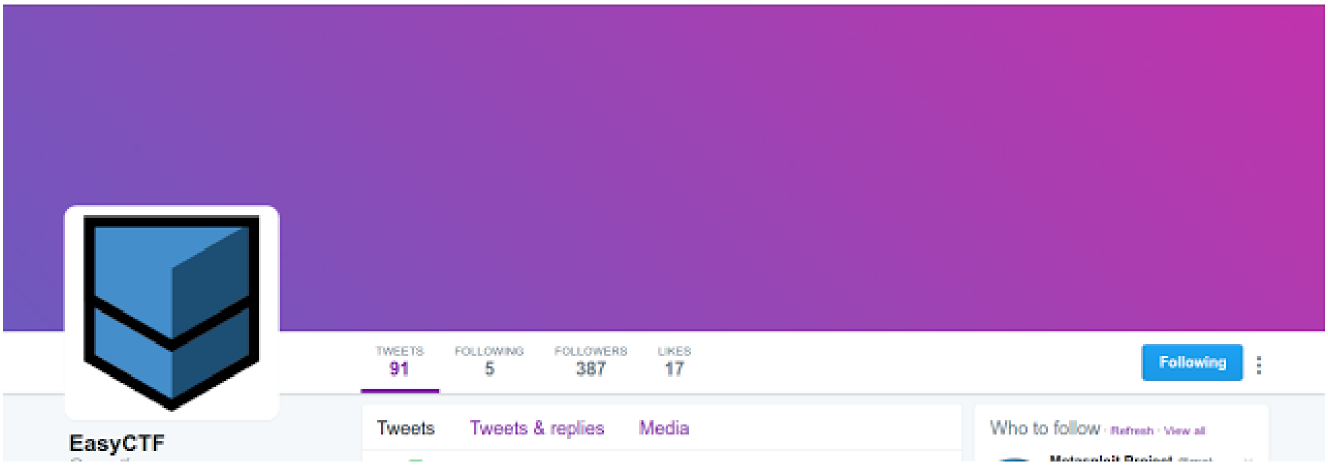
(http://0xbugsbunny.blogspot.tw/)

EasyCTF 2017{Tasks_WriteUps}

1:00:00 PM Iheb Ben Salem 0 Comments



(https://3.bp.blogspot.com/-QCgFTqxYkVs/WNFc7tDMMtI/AAAAAAAABcQ/ObzKyOP38fo5dlLLop2d1BrAbrFNYL4EwCPcB/s1600/Screenshot%2B-%2B03212017%2B-%2B06%253A03%253A12%2BPM.png)

> Hash on Hash , Cryptography , 100 pt –solved by chouaib(cho)

Task

There's a lot of hex strings here. Maybe they're hiding a message?

Hint: Thankfully you can solve this without even using a website

HexStrings file (https://drive.google.com/file/d/0B0FucB-or3U5ZlI2N3NTVTgwMFU/view?usp=sharing)

The first thing that We have hex strings file and we noticed it's MD5 hashes and every 256 char MD5's means one letter so we can make it easy and Solved with https://hashkiller.co.uk/md5-decrypter.aspx (https://hashkiller.co.uk/md5-decrypter.aspx)

This is what we got !

The first thing that Im far too lazy to put anything meaningful here. Instead, here's some information about what you just solved. The MD5 algorithm is a widely used hash function producing a 128-bit hash value. Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities. It can still be used as a checksum to verify data integrity, but only against unintentional corruption. Like most hash functions, MD5 is neither encryption nor encoding. It can be cracked by brute-force attack and suffers from extensive vulnerabilities as detailed in the security section below. MD5 was designed by Ronald Rivest in 1991 to replace an earlier hash function MD4.[3] The source code in RFC 1321 contains a "by attribution" RSA license. The abbreviation "MD" stands for "Message Digest." The security of the MD5 has been severely compromised, with its weaknesses having been exploited in the field, most infamously by the Flame malware in 2012. The CMU Software Engineering Institute considers MD5 essentially "cryptographically broken and unsuitable for further use". easyctf{1_h0p3_y0u_d1dn7_d0_7h47_by_h4nd}

the flag is : easyctf{1_h0p3_y0u_d1dn7_d0_7h47_by_h4nd}

> RSA1, Cryptography ,50 pt – solved by Chouaib(cho)

Task

The first thing that I found somebody's notes on their private RSA! Help me crack this.

