Terlantarkan

Universitas Bina Nusantara

Player list :

DarkAngels

Bigby

EternalBeats

Daftar Isi

Forensic

Ingatan_MR_1

Langkah Penyelesaian:

Menggunakan plugin filescan bisa menemukan Rahasia.txt dan Pengingat.txt di dumpfile dan dilihat contentnya

```
root@kali:~/Documents/final_techno/ingatan_mr1# cat Rahasiaku.txt
Hari ini aku mencadangkannya ke penyimpanan cloud.
Semoga aman dehroot@kali:~/Documents/final_techno/ingatan_mr1# cat Pengingat.txt
Password zipnya juga udah aku hapus, supaya aman.
Aku takut rahasianya terbongkar :(root@kali:~/Documents/final_techno/ingatan_mr1#
```

Kita bisa mendapat Rahazia.zip dari iehistory

```
root@kali:~/Documents/final_techno/ingatan_mr1# cat iehistory.log
**********************************************
Process: 1608 explorer.exe
Cache type "DEST" at 0×5ea2bb1
Last modified: 2021-03-30 21:13:50 UTC+0000
Last accessed: 2021-03-30 14:13:50 UTC+0000
URL: MR@https://www.gunadarma.ac.id/
Title: Universitas Gunadarma
**********************************************
Process: 3884 iexplore.exe
Cache type "DEST" at 0×3f2cc91
Last modified: 2021-03-30 21:13:50 UTC+0000
Last accessed: 2021-03-30 14:13:50 UTC+0000
URL: MR@http://ccug.gunadarma.ac.id/
Title: Cyber Community Universitas Gunadarma
**********************************************
Process: 3884 iexplore.exe
Cache type "DEST" at 0×50a36a1
Last modified: 2021-03-30 21:13:50 UTC+0000
Last accessed: 2021-03-30 14:13:50 UTC+0000
URL: MR@https://www.gunadarma.ac.id/
Title: Universitas Gunadarma
**********************************************
Process: 3884 iexplore.exe
Cache type "DEST" at 0×51442d1
Last modified: 2021-03-30 21:13:42 UTC+0000
Last accessed: 2021-03-30 14:13:42 UTC+0000
URL: MR@https://pastebin.com/CHs5NgaK
Title: Secrettttt - Pastebin.com
root@kali:~/Documents/final_techno/ingatan_mr1#
```

https://pastebin.com/CHs5NgaK
Didapatkan link mega.nz

https://mega.nz/file/4wVxSIaC#EEwox2FIB-Dw18UBm1k2i9lJozDj-NmI9hagiv4zuVg

Didapatkan zip dengan password, password sudah dihapus jadi penulis mencari menggunakan mftparser

```
MFT entry found at offset 0x2a845400
Attribute: In Use & File
Record Number: 28681
Link count: 1


$STANDARD_INFORMATION
Creation                       Modified                       MFT Altered                    Access Date                    Type
------------------------------ ------------------------------ ------------------------------ ------------------------------ ----
2021-03-28 07:49:56 UTC+0000 2021-03-28 16:23:25 UTC+0000   2021-03-28 16:23:30 UTC+0000   2021-03-28 07:49:56 UTC+0000   Archive

$FILE_NAME
Creation                       Modified                       MFT Altered                    Access Date                    Name/Path
------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------
2021-03-28 07:49:56 UTC+0000 2021-03-28 16:23:25 UTC+0000   2021-03-28 16:23:25 UTC+0000   2021-03-28 07:49:56 UTC+0000
$Recycle.Bin\S-1-5-21-646679477-4016880729-3615170523-1002\$RVV237U.txt

$OBJECT_ID
Object ID: 114839bb-0d90-eb11-a6a8-080027af13de
Birth Volume ID: 80000000-3000-0000-0000-180000000100
Birth Object ID: 17000000-1800-0000-3254-36746b516e45
Birth Domain ID: 4b573833-3751-3333-6144-6b474271326e

$DATA
0000000000: 32 54 36 74 6b 51 6e 45 4b 57 38 33 37 51 33 33   2T6tkQnEKW837Q33
0000000010: 61 44 6b 47 42 71 32                              aDkGBq2
```

