 0;

            // 1'000 iterations.

            for (int i = 1; i <= 1000; i++) //Fixed loop
            {
                // If c is greater than d, double c and d.
                if (c > d) //fixed If statement
                {
                    c *= 2;
                    d *= 2;
                }
                // Calculate new values.
                // a: Take sum of a and b, and c and d. Then, multiply the two values.

                a = (a + b) * (c + d); //Fixed using ()
                // b: multiply with 3. Using two additions instead of multiplying ->
performance booooost!
                b *= 3; //Fixied mulplication

                // c, d: Swap c and d
                e = c; //Fixed Swap
                c = d;
                d = e;
                // Take last four digits (modulo 10'000),

                a %= 10000;
                b %= 10000;
                c %= 10000;
                d %= 10000; //Fixed Modulo
            }

            // Password: a,b,c and d with "X" in between
            string password = a + "X" + b + "X" + c + "X" + d;
            return password;
            }

    }
}
```

DeKrypt!

Challenge 19

Assemble This!

In this challenge, you must crack a server-side program. Lucky for you, you got the assembly file of the program. First, reverse-engineer the program and find a valid code! Then, submit the code to the server.

File at: http://hackyeaster.hacking-lab.com/hackyeaster/files/assembly.txt:

The server is located at:

hackyeaster.hacking-lab.com:1234

Important: Do not launch brute-force attacks on the server - you'll not be lucky with it!

Answer:

First I have compiled the code to be debugged.

```
# gcc -c egg19.S -o egg19.o && gcc egg19.o -o egg19
```

Using IDA and gdb for debugg I have developed an exploit to send the correct caracthers and grab the password. The script in python is on Appendix section.

Password is : ida.lo\/es.you

Challenge 20

Humpty's Dump

You got hold of a dump of Humpty Dumpty's secret egg database.

Search and extract the egg hidden in the dump!

Hints

The 'puzz' is not more than 8 chars, letters only.

The decryption of the file can be done using AES_DECRYPT().

File at: http://hackyeaster.hacking-lab.com/hackyeaster/files/dump2.zip:

Answer Stage 1:

My initial approach to solve challenge 20 was to create the MySQL database "humpty" and load the dump data. After I star to analyze "humpty_routines.sql" file.

Special attention to GetPuzzMishMash procedure. I guess that the way to solve the puzzle is cracking the password on "uzr" table to decrypt the egg stored on fyle table.

```
+------+--------+------------------------------------------+-------+
| id   | neighm | puzz                                     | sawlt |
+------+--------+------------------------------------------+-------+
| 1342 | stuart | de2278f5bcafcbb097ecc1fb54e5ab8a9e912c55 | efgh  |
| 1875 | beaver | 943f9ecbbd91306a561d0e3c15e18ee700007083 | abcd  |
| 3443 | chuck  | 0cf32f8f418659f23f8968d4f63ea5c98b39f833 | zyxw  |
| 5420 | yogo   | 1742ae4507fc480958e2437104e677e70aa5e857 | jklm  |
| 8944 | flint  | 915d253cb5ba6f0a220bca83e2d6d3258af15e68 | nmlk  |
+------+--------+------------------------------------------+-------+
```

I have used hashcat to bruteforce the passwords(-a3) using algorithm SHA1 with salt. (-m 4900) and a mask file as well(following the hint "The 'puzz' is not more than 8 chars, letters only."). It is on Appendix section.

The content of hash.txt file:

```
43f9ecbbd91306a561d0e3c15e18ee700007083:abcd
915d253cb5ba6f0a220bca83e2d6d3258af15e68:nmlk
1742ae4507fc480958e2437104e677e70aa5e857:jklm
0cf32f8f418659f23f8968d4f63ea5c98b39f833:zyxw
de2278f5bcafcbb097ecc1fb54e5ab8a9e912c55:efgh
```

#hashcat -a3 -m 4900 hash.txt -1 ?l mask.txt

```
----------------------------------------------------------
0cf32f8f418659f23f8968d4f63ea5c98b39f833:zyxw:.snakeoil.
[s]tatus [p]ause [r]esume [b]ypass [q]uit =>
Input.Mode: Mask (.?1?1?1?1?1?1?1?1.) [10] (100.00%)
Index.....: 0/1 (segment), 208827064576 (words), 0 (bytes)
Recovered.: 1/4 hashes, 1/4 salts
Speed/sec.: 1.70M plains, 568.29k words
Progress..: 208827064576/208827064576 (100.00%)
Running...: 04:06:04:26
```

Ok we get the password of hash 0cf32f8f418659f23f8968d4f63ea5c98b39f833 with salt zyxw, user chuck. The password is: snakeoil

Answer Stage 2:

The next step was decrypt the egg stored in fyle table. I followed the second hint "The decryption of the file can be done using AES_DECRYPT()."

```
mysql> use humpty
Database changed
#Geting the kee value
mysql> SELECT kee FROM kee WHERE id=2332 AND uzrid=3443;
+----------------------------------------------------------------+
| kee                                                            |
+----------------------------------------------------------------+
| 1ABF4B7CD25C61FDF0E74EC2BFB43BD1C2D8ECD803AFA3AA376F4C0000052813 |
+----------------------------------------------------------------+


#Geting the p_kee value
mysql> CALL DeekryptKee("snakeoil",
"1ABF4B7CD25C61FDF0E74EC2BFB43BD1C2D8ECD803AFA3AA376F4C0000052813", @p_kee);
Query OK, 1 row affected (0.07 sec)
```

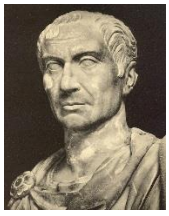
```
mysql> SELECT @p_kee;
+----------------------+
| @p_kee               |
+----------------------+
| jpP8HeoEC5OCCBqdf9N3 |
+----------------------+