Hint: Go google RSA if you're stuck.

File = ciphertest1.txt p:

```
p: 33499881069427614105926941260008415630190853527846401734073924527104092366847259
q: 34311544767652906613104559081988349779622789386528780506962212898921316785995851
e: 65537
c: 434652482992786587120132160490031724278987822619903723162822143760418735144813869087939435323
```

The first thing that So,I wrote this humble script to solve this problem using gmpy Module :

```
import gmpy

p = 33499881069427614105926941260008415630190853527846401734073924527104092366847259
q = 34311544767652906613104559081988349779622789386528780506962212898921316785995851
e = 65537
c =
4346524829927865871201321604900317242789878226199037231628221437604187351448138690879394353236 34

f = (p-1) * (q-1)

d = gmpy.invert(e,f)

print "private key d value is : %d" % d
plain = hex(pow(c,d,n))[2:]
flag = plain.decode("hex")
print "The Flag is %s "  % flag
```

> Decode me , Cryptography ,100 pt –solved by Chouaib (cho)

Task

Someone I met today told me that they had a perfect encryption method. To prove that there is no such thing, I want you to decrypt this encrypted flag he gave me.

Hint: Simple decoding :)

The first thing that The input in the end of file is " = " what make me release it is base64. The input in the end of file is " = " what make me released it is base64 but the file file size too long so i need to decrypt it many time until i found the Flag. So I wrote a short python script to do that using the Base64 Module .

```
import base64

file = open('file.txt').read()
dec = lambda x :base64.b64decode(file)
flag = dec(file)
while 'easyctf' not in flag:
    flag = base64.b64decode(flag)
print flag
```

the flag is : easyctf{what_1s_l0v3_bby_don7_hurt_m3}

> RSA2 , Cryptography , 80 pt – solved by Chouaib(cho)

Task

The first thing that some more RSA : This time, there's no P and Q .. this :

```
n: 2669654819154578051877029177265503296931 57
```

Hint: Simple decoding :)

As you see above there's no P and Q i had only N , so i used http://factordb.com/ to get the Prime Factor of P and Q :

p = 4580704200834875508 83

q = 5828044558450224498 79

And then i wrote this script to the flag of RSA challenge also using gmpy Module that supports multiple-precision arithmetic :

```
import gmpy

n = 266965481915457805187702917726550329693157
p = 458070420083487550883
q = 582804455845022449879
e = 65537
c = 78670065603555615007383828728708393504251
f = (p-1) * (q-1)

d = gmpy.invert(e,f)
plain = hex(pow(c,d,n))[2:]
flag = plain.decode("hex")
print "The Flag is %s "  % flag
```

the is flag : flag{l0w_n_0eb6}

> RSA3, Cryptography , 135 pt –solved by chouaib(cho)

Task

We can across another message that follows the same cryptographic schema as those other RSA message. Take a look and see if you can crack it .

Hint: You might want to read up on how RSA works.

File:

```
{N : e : c}
{0x27335d21ca51432fa000ddf9e81f630314a0ef2e35d81a839584c5a7356b94934630ebfc2ef9c55b111e8c373f2db
: 0x10001
: 0x9b9c138e0d473b6e6cf44acfa3becb358b91d0ba9bfb37bf11effcebf9e0fe4a86439e8217819c273ea5c1c5acfd
```

Almost the same as the last RSA challenge there's no P and Q i had only N but as you can see clearly this time N , E , C is encrypted with base 16 (hex) so i need to take it back , and then using the http://factordb.com/ to get the Prime Factor of P and Q : This is my script to solve RSA3 :

```
import gmpy

n =
int('0x27335d21ca51432fa000ddf9e81f630314a0ef2e35d81a839584c5a7356b94934630ebfc2ef9c55b111e8c373f
e = int('0x10001',16)
c =
int('0x9b9c138e0d473b6e6cf44acfa3becb358b91d0ba9bfb37bf11effcebf9e0fe4a86439e8217819c273ea5c1c5ac

""" p and q find on FactorDB """
```

```
p =
342361685330529670826140492590369748595603665031522100150728537425895408799449253294708458641278
q =
342361685330529670826140492590369748595603665031522100150728537425895408799449253294708458641278

n=p*q
f = (p-1) * (q-1)

d = gmpy.invert(e,f)
plain = hex(pow(c,d,n))[2:]
flag = plain.decode("hex")
print "The Flag is %s "  % flag
```