Ditemukan file.txt yang isinya semacem base64 , pertamanya kirain itu base lain tapi dimasukin sebagai password zip nya dan terbuka.

```
root@kali:~/Documents/final_techno/ingatan_mr1/zip# cat Rahasiaku.txt
technofair{R4haS1akU_h4nYA_s3bUaH_fLaG}root@kali:~/Documents/final_techno/ingatan_mr1/zip#
```

Code:

Menggunakan volatility 2.6.1 dengan windows profile dari suggested profiles

Flag:

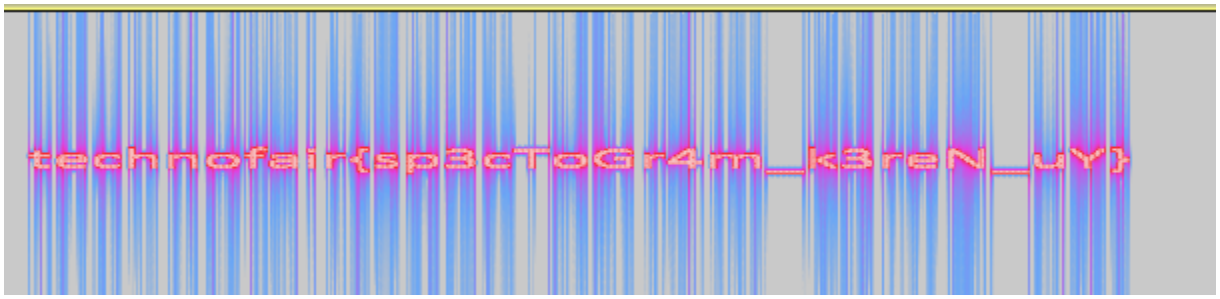
technofair{R4haS1akU_h4nYA_s3bUaH_fLaG}

Ingatan_MR_2

Langkah Penyelesaian:

Sebenarnya saya menyelesaikannya saat mengerjakan Ingatan_MR_1
Sedang lihat-lihat filescan eh ketemu .wav You Win

Pas didengerin mirip morse code tapi mirip jangkrik, karena sakit telinga coba liat spectogramnya di audacity



Code:

Menggunakan volatility 2.6.1 dengan windows profile dari suggested profiles

Flag:

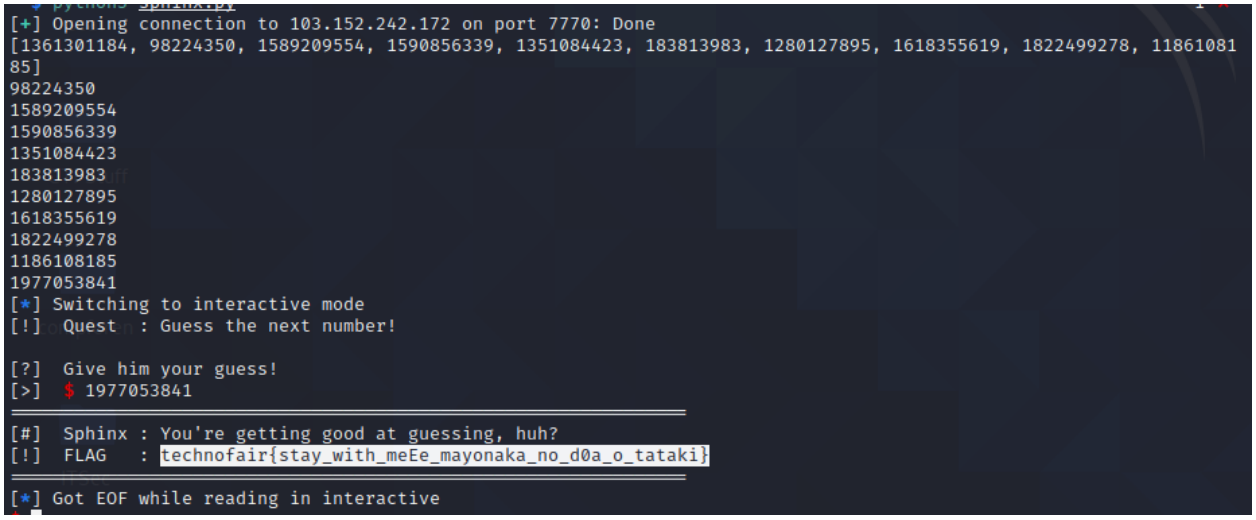
technofair{sp3cToGr4m_k3reN_uY}

Cryptography

Sphinx SPARK

Langkah Penyelesaian:

Diberikan 10 digits random numbers, lempar LCG langsung dapat…



Code:

```
from functools import reduce
from math import gcd
import pwn

class Random():
    def __init__(self, s, n, m, c):
        self.state = s
        self.n = n
        self.m = m
        self.c = c

    def next(self):
        self.state = (self.m * self.state + self.c) % self.n
        return self.state

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
```

```
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)

def modinv(b, n):
    g, x, _ = egcd(b, n)
    if g == 1:
        return x % n
    else:
        raise Exception("Modular inverse does not exist")

def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return modulus, multiplier, increment

def crack_unknown_multiplier(states, modulus, index=0):
    if index > (len(states)-1):
        raise Exception("Multiplier cannot be found")
    try:
        multiplier = (states[index + 2] - states[index + 1]) *
modinv(states[index + 1] - states[index], modulus) % modulus
    except Exception:
        index += 1
        crack_unknown_multiplier(states, modulus, index)
    multiplier = (states[index + 2] - states[index + 1]) *
modinv(states[index + 1] - states[index], modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)

def crack_unknown_modulus(states):
    diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
    zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:],
diffs[2:])]
    modulus = abs(reduce(gcd, zeroes))
    return crack_unknown_multiplier(states, modulus)

host, port = "103.152.242.172", 7770
s = pwn.remote(host, port)
s.recvuntil('The Sphinx gives you 10 numbers\n[!]  ')
numbers = s.recvuntil('\n').strip().split(b' ')
tmp = []
for n in numbers:
    tmp.append(int(n))
numbers = tmp
```