#Finally decrypting the egg and dumping the content to file
mysql> SELECT AES_DECRYPT(blahb, "jpP8HeoEC5OCCBqdf9N3") FROM fyle WHERE id=3492 and
keeid=2332 INTO DUMPFILE '/tmp/egg20.png';
```

I did not have spare time to write a script to automatize this, but it is feasible, maybe if I have time in future I do that. ☺

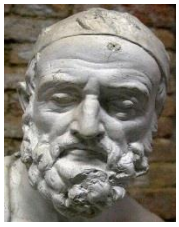
Challenge 21

Crypto Council

Clever crypto brains had a little get-together. Find out who they are and break the riddles they created! Each plain text contains a password - once you've got them all, enter them in the Egg-O-Matic below.Lowercase only.



DV D UXOH PHQ ZRUUB PRUH DERXW ZKDW WKHB FDQW VHH WKDQ DERXW ZKDW WKHB FDQ SDVVZRUG LV FDUWKDJR



4423154215 2443 3334 52244433154343 4334 1442151114214531 3334 11131345431542 4334 4415424224123115 1143 442315 13343343132415331315 44231144 145215313143 2433 442315

2315114244 3421 1551154254 321133 35153134353433331543 15 2443 442315 3511434352344214

EHIIKT YFC FTEU QK PLTPWBY MQYTNVZW LAJ JGGN ZVLD A EWTAE WIEXLP QF IHV DAALROW DF JIACT GWMGCRQF WIJ NSIHVZ BTAE IJGAEOWS FFZ ZXM KW ZPVV I UAAJAARAC MVJCRBADN ZV HPRZA TAAZAW SE MQYTNVZW HTLLATD XZWTK RVV WESZWL UELWG AUZAPNLA LJREMTJS RVV YERV VDRRB SI TYM SVE FN SVE JMNTNKMWC HV MFIEIMV IHV LAELFUSIIT AWGVZKW PNU ZWBAZVWS TYMJT FFZ LWIIBQ NERZK UIMM QTAIA ACTF PAH CRZWTR YM SRCFUHPNZMV IHV NJTNTP WCVFG DDUZA SSHVUSG DV OJXGEIF IO KPW SIVB GU WFZEH AJ I BJNZWJ HETZWIAIG ZT EEBWGEU BZT SVZNXCV WX IHV LMZE FN FTVVZK PS YQK HETZWIAIG S EOJQLXOE PW WECL MCTZT LWE UMSIHJ WX IHV LMZE RVV WIJ AGC HV IDHO JMJKEU IK P SVKJTTRZQ IO YMFGY ZQA

WEN XQWVIBQZ KGQEAL TWB WEH GKQCW QLTBAKBTU LKIQTME DWCOAKWZAKNB BKMETP WEW NOTHPA HWB GBXHCEWG IA OTTQPWD SEATWCWNBA NZW HTO BHQWG HIWAL MCMG LQWVIBQL TOF QLFVCHWG WEW XNW FM WET NGQEAL QWBDSCMF KO CPWCKWMAX

Answer:

Straightforward challenge.
After a quick Image search on Google, I figured out the names of the ciphers creators.

The decrypted messages are:

Cesar

Rot 23
AS A RULE MEN WORRY MORE ABOUT WHAT THEY CANT SEE THAN ABOUT WHAT THEY CAN
PASSWORD IS CARTHAGO

Polybius Cipher

THERE IS NO WITNESS SO DREADFUL NO ACCUSER SO TERRIBLE AS THE CONSCIENCE THAT DWELLS IN THE HEART OF EVERY MAN PELOPONNESE IS THE PASSWORD

```
ABCDE
FGHIK
LMNOP
```

```
QRSTU
VWXYZ
```

Vignere

Key Paris

phrase you need is alchemy vigenere was born in to a noble family in the village of saint pourcain his father jean arranged for him to have a classical education in paris blaisede vigenere studied greek and Hebrew under adrianu sturnebus and jean dorat at the age of age seventeen he entered the diplomatic service and remained there for thirty years five years in to his career he accompanied the French envoy louis adhemar de grignan to the diet of worms as a junior secretary he entered the service of the duke of nevers as his secretary a position he held until the deaths of the duke and his son he also served as a secretary to henryiii

Playfar

The playfair cipher was the first practical digraph substitution cipher the scheme was invented by Charles wheatstone but was named after lord playfair who promoted the use of the cipher password is bletchley

```
DFGIK
LMPQR
UVXYZ
WHEAT
SONBC
```

Passwords are:
carthago
peloponnese
alchemy
bletchley

Challenge 22

Dumpster Diving

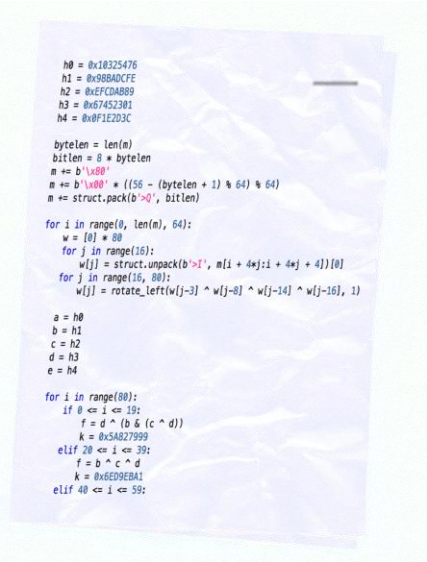
You've sniffed some password hashes of a web site:

hash 1: fad202a6e094dd8f1d63da8bdf85b3ba099971d3
hash 2: f71e1b0b9b3a57d864c2e9f7bd6dd90f66b5a7d6

hash 3: 84c6bcb681b79b690b53f9f3a8ba24e1e970d348
hash 4: 0d6bb0df8918168798ce6b770014aeb81ac6ce76

However, none of your tools succeeded in cracking the hashes. As a last resort, you inspected the dumpster of the software development company, which created the web site. And indeed you found something: a paper with a part of the hash calculation code.

Can you crack the hashes now?



```
h0 = 0x10325476
h1 = 0x98BADCFE
h2 = 0xEFCDAB89
h3 = 0x67452301
h4 = 0xC3D2E1F0

bytelen = len(m)
bitlen = 8 * bytelen
m += b'\x80'
m += b'\x00' * ((56 - (bytelen + 1) % 64) % 64)
m += struct.pack(b'>Q', bitlen)

for i in range(0, len(m), 64):
    w = [0] * 80
    for j in range(16):
        w[j] = struct.unpack(b'>I', m[i + 4*j:i + 4*j + 4])[0]
    for j in range(16, 80):
        w[j] = rotate_left(w[j-3] ^ w[j-8] ^ w[j-14] ^ w[j-16], 1)

    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    for i in range(80):
        if 0 <= i <= 19:
            f = d ^ (b & (c ^ d))
            k = 0x5A827999
        elif 20 <= i <= 39:
            f = b ^ c ^ d
            k = 0x6ED9EBA1
        elif 40 <= i <= 59:
```

Answer:

Quick search on Google showed me that the chunk code on the paper was part of SHA1 hash implementation with some minor changes on the Initial digest variables.