The Flag is easyctf{tw0_v3ry_merrry_tw1n_pr1m35!!_417c0d}

> Flip my letters , Cryptography ,50 pt –solved by Chouaib (cho)

Task

We have given a flag :easyctf{r_wlmg_vevm_mvvw_zm_zhxrr_gzyov}

Hint : What happens if you turn the alphabet upside down?

Hummm alphabet upside down it is means decode the flag with Reverse Alphabet , I feel too lazy to write script so with simple search on Google for Atbash Cipher (http://rumkin.com/tools/cipher/atbash.php)

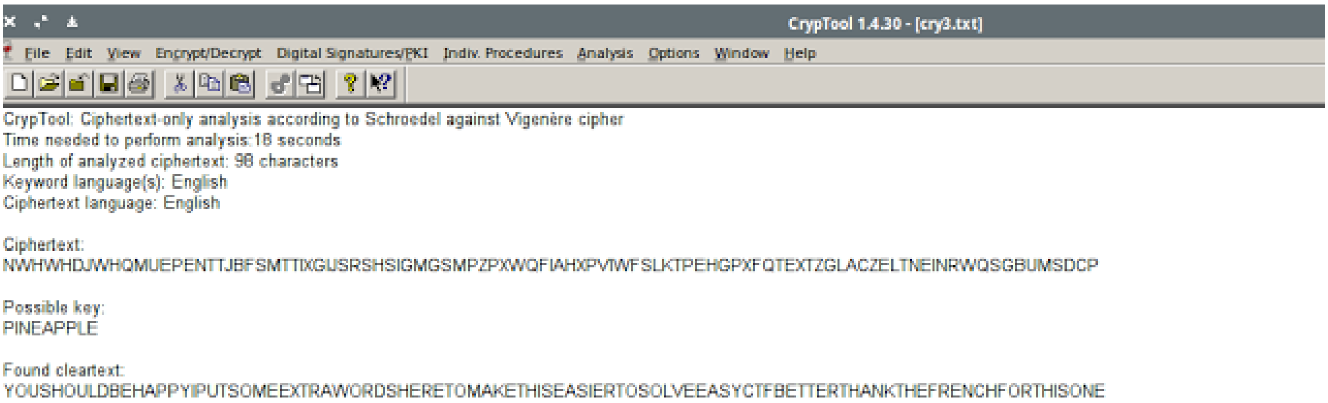
The flag is easyctf{i_dont_even_need_an_ascii_table}

> Let Me Be Frank , Cryptography ,50 pt –solved by S0ld1er

Here we have the following text given:

```
Nwh whdjwh qm uepen, T tjb fsmt tixgi jsrsh sigm gs mpzp xwqf iahxpv iw fslkt. pehgpxf{qtextz_gla
```

That might be **Vigenère cipher** , decoding the flag using Cryptool.


CrypTool 1.4.30 - [cry3.txt]
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help

CrypTool: Ciphertext-only analysis according to Schroedel against Vigenère cipher
Time needed to perform analysis:18 seconds
Length of analyzed ciphertext: 98 characters
Keyword language(s): English
Ciphertext language: English

Ciphertext:
NWHWHDJWHQMUEPENTTJBFSMTTIXGUSRSHSIGMGSMPZPXWQFIAHXPVIWFSLKTPEHGPXFQTEXTZGLACZELTNEINRWQSGBUMSDCP

Possible key:
PINEAPPLE

Found cleartext:
YOUSHOULDBEHAPPYIPUTSOMEEXTRAWORDSHERETOMAKETHISEASIERTOSOLVEEASYCTFBETTERTHANKTHEFRENCHFORTHISONE

(https://4.bp.blogspot.com/-Cv0igRDBx8I/WNGiwhUgI0I/AAAAAAAABck/QsuulVyEOyQx1YI99rHUfmehGzcRUF2VQCLcB/s1600/Screenshot%2B-%2B03212017%2B-%2B11%253A01%253A19%2BPM.png)