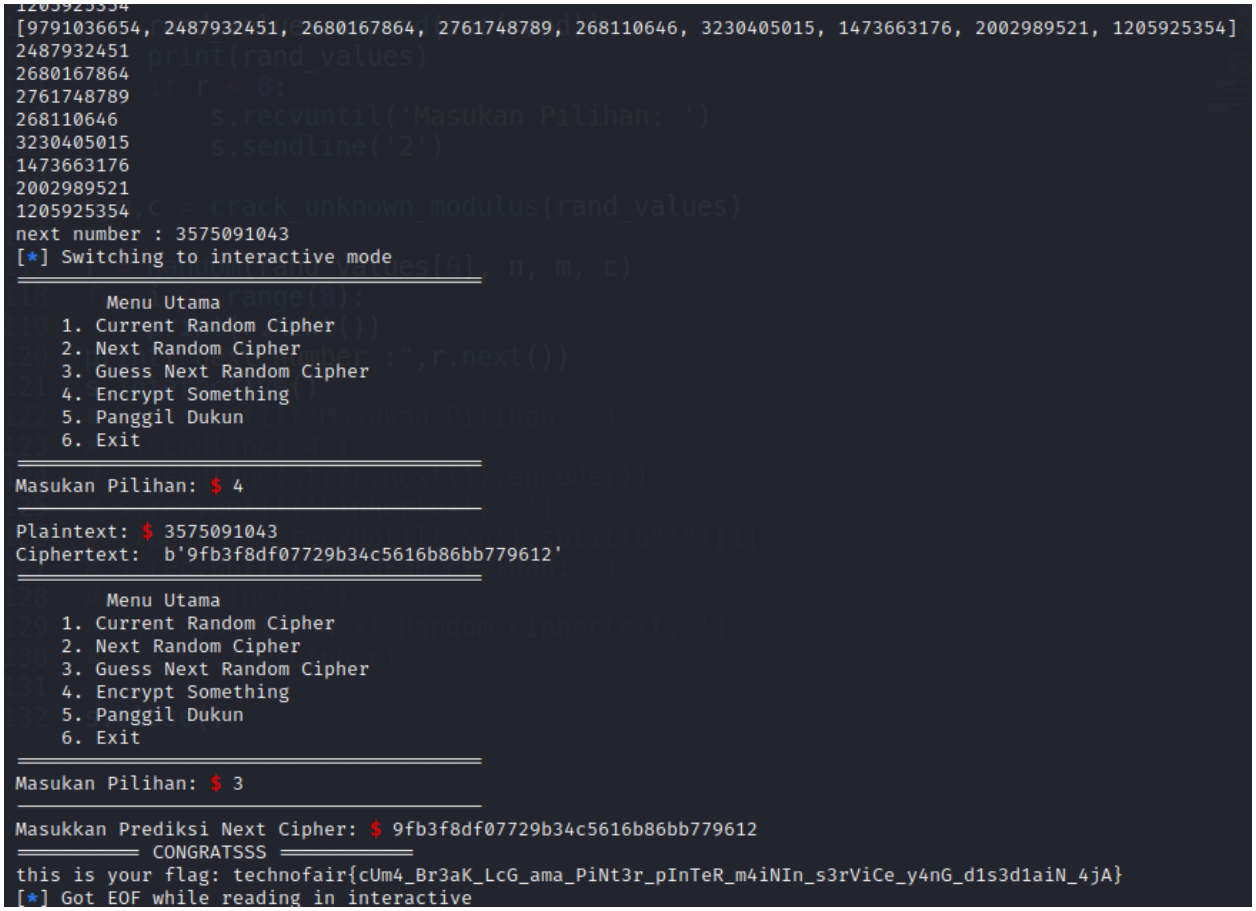
```
print(numbers)


n,m,c = crack_unknown_modulus(numbers)
r = Random(numbers[0],n,m,c)
for i in range(9):
    print(r.next())
print(r.next())
s.interactive()
s.close()
```

Flag: technofair{stay_with_meEe_mayonaka_no_d0a_o_tataki}

Legend Said

Langkah Penyelesaian:

Encryption ECB untuk angka yang di generate dari LCG, untuk LCG nya tinggal pakai cara yang sama seperti Sphinx SPARK, untuk ECB nya mainin padding, tembusin 1 block, brute character (digit) pertama.



Code:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import binascii
import pwn
from functools import reduce
from math import gcd

class Random():
    def __init__(self, s, n, m, c):
        self.state = s
        self.n = n
```

```
        self.m = m
        self.c = c

    def next(self):
        self.state = (self.m * self.state + self.c) % self.n
        return self.state

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)

def modinv(b, n):
    g, x, _ = egcd(b, n)
    if g == 1:
        return x % n
    else:
        raise Exception("Modular inverse does not exist")

def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return modulus, multiplier, increment

def crack_unknown_multiplier(states, modulus, index=0):
    if index > (len(states)-1):
        raise Exception("Multiplier cannot be found")
    try:
        multiplier = (states[index + 2] - states[index + 1]) *
modinv(states[index + 1] - states[index], modulus) % modulus
    except Exception:
        index += 1
        crack_unknown_multiplier(states, modulus, index)
    multiplier = (states[index + 2] - states[index + 1]) *
modinv(states[index + 1] - states[index], modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)

def crack_unknown_modulus(states):
    diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
    zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:],
diffs[2:])]
    modulus = abs(reduce(gcd, zeroes))
```