```
0x67452301
0xEFCDAB89
0x98BADCFE
0x10325476
0xC3D2E1F0
```

I wrote a cracker to brute force the hashes. The code is on Appendix section.
My cracker is based on SHA1 pure implementation coded by AJ Alt on https://github.com/ajalt/python-sha1 . Tks AJ .

To crack the Hashes 1 to 3 I used famous Rockyou.txt wordlist.

Hash 4 was most difficult, I spent couple hours trying different wordlists up to reach crackstation-human-only wordlist from https://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm

Below are the password in red and the time spent to crack.

HASH 1 SHA1=zombie
Duration: 0:00:02.111941

HASH 2 SHA1=Denver1
Duration: 0:01:09.258620

HASH 3 SHA1=Placebo
Duration: 0:00:21.568771

HASH 4 SHA1=SHADOWLAND
Duration: 0:15:07.694476

Challenge 23

Heizohack

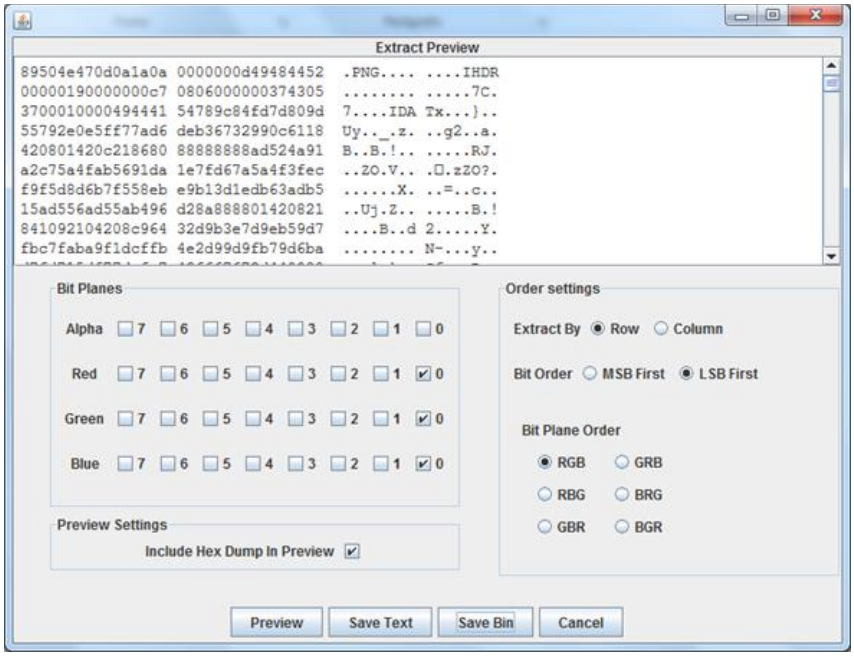
Can you crack the Heizohack?

The password for the Egg-O-Matic is hidden in the image.

Very difficult challenge, congrats to the HL team.
It is a two-stage challenge. Here is my solution.

Answer stage1:

I started checking the heizohack.bmp file with my favorite Steg tool Stegsolver. After some changes on the settings, I figured out a PNG image hidden on the heizohack.bmp.



The hidden image is

Answer stage2:

Following the hint on the image, the final message done by the group of Red LSB value attending the condition "XOR of Red, Green, Blue and Alpha LSB is equal 1".

I wrote a small python script to scan the image and grab the value of Red LSB. The output is a binary that converted to Ascii is

Binary output:
01101100011011110111001101110100011010010110111001110100011010000110010101110111
01101111011011110111011110110111101100010001110011

Password is : lostinthewoooods

Challenge 24

crunch.ly

Do you know crunch.ly, the fancy new URL shortener? It was used to create a short URL for Hacky Easter. In order to lure people onto the web site of your alternative hacking competition "Evil Easter", you decide to attack this service.

What you know

Short URL for Hacky Easter: http://crunch.ly/IU66SMI

Web site of crunch.ly (not a real domain!): OPEN WEB SITE

Algorithms used on the web site: DOWNLOAD

Your mission

- find a URL starting with http://evileaster.com, which produces the same short URL
- make the web site store your URL, instead of the original URL
- open the short URL on the web site
- do not bomb or DoS the server - you'll have no luck with it; cracking must happen offline

Answer stage1:

First of all I have analyzed the algorithm provided on crunchly.txt file.
My first approach is to crack the KEY. Based on the comment on KEY_FULL variable, I know that the KEY_FULL is 16 bytes long(128 bits) and it is the concatenation of `"x" + KEY + KEY + KEY`.