YOUSHOULDBEHAPPYIPUTSOMEEXTRAWORDSHERETOMAKETHISEASIERTOSOLVE
EASYCTF{BETTER_THANK_THE_FRENCH_FOR_THIS_ONE}

Lowercase the flag

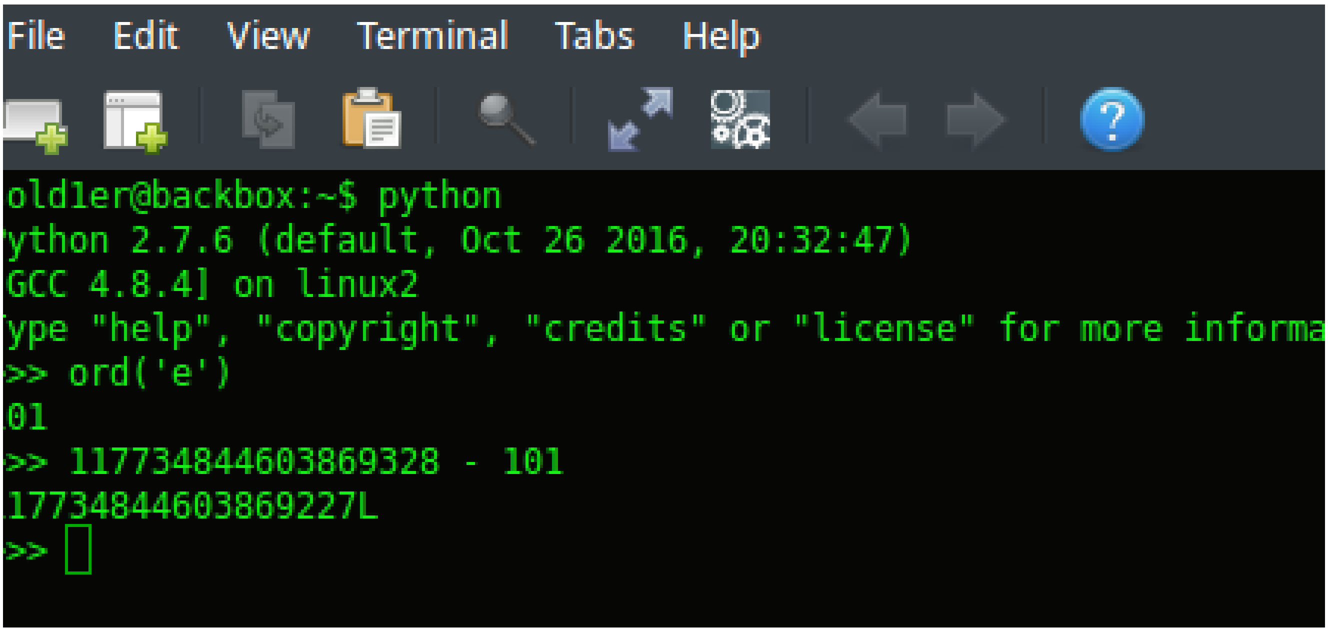
the flag is easyctf{better_thank_the_french_for_this_one}

> Phunky Python ,reverse ,50 pt –solved by S0ld1er

The goal of this task is to find the correct value of x, so the script below prints out the word “easyctf”.

```
x = 0 # REDACTED
digs =
[117734844603869328, 117734844603869324, 117734844603869342, 117734844603869348, 1177348446038693
out = ""
for letter in reversed(digs):
    out = chr(letter - x) + out
print out
```