```
    return crack_unknown_multiplier(states, modulus)

host, port = "103.152.242.172", 9070
s = pwn.remote(host,port)

rand_values = []

for r in range(9):
    blockPadLimit = 'a'*7
    rand = ''
    for i in range(10):
       s.recvuntil('Masukan Pilihan: ')
       s.sendline('5')
       s.recv(1024)
       m = blockPadLimit+'a'*i
       s.sendline(m)
       s.recvuntil('Balasan dari dukun: ')
       target = s.recvuntil('\n').split(b"'")[1][32:]
       for j in range(10):
             prompt = s.recvuntil('Masukan Pilihan: ')
             m = pad((str(j)+rand).encode(),16)
             if b'\n' in m:
                   s.sendline('5')
                   s.recv(1024)
                   m = blockPadLimit+'a'*(i+1)
                   s.sendline(m)
                   s.recvuntil('Balasan dari dukun: ')
                   target = s.recvuntil('\n').split(b"'")[1][32:]
                   for a in range(10):
                         for b in range(10):
                               s.recvuntil('Masukan Pilihan: ')
                               s.sendline('4')
                               m = pad((str(a)+str(b)+ rand).encode(),16)
                               s.sendline(m)
                               s.recvuntil('Ciphertext:  ')
                               cipher =
s.recvuntil('\n').split(b"'")[1][:32]
                               if target == cipher:
                                     rand = str(b) + rand
                                     print(rand)
                                     break
                         else:
                               continue
```

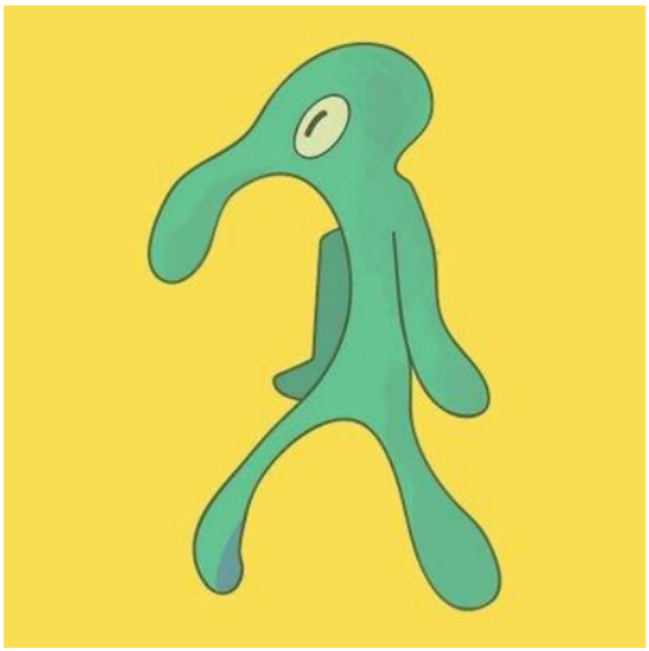
```
                        break
            else:
                  s.sendline('4')
                  p = s.recv(1024)
                  s.sendline(m)
                  s.recvuntil('Ciphertext:  ')
                  cipher = s.recvuntil('\n').split(b"'")[1][:32]
                  if target == cipher:
                        rand = str(j) + rand
                        print(rand)
                        break
    rand_values.append(int(rand))
    print(rand_values)
    if r < 8:
       s.recvuntil('Masukan Pilihan: ')
       s.sendline('2')

n,m,c = crack_unknown_modulus(rand_values)

r = Random(rand_values[0], n, m, c)
for i in range(8):
    print(r.next())
print("next number :",r.next())
s.interactive()

s.close()
```

Flag: technofair{cUm4_Br3aK_LcG_ama_PiNt3r_pInTeR_m4iNIn_s3rViCe_y4nG_d1s3d1aiN_4jA}

Dukun Said

Langkah Penyelesaian:

Reverse manually the obfuscation… yeah… just that nothing else…



technofair{i_se3_y0u_l0oK_s0_f4mIliaR_w1th_nuMpY_arR4y}

Code:

```
import numpy as np
import base64

def functionA():
    tmpVariable = np.copy(five[:,2])
    five[:,2] = three[:,2]
    three[:,2] = six[:,2]
    six[:,2] = one[:,2]
    one[:,2] = tmpVariable
    two[:] = np.rot90(two, 1)
    return one, six, three, five, two
```