```
private static final String KEY = [[CENSORED!!!]]
         private static final String KEY_FULL = "x" + KEY + KEY + KEY; // 128 bit
         private static final String IV = "hackyeasterisfun";
```

Then KEY is 5 byte long. To crack the KEY I do the following:

1) Generate a Wordlist using crunch wordlist tool
   #crunch 5 5 1234567890abcdefghijklmnopqrstuvWxyzABCDEFGHIJKLMNOPQRSTUVWXYZ -o dictionary.txt
2) Generate a valid ticket using crunch site.
3) Create a cracker to crack KEY variable. The program is on Appendix named KEYCRACKER.java

4) Compile and Run the cracker.
   #java -classpath . KEYCRACKER
5) The output is:
   ==Java Key Cracker==
   URL: http://hackyeaster.hacking-lab.com/
   ShortURL: http://crunch.ly/GQPVN4Q
   key = tKguF

Answer stage2:

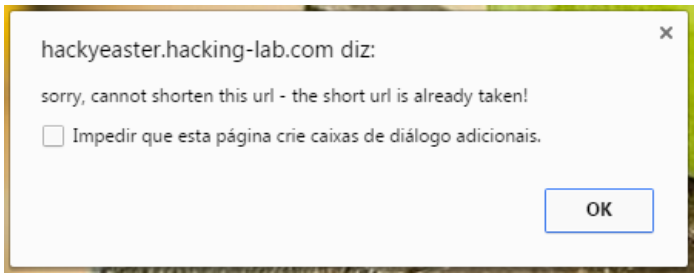
With the cracked key tKguF the next step was generate the URL http://evileaster.com that produce the same ShortURL **http://crunch.ly/IU66SMI** .

To help on this hard task I code a program named EVIL.java that is on Appendix. After 23 hours running ☹ the EVIL output was:
EUREKA, the url is :http://evileaster.com/lDdvCxa

Answer stage3:

With the URL http://evileaster.com/lDdvCxa the next step is make the web site stores the URL, instead of the original URL.

Trying to store using the Website outputs an error.



To achieve this I had to generate new ticket to "http://evileaster.com/lDdvCxa" URL. I did using a script named TICKET.java. The script is on Appendix.

The TICKET output is:
==Java Ticket Creator==
Evil URL is : http://evileaster.com/lDdvCxa
Short URL=http://crunch.ly/IU66SMI
Valid Ticket is:
Hgo3UsPWbH+4kkfQwZ0dOFDpPjOhGDNF5HxHfiTzl9wguHyNPHIGoJzdWW8B+Nr/H70MFUp1pkwA
pljLmy9SWJhlwZyeEhoKiQHHMgBJ2Ak=

After, using Fiddler Web Debugger I did a fake request to crunch site replacing the ticket by the Ticket generated to "http://evileaster.com/lDdvCxa"
(Hgo3UsPWbH+4kkfQwZ0dOFDpPjOhGDNF5HxHfiTzl9wguHyNPHIGoJzdWW8B+Nr/H70MFUp1pkwA
pljLmy9SWJhlwZyeEhoKiQHHMgBJ2Ak=) forcing the website to store my ShortURL.

Answer final

Now opening the short URL IU66SMI on the web site sent me to the URL http://hackyeaster.hacking-lab.com/hackyeaster/images/egg24_bHIrQh1VR141TPmapETM.png

My deep thanks to all HL team for one more amazing HackyEaster challenge. This is my second year solving the HE challenges and I am keeping evolving my knowledge year by year with yours challenges.

Thank you very Much!!!

Ed SH

APPENDIX – SOURCE CODE

Challenge 11

#Sort by eash#

```
w1 = 'beanstzkcdauueiyzybmvxgpjlcxnjqwoowqfdhilfrmgpsrtvk'
w2 = 'buwxhappysdfbkcooltqwreezymklcfdtvqhvsmrzxu'
w3 = 'cpidephjklonkamuffyiolyumsschajdett'
w4 = 'idlyrvchefpdopoefussycrlhuv'
w5 = 'xjqsimpeljqninxmpel'
w6 = 'kakosfflaohls'
w7 = 'peoplle'

x1 = []
x2 = []
x3 = []
x4 = []
x5 = []
x6 = []
x7 = []

for l in w1:
    x1.append(l)
for l in w2:
    x2.append(l)
for l in w3:
    x3.append(l)
for l in w4:
    x4.append(l)
for l in w5:
    x5.append(l)
for l in w6:
    x6.append(l)
for l in w7:
    x7.append(l)

x1.sort()
x2.sort()
x3.sort()
x4.sort()
x5.sort()
x6.sort()
x7.sort()


print "Whell 1 : ", ''.join(x1)
print "Whell 2 : ", ''.join(x2)
print "Whell 3 : ", ''.join(x3)
print "Whell 4 : ", ''.join(x4)
print "Whell 5 : ", ''.join(x5)
print "Whell 6 : ", ''.join(x6)
print "Whell 7 : ", ''.join(x7)
```

Challenge 12

#Coded by eash#

```
from dulwich.repo import Repo
import zipfile
import shutil
from shutil import copyfile
import subprocess
import os.path
from string import rstrip
```

```
import itertools
import string
import re
import os



for i in range(1000, 0, -1):
   for fn in os.listdir("./saida"):
      with zipfile.ZipFile("./saida/" + fn, "r") as z:
         z.extractall("./output")
         check = "./output/"+fn
         file_path = os.path.splitext(check)[0]
         #print file
         num = str(i)
         Repo(file_path + "/.git").clone("./gits/output" + num + "/")
         x = os.listdir("./gits/output" + num + "/")
         copyfile("./gits/output" + num + "/" + x[1],"./saida/" + x[1])
         os.remove("./gits/output" + num + "/" + x[1])
         os.remove("./saida/" + fn)
         dirname = "output/*"
         dirname1 = "./output" + num + "/"
         os.system("rm -rf %s" % dirname)
```

Challenge 13

```
#Slicer#
#Coded by eash#
import image_slicer
import qrtools
import os
#tiles = image_slicer.slice('wallpaper.jpg', 441,save=False)

qr = qrtools.QR()

for fn in  os.listdir("./wallpaper_slices"):
    file = os.path.splitext(fn)[0]
    file1 = "./wallpaper_slices/" + file + '-dir'
    file2 = "./wallpaper_slices/" + fn
    os.mkdir(file1, 0755 );
    tiles = image_slicer.slice(file2, 441, save=False)
    image_slicer.save_tiles(tiles, directory=file1, prefix='slice')



#Resize QrCodes
#Coded by eash#
import PIL
from PIL import Image
import os
count = 0

for fn in  os.listdir("wallpaper_slices"):
       for fx in  os.listdir("wallpaper_slices/"+fn):
              count = count + 1
              print count
              #print fn
              basewidth = 100
              img = Image.open('wallpaper_slices/' + fn + '/' + fx)
              wpercent = (basewidth/float(img.size[0]))
              hsize = int((float(img.size[1])*float(wpercent)))
              img = img.resize((basewidth,hsize), PIL.Image.ANTIALIAS)
              #print fx
              img.save('output/' + fn + "-"+ fx , quality=100)
```

```
#QRCode Reader#
#coded by eash
import pyqrcode
import qrtools
import os
from PIL import Image

qr = qrtools.QR()
output = []
f = open('list.txt','w')

for fn in  os.listdir("output"):
         qr.decode("output/" + fn )
         f.write(fn + "=" + qr.data + "/n")
f.close()
```

Challenge 14

#Coded by eash#