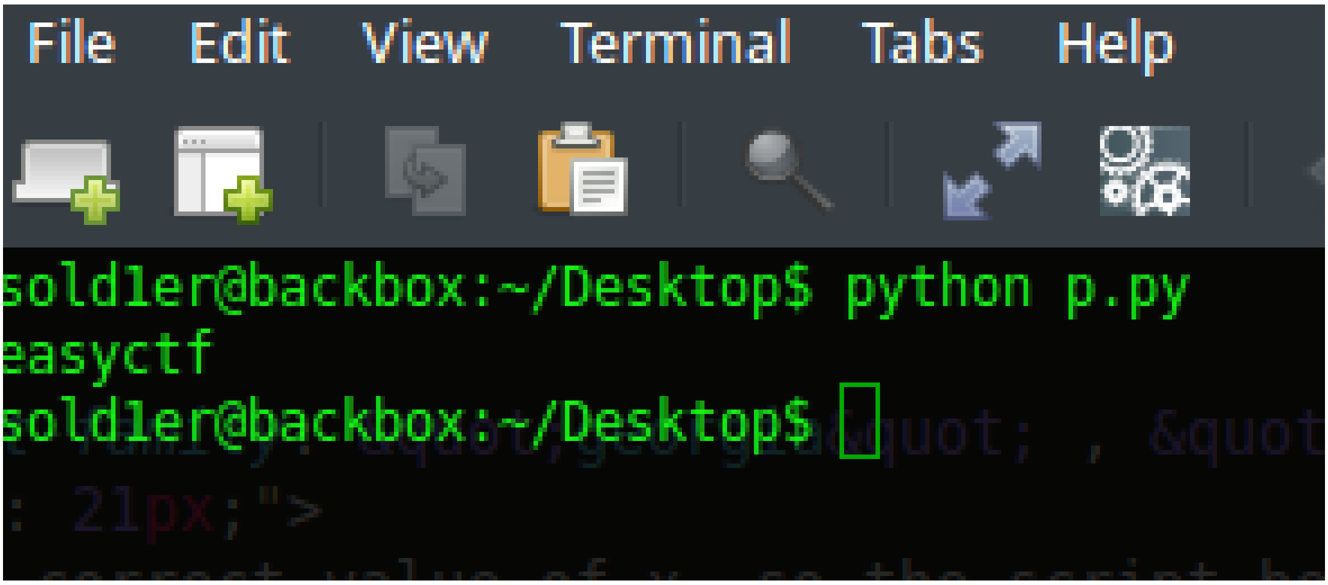
The first letter of the flag is "e", means 101 in ASCII. calculate the difference between the given value and x to get ord("e")= 101.YES, we love math :p



```
oldler@backbox:~$ python
ython 2.7.6 (default, Oct 26 2016, 20:32:47)
GCC 4.8.4] on linux2
ype "help", "copyright", "credits" or "license" for more informa
>> ord('e')
01
>> 117734844603869328 - 101
17734844603869227L
>>
```

(https://1.bp.blogspot.com/-ATXaPCidjAU/WNGlUijWfgI/AAAAAAAABc4/a-6qmwTBl-sFp5jfli_nAOgvVBwJy3ltgCLcB/s1600/Screenshot%2B-%2B03212017%2B-%2B11%253A11%253A54%2BPM.png)

Replace x with this value and check out the script



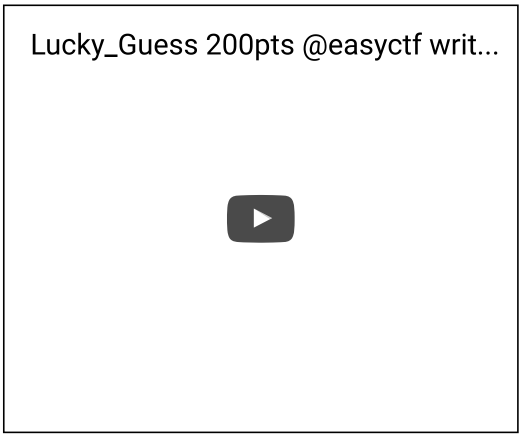
```
soldler@backbox:~/Desktop$ python p.py
easyctf
soldler@backbox:~/Desktop$
```

(https://3.bp.blogspot.com/-wYdiEYc3MhM/WNGIoz80I3I/AAAAAAAABc8/k_f-

LlwU54gfVeWbLXLlbzGSb8YeNeChACLcB/s1600/Screenshot%2B-%2B03212017%2B-%2B11%253A13%253A40%2BPM.png)