```
def functionB():
    tmpVariable = np.copy(six[:,0])
    six[:,0] = three[:,0]
    three[:,0] = five[:,0]
    five[:,0] = one[:,0]
    one[:,0] = tmpVariable
    four[:] = np.rot90(four,1)
    return one, six, three, five, four

def functionC():
    tmpVariable = np.copy(four[:1])
    tmpVariable2 = np.copy(two[:1])
    tmpVariable2 = np.flip(tmpVariable2,1)
    tmpVariable3 = np.copy(three[2:])
    tmpVariable3 = np.flip(tmpVariable3,1)
    four[:1] = tmpVariable3
    three[2:] = tmpVariable2
    two[:1] = one[:1]
    one[:1] = tmpVariable
    five[:] = np.rot90(five,1)
    return one, two, three, four, five

def functionD():
    tmpVariable = np.copy(two[2:])
    tmpVariable2 = np.copy(four[2:])
    tmpVariable2 = np.flip(tmpVariable2,1)
    tmpVariable3 = np.copy(three[:1])
    tmpVariable3 = np.flip(tmpVariable3,1)
    two[2:] = tmpVariable3
    three[:1] = tmpVariable2
    four[2:] = one[2:]
    one[2:] = tmpVariable
    six[:] = np.rot90(six,1)
    return one, four, three, two, six

def functionE():
    tmpVariable = np.copy(six[:1])
    tmpVariable2 = np.copy(five[2:])
    tmpVariable2 = np.flip(tmpVariable2)
    tmpVariable3 = np.flip(tmpVariable)
    six[:1] = four[:,2]
    four[:,2] = tmpVariable2
    five[2:] = two[:,0]
```

```
    two[:,0] = tmpVariable3
    one[:] = np.rot90(one,1)
    return two, five, four, six, one

def functionF():
    tmpVariable = np.copy(five[:1])
    tmpVariable2 = np.copy(two[:,2])
    tmpVariable2 = np.flip(tmpVariable2)
    tmpVariable3 = np.copy(four[:,0])
    tmpVariable3 = np.flip(tmpVariable3)
    five[:1] = tmpVariable3
    four[:,0] = six[2:]
    six[2:] = tmpVariable2
    two[:,2] = tmpVariable
    three[:] = np.rot90(three,1)
    return two, six, four, five, three

def functionEncrypt():
    for i in key:
       if i=='A':
             functionA()
       elif i=='B':
             functionB()
       elif i =='C':
             functionC()
       elif i =='D':
             functionD()
       elif i =='E':
             functionE()
       elif i =='F':
             functionF()
       else:
             continue
    ret = ''
    for i in range(6):
       for j in range(9):
             ret += chr(reshapedtedList[i][j])
    return ret

key = "FEAABEADDFEAADEA"[::-1]
cipher = open('result.enc','rb').read()
add = 8
pad = 54-add
```

```
partOfFlag = []
plain = ''
for a in range(0, len(cipher)//54):
    for b in range(a*54, (a+1)*54):
       partOfFlag.append(cipher[b])
    partOfFlag = np.array(partOfFlag)
    reshapedtedList = partOfFlag.reshape(6,9)
    one = reshapedtedList[0].reshape((3,3))
    two = reshapedtedList[1].reshape((3,3))
    three = reshapedtedList[5].reshape((3,3))
    four = reshapedtedList[2].reshape((3,3))
    five = reshapedtedList[3].reshape((3,3))
    six = reshapedtedList[4].reshape((3,3))
    plain += functionEncrypt()
    partOfFlag = []

print(plain[:4])
target = "JVBE" #pdf file signature first 3 bytes encoded with base64
if plain[:4] == target:
    print("found it")
    with open('flag.pdf','wb') as handle:
       handle.write(base64.b64decode(plain.encode()+b'=='))
```

Flag: technofair{i_se3_y0u_l0oK_s0_f4mIliaR_w1th_nuMpY_arR4y}