```
import itertools
from datetime import datetime

value = []

start_time = datetime.now()
print('Duration: {}'.format(start_time))

for t in itertools.product('9876543210', repeat=10):
    value = ''.join(t)
    used = [0,0,0,0,0,0,0,0,0,0]
    ok = 0
    for x in range(1, 11):
        y = x - 1
        digit =  int(value[y])
        part  =  int(value[0:x])
        if (used[digit] != 0) | (part%x != 0): #& len(str(part)) == 10:
            ok = 1
        if used[digit] ==  0 :
            used[digit] = 1
        y = 0
        #print ok
    if(ok == 0 ): #& len(str(part)) == 10:
        print "FOUND!!!" + str(value)
        end_time = datetime.now()
        print('Duration: {}'.format(end_time - start_time))
        exit()
    print value
```

Challenge 17

#Coded by eash#

```
clearscreen window hideturtle
window
to lineofeggs :cnt
  repeat :cnt [egg fd 10 ]
  bk 10
end
to egg
   PD
   setpensize 9
   setcolor pick [ red orange yellow green blue violet ]
   filled 'black [repeat 4[FD 1 RT 90] ]
   PU RT 45 FD 4 PD
   PU BK 4 PD
   LT 45
   PU
end
lineofeggs 7
```

```
fd 20
lineofeggs 6
fd 30
rt 90
lineofeggs 5
rt 180
fd 40
rt 90
fd 20
lineofeggs 7
rt 90
fd 10
rt 90
fd 240
lt 90
lineofeggs 6
rt 180
fd 50
rt 90
fd 60
rt 90
lineofeggs 6
rt 180
fd 50
rt 90
fd 60
lineofeggs 5
fd 20
rt 90
lineofeggs 6
rt 180
fd 50
rt 90
fd 60
rt 90
lineofeggs 6
rt 180
fd 40
lt 90
fd 220
rt 180
lineofeggs 3
fd 60
egg
fd 20
egg
fd 20
egg
fd 60
lineofeggs 3
rt 90
fd 10
rt 90
fd 200
rt 180
lineofeggs 3
fd 50
egg
fd 20
egg
fd 20
egg
fd 20
rt 90
lineofeggs 3
rt 180
fd 20
rt 90
fd 50
lineofeggs 3
rt 90
fd 10
rt 90
fd 200
rt 180
lineofeggs 3
fd 40
```

```
egg
fd 20
egg
fd 100
lineofeggs 3
rt 90
fd 10
rt 90
fd 90
rt 180
egg
rt 90
fd 10
rt 90
fd 120
rt 180
lineofeggs 6
fd 20
egg
fd 20
egg
fd 20
egg
fd 20
egg
fd 20
rt 90
lineofeggs 5
rt 180
fd 40
rt 90
fd 30
lineofeggs 6
rt 90
fd 10
rt 90
fd 150
lt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 20
egg
fd 20
rt 90
lineofeggs 5
rt 180
fd 30
lt 90
fd 130
rt 180
lineofeggs 5
fd 20
lineofeggs 2
fd 50
lineofeggs 3
fd 40
lineofeggs 2
fd 20
rt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 20
rt 90
lineofeggs 2
rt 90
fd 220
rt 180
egg
fd 50
egg
fd 50
lineofeggs 6
fd 60
```

```
lineofeggs 3
rt 90
fd 10
rt 90
fd 220
rt 180
egg
fd 20
lineofeggs 2
fd 40
lineofeggs 3
fd 70
lineofeggs 3
fd 20
rt 90
lineofeggs 2
rt 90
fd 180
rt 180
lineofeggs 3
fd 60
lineofeggs 3
fd 70
lineofeggs 3
rt 90
fd 10
rt 90
fd 230
rt 180
lineofeggs 2
fd 20
rt 90
lineofeggs 7
rt 180
fd 60
rt 90
fd 30
egg
fd 20
lineofeggs 2
fd 20
rt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 30
egg
fd 20
lineofeggs 3
fd 20
rt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 30
rt 90
lineofeggs 5
rt 180
fd 30
lt 90
fd 220
rt 180
lineofeggs 3
fd 60
lineofeggs 3
fd 40
egg
fd 40
rt 90
lineofeggs 4
rt 180
fd 30
rt 90
fd 20
rt 90
```

```
lineofeggs 3
rt 180
fd 10
lt 90
fd 210
lt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 20
lineofeggs 2
fd 20
egg
fd 30
egg
fd 20
rt 90
lineofeggs 3
rt 180
fd 20
rt 90
fd 30
lineofeggs 2
fd 20
lineofeggs 4
fd 30
lineofeggs 2
rt 90
fd 10
rt 90
fd 190
lt 90
lineofeggs 6
rt 180
fd 50
rt 90
fd 50
lineofeggs 2
fd 30
lineofeggs 3
fd 20
rt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 30
lineofeggs 5
rt 90
fd 10
rt 90
fd 230
rt 180
egg
fd 40
lineofeggs 4
fd 30
rt 90
lineofeggs 4
rt 180
fd 30
rt 90
fd 10
rt 90
lineofeggs 4
rt 180
fd 30
rt 90
fd 30
egg
fd 30
lineofeggs 3
fd 40
egg
rt 90
fd 10
```

```
rt 90
fd 240
rt 180
egg
fd 80
rt 90
lineofeggs 7
rt 180
fd 60
rt 90
fd 10
rt 90
lineofeggs 4
rt 180
fd 30
rt 90
fd 40
rt 90
lineofeggs 3
rt 180
fd 20
rt 90
fd 30
egg
rt 90
fd 10
rt 90
fd 150
lt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 50
egg
fd 40
lineofeggs 6
fd 30
lineofeggs 7
rt 90
fd 10
rt 90
fd 220
rt 180
egg
fd 80
lineofeggs 4
fd 20
lineofeggs 2
fd 20
rt 90
lineofeggs 6
rt 180
fd 50
rt 90
fd 60
rt 90
lineofeggs 6
rt 180
fd 40
lt 90
fd 210
rt 180
lineofeggs 8
fd 100
lineofeggs 3
rt 90
fd 10
rt 90
fd 220
rt 180
egg
fd 20
egg
fd 50
lineofeggs 2
fd 30
```