The flag is easyctf{117734844603869227L }

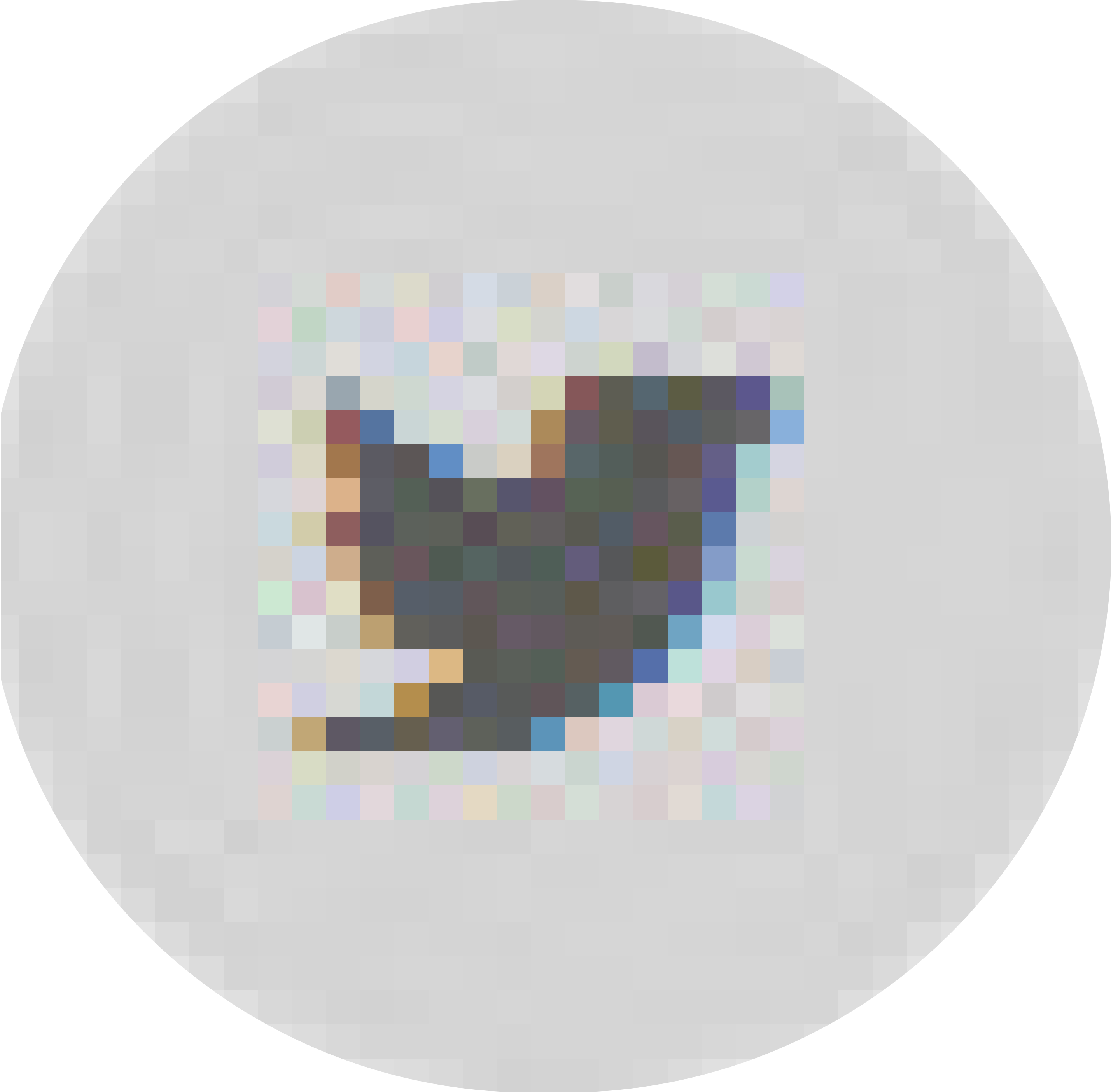
> luckyguess, reverse engineering 200 pt


Lucky_Guess 200pts @easyctf writ...

Iheb Ben Salem

Share story

(http://www.facebook.com/sharer.php?u=http://0xbugsbunny.blogspot.tw/2017/03/easyctf-2017taskswriteups.html&title=EasyCTF 2017{Tasks_WriteUps})



(http://twitter.com/share?url=http://0xbugsbunny.blogspot.tw/2017/03/easyctf-2017taskswriteups.html&title=EasyCTF 2017{Tasks_WriteUps})