```
egg
fd 20
egg
fd 20
lineofeggs 2
fd 40
lineofeggs 3
rt 90
fd 10
rt 90
fd 180
lt 90
lineofeggs 3
rt 180
fd 20
rt 90
fd 20
lineofeggs 3
fd 20
rt 90
lineofeggs 3
rt 180
fd 20
rt 90
fd 20
egg
fd 80
lineofeggs 3
rt 90
fd 10
rt 90
fd 220
lt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 30
lineofeggs 2
fd 20
rt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 50
rt 90
lineofeggs 2
rt 180
fd 10
rt 90
fd 20
lineofeggs 2
fd 20
egg
rt 90
fd 10
rt 90
fd 150
rt 180
lineofeggs 6
fd 70
lineofeggs 2
fd 50
lineofeggs 6
```

Challenge 19

#Coded by eash#

```
import telnetlib
payload=''.join( map(chr, [49, 24, 44, 52, 25, 32, 44, 34, 19, 84, 44, 21, 53, 23, 44, 53, 21]) )
```

exploit=telnetlib.Telnet('hackyeaster.hacking-lab.com', 1234);

exploit.write(payload+'\n')
exploit.interact()

Challenge 20

```
######################
#     Mask.txt content
######################
.?1.
.?1?1.
.?1?1?1.
.?1?1?1?1.
.?1?1?1?1?1.
.?1?1?1?1?1?1.
.?1?1?1?1?1?1?1.
.?1?1?1?1?1?1?1?1.
.?1?1?1?1?1?1?1?1?1.
```

Challenge 22

```
#Code by eash#
#Credits for AJ Alt on https://github.com/ajalt/python-sha1#
import struct
import io
import sys


hash  = ['fad202a6e094dd8f1d63da8bdf85b3ba099971d3',
         'f71e1b0b9b3a57d864c2e9f7bd6dd90f66b5a7d6',
         '84c6bcb681b79b690b53f9f3a8ba24e1e970d348',
         '0d6bb0df8918168798ce6b770014aeb81ac6ce76']
try:
    range = xrange
except NameError:
    pass

def _left_rotate(n, b):
    """Left rotate a 32-bit integer n by b bits."""
    return ((n << b) | (n >> (32 - b))) & 0xffffffff

def _process_chunk(chunk, h0, h1, h2, h3, h4):
    """Process a chunk of data and return the new digest variables."""
    assert len(chunk) == 64

    w = [0] * 80

    # Break chunk into sixteen 4-byte big-endian words w[i]
    for i in range(16):
        w[i] = struct.unpack(b'>I', chunk[i*4:i*4 + 4])[0]

    # Extend the sixteen 4-byte words into eighty 4-byte words
    for i in range(16, 80):
        w[i] = _left_rotate(w[i-3] ^ w[i-8] ^ w[i-14] ^ w[i-16], 1)

    # Initialize hash value for this chunk
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4
```

```
    for i in range(80):
        if 0 <= i <= 19:
            # Use alternative 1 for f from FIPS PB 180-1 to avoid bitwise not
            f = d ^ (b & (c ^ d))
            k = 0x5A827999
        elif 20 <= i <= 39:
            f = b ^ c ^ d
            k = 0x6ED9EBA1
        elif 40 <= i <= 59:
            f = (b & c) | (b & d) | (c & d)
            k = 0x8F1BBCDC
        elif 60 <= i <= 79:
            f = b ^ c ^ d
            k = 0xCA62C1D6

        a, b, c, d, e = ((_left_rotate(a, 5) + f + e + k + w[i]) & 0xffffffff,
                         a, _left_rotate(b, 30), c, d)

    # Add this chunk's hash to result so far
    h0 = (h0 + a) & 0xffffffff
    h1 = (h1 + b) & 0xffffffff
    h2 = (h2 + c) & 0xffffffff
    h3 = (h3 + d) & 0xffffffff
    h4 = (h4 + e) & 0xffffffff

    return h0, h1, h2, h3, h4

class Sha1Hash(object):
    """A class that mimics that hashlib api and implements the SHA-1 algorithm."""

    name = 'python-sha1'
    digest_size = 20
    block_size = 64

    def __init__(self):
        # Initial digest variables
        # Changed following the HE Hints
        self._h = (
            0x10325476,
            0x98BADCFE,
            0xEFCDAB89,
            0x67452301,
            0x0F1E2D3C,
        )
        # bytes object with 0 <= len < 64 used to store the end of the message
        # if the message length is not congruent to 64
        self._unprocessed = b''
        # Length in bytes of all data that has been processed so far
        self._message_byte_length = 0

    def update(self, arg):
        """Update the current digest.

        This may be called repeatedly, even after calling digest or hexdigest.

        Arguments:
            arg: bytes, bytearray, or BytesIO object to read from.
        """
        if isinstance(arg, (bytes, bytearray)):
            arg = io.BytesIO(arg)

        # Try to build a chunk out of the unprocessed data, if any
        chunk = self._unprocessed + arg.read(64 - len(self._unprocessed))

        # Read the rest of the data, 64 bytes at a time
        while len(chunk) == 64:
            self._h = _process_chunk(chunk, *self._h)
            self._message_byte_length += 64
            chunk = arg.read(64)

        self._unprocessed = chunk
        return self


    def digest(self):
        """Produce the final hash value (big-endian) as a bytes object"""
```

```
        return b''.join(struct.pack(b'>I', h) for h in self._produce_digest())

    def hexdigest(self):
        """Produce the final hash value (big-endian) as a hex string"""
        return '%08x%08x%08x%08x%08x' % self._produce_digest()

    def _produce_digest(self):
        """Return finalized digest variables for the data processed so far."""
        # Pre-processing:
        message = self._unprocessed
        message_byte_length = self._message_byte_length + len(message)

        # append the bit '1' to the message
        message += b'\x80'

        # append 0 <= k < 512 bits '0', so that the resulting message length (in bytes)
        # is congruent to 56 (mod 64)
        message += b'\x00' * ((56 - (message_byte_length + 1) % 64) % 64)

        # append length of message (before pre-processing), in bits, as 64-bit big-endian
integer
        message_bit_length = message_byte_length * 8
        message += struct.pack(b'>Q', message_bit_length)

        # Process the final chunk
        # At this point, the length of the message is either 64 or 128 bytes.
        h = _process_chunk(message[:64], *self._h)
        if len(message) == 64:
            return h
        return _process_chunk(message[64:], *h)


def sha1(data):
    """SHA-1 Hashing Function

    A custom SHA-1 hashing function implemented entirely in Python.

    Arguments:
        data: A bytes or BytesIO object containing the input message to hash.

    Returns:
        A hex SHA-1 digest of the input message.
    """
    return Sha1Hash().update(data).hexdigest()

if __name__ == '__main__':
    # Imports required for command line parsing. No need for these elsewhere
    import argparse
    import sys
    import os
    from datetime import datetime
    import fileinput

    start_time = datetime.now()
    file=open("rockyou.txt","r");
    #file=open("crackstation-human-only","r");
    names=file.read().split("\n")
    for line in names:
        strSHA1 =  sha1(line)
        for i in range(4):
                if strSHA1 == hash[i] :
                    print 'HASH ' + str(i+1)  + ' SHA1=' + line
                    end_time = datetime.now()
                    print('Duration: {}'.format(end_time - start_time))
```

Challenge 23

```
#Coded by eash#
from PIL import Image

#Hidden image extracted from heizohack.bmp
```

```
img = Image.open("out.png")

#Converting image  to RGBA format
pixels = img.convert('RGBA').load()
w, h = img.size

#Print image infor & size
print 'Image info: '
print img.info
print "Width= " + str(w) + "   "  + "Height= " +  str(h)

#Init msg vector
msg = []

#Starting reading line by line
for y in xrange(h):
        for x in xrange(w):
            r, g , b, a = pixels[x,y]
            rlsb = r & 1
            glsb = g & 1
            blsb = b & 1
            alsb = a & 1
            check =  rlsb ^ glsb ^ blsb ^ alsb
            if check == 1:
                   msg.append(str(rlsb))

mac =  ''.join(msg)
#Get Result in ASCii
m = ''.join(chr(int(mac[i:i+8], 2)) for i in xrange(0, len(mac), 8))

#print Solution
print "Super, the Egg-O-Matic password is: " + m
```

Challenge 24

```
//KEYCRACKER
//Coded by eash#
import java.security.MessageDigest;
import java.util.Arrays;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base32;
import org.apache.commons.codec.binary.Base64;
import java.util.Random;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.lang.*;
import java.util.concurrent.TimeUnit;
import java.sql.Timestamp;
import com.google.common.base.Stopwatch;
import org.paukov.combinatorics.Factory;
import org.paukov.combinatorics.Generator;
import org.paukov.combinatorics.ICombinatoricsVector;
import java.util.List;
import java.util.Arrays;
import java.lang.String;



public class KEYCRACKER {
  static final String IV = "hackyeasterisfun";
  static final String url = "http://hackyeaster.hacking-lab.com/";
```

```
  //Ticket created using http://hackyeaster.hacking-lab.com/hackyeaster/crunchly/crunchly.html
with URL http://hackyeaster.hacking-lab.com/
  static final String ticket =
"+bmSlmNc6Nw08rPHt5Lsjydrl1cg6D1Gn23703biSv4lIdmNz0Xg0rymcyjBL39HGBtNROdOafwCC0eo5WpfSkts4fX0Q
C4bJjclJ4mpGXKmq7oxVzDx3hzzko+4+lRH";


  public static void main(String [] args) {
    try {

      System.out.println("==Java Key Cracker==");

      String shortUrl = calculateShortUrl(url);
      System.out.println("URL: " + url);
      System.out.println("ShortURL: " + shortUrl);
      String fileName = "dictionary.txt";
      BufferedReader br = new BufferedReader(new FileReader(fileName));
      String line;
      System.out.println(new Timestamp(System.currentTimeMillis()));
      int amount = 0;
      final Stopwatch stopwatch = Stopwatch.createStarted();
      while ((line = br.readLine()) != null) {
            amount++;
            final String  key_full = "x" + line + line + line; // 128 bit
            String crypt = cryptTicket(url, shortUrl, key_full);
            if (crypt.equals(ticket)){
              System.out.println(new Timestamp(System.currentTimeMillis()));
              System.out.println(stopwatch.stop());
              System.out.println("EUREKA, the key is:" + line);
              System.out.println("Ticket is:" + crypt);
              System.exit(0);
            }
      }
      System.out.println("Total=" + amount);
      System.out.println(new Timestamp(System.currentTimeMillis()));
      System.out.println(stopwatch.stop());
     }catch (Exception e) {
           e.printStackTrace();
     }
  }


  public static String calculateShortUrl(String url) {
               try {
                     MessageDigest md = MessageDigest.getInstance("SHA-256");
                     md.update(url.getBytes("UTF-8"));
                     byte[] hash = md.digest();
                     byte[] part = new byte[4];
                     System.arraycopy(hash, 0, part, 0, 4);
                     String b32 = new String(new Base32().encode(part), "UTF-8");
                     b32 = b32.replaceAll("=", "");
                     return "http://crunch.ly/" + b32;
               } catch (Exception e) {
                     e.printStackTrace();
               }
               return null;
  }

  public static String cryptTicket(String url, String shortUrl, String key_full) {
               try {
                     Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                     SecretKeySpec keySpec = new SecretKeySpec(key_full.getBytes("UTF-8"),
"AES");
                     cipher.init(Cipher.ENCRYPT_MODE, keySpec, new
IvParameterSpec(IV.getBytes("UTF-8")));
                     String plain = new String(Base64.encodeBase64(url.getBytes("UTF-8")),
"UTF-8");
                     plain += "@" + new String(Base64.encodeBase64(shortUrl.getBytes("UTF-
8")), "UTF-8");
                     byte[] crypted = cipher.doFinal(plain.getBytes("UTF-8"));
                     return Base64.encodeBase64String(crypted);
               } catch (Exception e) {
                     e.printStackTrace();
               }
               return null;
  }
```

```
}

//EVIL
//Coded by eash#
import java.security.MessageDigest;
import java.util.Arrays;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base32;
import org.apache.commons.codec.binary.Base64;
import java.util.Random;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.lang.*;
import java.util.concurrent.TimeUnit;
import java.sql.Timestamp;
import com.google.common.base.Stopwatch;
import org.paukov.combinatorics.Factory;
import org.paukov.combinatorics.Generator;
import org.paukov.combinatorics.ICombinatoricsVector;
import java.util.List;
import java.util.Arrays;
import java.lang.String;



public class EVIL {

  static final String KEY =  "tKguf";  //[[CENSORED!!!]]
  static final String key_full = "x" + KEY + KEY + KEY; // 128 bit
  static final String IV = "hackyeasterisfun";
  static final String evil  = "http://evileaster.com";

  public static void main(String [] args) {
    try {

      System.out.println("==Java Evil URL Generator==");
      ICombinatoricsVector<Character> initialVector = Factory.createVector(
                new Character[]{'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',
                '0','1', '2', '3', '4', '5', '6', '7', '8', '9',
                'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q',
                'R','T','U','V','W','X','Y','Z'});


      String crunch = "IU66SMI";
      String line;
      System.out.println(new Timestamp(System.currentTimeMillis()));
      int amount = 0;
      final Stopwatch stopwatch = Stopwatch.createStarted();
      Generator<Character> gen =
Factory.createPermutationWithRepetitionGenerator(initialVector, 7);

      // Print all possible combinations

      for (ICombinatoricsVector<Character> combination : gen) {
           List  list = combination.getVector();
           String url = Arrays.toString(list.toArray());
           String saida = url.replaceAll("[^a-zA-Z0-9]","");
           String url2 = evil + "/" + saida ;
           String shortUrl1 = calculateShortUrl(url2);
           amount++;
           if (shortUrl1.equals(crunch)){
                  System.out.println(new Timestamp(System.currentTimeMillis()));
                  System.out.println(stopwatch.stop());
                  System.out.println("EUREKA, the url is:" + url2);
```

```
                System.out.println("Elapsed time in Minutes ==> " +
stopwatch.elapsed(TimeUnit.MINUTES));
                System.exit(0);
           }
           if (amount % 10000000 == 0){
              System.out.println(amount); System.out.println(url2);

           }

      }

      System.out.println("Total=" + amount);
      System.out.println(new Timestamp(System.currentTimeMillis()));
      System.out.println(stopwatch.stop());
     }catch (Exception e) {
           e.printStackTrace();
     }
  }


  public static String calculateShortUrl(String url) {
               try {
                     MessageDigest md = MessageDigest.getInstance("SHA-256");
                     md.update(url.getBytes("UTF-8"));
                     byte[] hash = md.digest();
                     byte[] part = new byte[4];
                     System.arraycopy(hash, 0, part, 0, 4);
                     String b32 = new String(new Base32().encode(part), "UTF-8");
                     b32 = b32.replaceAll("=", "");
                     //return "http://crunch.ly/" + b32;
                     return b32;
               } catch (Exception e) {
                     e.printStackTrace();
               }
               return null;
  }

  public static String cryptTicket(String url, String shortUrl, String key_full) {
               try {
                     Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                     SecretKeySpec keySpec = new SecretKeySpec(key_full.getBytes("UTF-8"),
"AES");
                     System.out.println("KEY: " + key_full.getBytes("UTF-8"));
                     cipher.init(Cipher.ENCRYPT_MODE, keySpec, new
IvParameterSpec(IV.getBytes("UTF-8")));
                     System.out.println("IV: " + IV.getBytes("UTF-8"));
                     String plain = new String(Base64.encodeBase64(url.getBytes("UTF-8")),
"UTF-8");
                     System.out.println("Ticket: " + plain);
                     plain += "@" + new String(Base64.encodeBase64(shortUrl.getBytes("UTF-
8")), "UTF-8");
                     System.out.println("Ticket2: " + plain);
                     byte[] crypted = cipher.doFinal(plain.getBytes("UTF-8"));
                     return Base64.encodeBase64String(crypted);
               } catch (Exception e) {
                     e.printStackTrace();
               }
               return null;
  }

}


//TICKET
//Coded by eash#
import java.security.MessageDigest;
import java.util.Arrays;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base32;
import org.apache.commons.codec.binary.Base64;
import java.util.Random;
```

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.lang.*;
import java.util.concurrent.TimeUnit;
import java.sql.Timestamp;
import com.google.common.base.Stopwatch;
import org.paukov.combinatorics.Factory;
import org.paukov.combinatorics.Generator;
import org.paukov.combinatorics.ICombinatoricsVector;
import java.util.List;
import java.util.Arrays;
import java.lang.String;



public class TICKET {
  static final String KEY =  "tKguF";  //[[CENSORED!!!]]
  static final String key_full = "x" + KEY + KEY + KEY; // 128 bit
  static final String IV = "hackyeasterisfun";
  static final String url = "http://evileaster.com/lDdvCxa";

  public static void main(String [] args) {
    try {

      System.out.println("==Java Ticket Creator==");
      System.out.println("Evil URL is" + url);
      String shortUrl = calculateShortUrl(url);
      System.out.println("Short URL=" + shortUrl);
      String crypt = cryptTicket(url, shortUrl, key_full);
      System.out.println("Valid Ticket is:" + crypt);
     }catch (Exception e) {
           e.printStackTrace();
     }
  }


  public static String calculateShortUrl(String url) {
               try {
                     MessageDigest md = MessageDigest.getInstance("SHA-256");
                     md.update(url.getBytes("UTF-8"));
                     byte[] hash = md.digest();
                     byte[] part = new byte[4];
                     System.arraycopy(hash, 0, part, 0, 4);
                     String b32 = new String(new Base32().encode(part), "UTF-8");
                     b32 = b32.replaceAll("=", "");
                     return "http://crunch.ly/" + b32;
               } catch (Exception e) {
                     e.printStackTrace();
               }
               return null;
  }

  public static String cryptTicket(String url, String shortUrl, String key_full) {
               try {
                     Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                     SecretKeySpec keySpec = new SecretKeySpec(key_full.getBytes("UTF-8"),
"AES");
                     cipher.init(Cipher.ENCRYPT_MODE, keySpec, new
IvParameterSpec(IV.getBytes("UTF-8")));
                     String plain = new String(Base64.encodeBase64(url.getBytes("UTF-8")),
"UTF-8");
                     plain += "@" + new String(Base64.encodeBase64(shortUrl.getBytes("UTF-
8")), "UTF-8");
                     byte[] crypted = cipher.doFinal(plain.getBytes("UTF-8"));
                     return Base64.encodeBase64String(crypted);
               } catch (Exception e) {
                     e.printStackTrace();
               }
               return null;
  }

}
```